Segment Routing Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 24.1.x, 24.2.x, 24.3.x

First Published: 2024-04-12

Last Modified: 2024-09-02

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

CONTENTS

PREFACE

Preface xiii

Communications, Services, and Additional Information xiii

Changes to This Document xiii

CHAPTER 1

New and Changed Information for Segment Routing Features 1

New and Changed Segment Routing Features-24.x.x 1

CHAPTER 2

YANG Data Models for Segment Routing Features 5

Using YANG Data Models 5

CHAPTER 3

Configure Segment Routing over IPv6 (SRv6) with Micro-SIDs 7

Segment Routing over IPv6 Overview 8

SRv6 Micro-Segment (uSID) 10

SRv6 Head-End Behaviors 11

SRv6 Endpoint Behaviors 13

SRv6 uSID Terminology 15

SRv6 uSID Carrier Format 15

SRv6 uSID Allocation Within a uSID Block 16

SRv6 Endpoint Behaviors Associated with uSID 20

SRv6 uSID in Action - Example 20

Usage Guidelines and Limitations 25

Configuring SRv6 27

Configuring SRv6 under IS-IS 32

Configuring SRv6 Flexible Algorithm under IS-IS 32

Configuring SRv6 Locator Prefix Summarization 34

Configuring TI-LFA with SRv6 IS-IS 35

Configuring SRv6 IS-IS Microloop Avoidance 37
Configuring SRv6 BGP-Based Services 38
SRv6 Services: IPv4 L3VPN 41
SRv6 Services: IPv6 L3VPN 53
SRv6 Services: IPv4 BGP Global 64
SRv6 Services: IPv6 BGP Global 69
BGP Signaling for co-existence of IP routes with or without SRv6 SID 75
Configure BGP Signaling over SRv6 Core 76
SRv6 Services: IPv4 L3VPN Active-Standby Redundancy using Port-Active Mode 80
SRv6 Services for L3VPN Active-Standby Redundancy using Port-Active Mode: Operation 81
Configure SRv6 Services L3VPN Active-Standby Redundancy using Port-Active Mode 81
SRv6 Services: IPv4 L3VPN Active-Active Redundancy 84
SRv6 Services: L3 EVPN 85
SRv6 Services: L2 and L3 Services with Remote SIDs from W-LIB 89
SRv6-Services: L3 Services with Local SIDs from W-LIB 91
SRv6/MPLS L3 Service Interworking Gateway 96
L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway 102
L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway 105
SRv6/MPLS Dual-Connected PE 108
SRv6 Provider Edge (PE) Lite Support 110
Configuring Explicit End.DT46 SRv6 SIDs 116
Configuring Explicit SRv6 uSID Allocation Start Range 118
Configure Seamless Bidirectional Forwarding Detection 120
Configuring the SBFD Reflector 122
SRv6 SID Information in BGP-LS Reporting 123
Full-Replace Migration to SRv6 Micro-SID 124
SRv6 Traffic Accounting 128
Usage Guidelines and Limitations 131
Configure SRv6 Traffic Accounting 132
Path Maximum Transmission Unit (MTU) Discovery for SRv6 Encapsulated Packets 135
Usage Guidelines and Limitations 136

CHAPTER 4 **Configure Segment Routing over IPv6 (SRv6) with Full-Length SIDs** 139
Segment Routing over IPv6 Overview 140

Configuring SRv6 under IS-IS 148
Configuring SRv6 IS-IS TI-LFA 149
Configuring SRv6 IS-IS Microloop Avoidance 152
Configuring SRv6 IS-IS Flexible Algorithm 153
SRv6 Services: IPv4 L3VPN 155
SRv6 Services: IPv6 L3VPN 159
SRv6 Services: L3VPN VPNv4 Active-Standby Redundancy using Port-Active Mode 167
SRv6 Services for L3VPN Active-Standby Redundancy using Port-Active Mode: Operation 167
Configure SRv6 Services L3VPN Active-Standby Redundancy using Port-Active Mode 168
Configuration Example 168
Running Configuration 168
Verification 169
SRv6 Services: L3VPN VPNv4 Active-Active Redundancy 171
SRv6 Services: BGP Global IPv6 172
SRv6 Services: BGP Global IPv6 177
SRv6 Services on EVPN E-Line 183
Benefits of SRv6 Services on EVPN E-line 184
Restrictions for SRv6 Services on EVPN E-Line 184
Configure SRv6 Services on EVPN E-Line 185
SRv6 SID Information in BGP-LS Reporting 187

CHAPTER 5 **Configure SRv6 Traffic Engineering** 189
SRv6-TE Overview 192
Usage Guidelines and Limitations 192
SRv6-TE Policy 194
Explicit Paths 195
Configure SRv6-TE Policy with Explicit Path 195
SRv6-TE Explicit Segment List SID Validation 199
Dynamic Paths 201
Optimization Objectives 201
Constraints 203
Configure SRv6 Policy with Dynamic Path 203
Named Interface Link Admin Groups and SRv6-TE Affinity Maps 206
Segment Protection-Type Constraint 208

SRv6 Flexible Algorithm 209
Automated Steering 213
Protocols 214
Path Computation Element Protocol 214
Configure the Head-End Router as PCEP PCC 215
SR-TE Application Programming Interface (API) 221
Reporting of SR-TE Policies Using BGP- Link State 230
Restrictions to Reporting of SRTE Policies using BGP-LS 231
Configure Reporting of SRTE Policies using BGP-LS 231

CHAPTER 6 **Configure Segment Routing Global Block and Segment Routing Local Block** 239
About the Segment Routing Global Block 239
About the Segment Routing Local Block 241
Understanding Segment Routing Label Allocation 242
Setup a Non-Default Segment Routing Global Block Range 245
Setup a Non-Default Segment Routing Local Block Range 246

CHAPTER 7 **Configure Segment Routing for IS-IS Protocol** 249
Enabling Segment Routing for IS-IS Protocol 249
Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface 252
Overriding MPLS Imposition (IP-to-MPLS) via Service Layer API (SL-API) 255
Configuring an Adjacency SID 263
Manually Configure a Layer 2 Adjacency SID 266
IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability 269
Prefix Attribute Flags 269
IPv4 and IPv6 Source Router ID 271
Configuring Prefix Attribute N-flag-clear 271
IS-IS Unreachable Prefix Announcement 273
Configuration Steps 274
IS-IS Partition Detection and Leakage of Specific Route Advertisement 275
Conditional Prefix Advertisement 279

CHAPTER 8 **Configure Segment Routing for OSPF Protocol** 283
Enabling Segment Routing for OSPF Protocol 283

Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface 285

CHAPTER 9 **Configure Segment Routing for BGP** 289

Segment Routing for BGP 289

Configure BGP Prefix Segment Identifiers 290

Segment Routing Egress Peer Engineering 291

Configure Segment Routing Egress Peer Engineering 292

Configuring Manual BGP-EPE Peering SIDs 294

Advertising EPE-Enabled BGP Neighbors via BGP-LU 297

IP Lookup Fallback for BGP Peering (EPE) Segments 322

Configure BGP Link-State 327

Configurable Filters for IS-IS advertisements to BGP-Link State 332

Configure Filters for IS-IS Advertisements to BGP-LS 332

Configure BGP Proxy Prefix SID 334

BGP Best Path Computation using SR Policy Paths 337

SRv6 Double Recursion for Multi-Layer BGP Underlay 343

Usage Guidelines and Limitations for SRv6 Double Recursion 345

Configure SRv6 Double Recursion for Multi-Layer BGP Underlay 346

CHAPTER 10 **Configure SR-TE Policies** 349

SR-TE Policy Overview 350

Usage Guidelines and Limitations 350

Instantiation of an SR Policy 352

On-Demand SR Policy – SR On-Demand Next-Hop 352

SR-ODN Configuration Steps 353

Configuring SR-ODN: Examples 356

Manually Provisioned SR Policy 364

PCE-Initiated SR Policy 364

Cumulative Metric Bounds (Delay-Bound Use-Case) 364

SR-TE Policy Path Types 367

Dynamic Paths 367

Optimization Objectives 367

Constraints 368

Configure SR Policy with Dynamic Path 370

Anycast SID-Aware Path Computation 371
Explicit Paths 377
SR-TE Policy with Explicit Path 377
Configuring Explicit Path with Affinity Constraint Validation 381
Protocols 383
Path Computation Element Protocol 383
Configure the Head-End Router as PCEP PCC 384
BGP SR-TE 388
Configure BGP SR Policy Address Family at SR-TE Head-End 388
Traffic Steering 390
Automated Steering 390
Color-Only Automated Steering 391
Setting the Color-Only Flag 392
Address-Family Agnostic Automated Steering 393
Per-Flow Automated Steering 394
Using Binding Segments 399
Stitching SR-TE Polices Using Binding SID: Example 400
Static Route Traffic-Steering using SR-TE Policy 404
Label Distribution Protocol (LDP) over Segment Routing Traffic Engineering (SR-TE) Policy 405
Autoroute Include 408
SR-TE Automated Steering Without BGP Prefix Path Label 413
Use Case: Centralized BGP EPE for 6PE in an SR-MPLS Network 415
Enabling SR-TE with Next-Hop Independent Scaling Optimization 421
Usage Guidelines and Limitations for Next-Hop Independent Scaling Optimization 421
Miscellaneous 422
Segment Routing Encapsulation Object Optimization 422
SR-TE Reoptimization Timers 425
Circuit-Style SR-TE Policies 426
Reporting of SR-TE Policies Using BGP- Link State 440
Restrictions to Reporting of SRTE Policies using BGP-LS 441
Configure Reporting of SRTE Policies using BGP-LS 441
SR Policy Path Computation for IPv6 448
Usage Guidelines and Limitations 449

CHAPTER 11 **Segment Routing Tree Segment Identifier** 451

Configure Segment Routing Tree-SID 451

Running Config 453

Multicast VPN: Tree-SID MVPN 455

Multicast: Cisco Nonstop Forwarding for Tree-SID 470

Multicast VPN IPv6: Dynamic Tree-SID Multicast VPN IPv6 471

Prerequisites for Tree-SID mVPN IPv6 478

Restrictions to Tree-SID mVPN IPv6 479

Configure Tree-SID mVPN IPv6 479

Verify Tree-SID mVPN IPv6 With TI-LFA 481

CHAPTER 12 **Enabling Segment Routing Flexible Algorithm** 489

Prerequisites for Flexible Algorithm 490

Building Blocks of Segment Routing Flexible Algorithm 490

Flexible Algorithm Definition 490

Flexible Algorithm Support Advertisement 490

Flexible Algorithm Definition Advertisement 491

Flexible Algorithm Prefix-SID Advertisement 491

Calculation of Flexible Algorithm Path 491

Installation of Forwarding Entries for Flexible Algorithm Paths 491

Flexible Algorithm Prefix-SID Redistribution 492

Configuring Flexible Algorithm 494

Configuring Flexible Algorithm with Exclude SRLG Constraint 496

Flexible Algorithm with Exclude Minimum Bandwidth Constraint 500

Flexible Algorithm with Exclude Maximum Delay Constraint 502

Maximum Paths Per IS-IS Flexible Algorithm Per Prefix 504

Example: Configuring IS-IS Flexible Algorithm 506

Example: Configuring OSPF Flexible Algorithm 506

User-Defined Generic Metric Support for IS-IS Flex Algo 508

Usage Guidelines and Limitations for User-Defined Generic Metrics 510

Configure User-Defined Generic Metrics 510

CHAPTER 13 **Configure Segment Routing Path Computation Element** 513

About SR-PCE 515
Usage Guidelines and Limitations 516
Configure SR-PCE 516
Disjoint Policy 518
Configure Disjoint Policy 519
PCE-initiated SR Policies for Traffic Management 519
Configure PCE-initiated SR Policy 520
Configure PCE-initiated SR Policy with Explicit SID List 520
Configure PCE-initiated SR Policy with Dynamic SID List 521
Exclude Network Resources during Path Computation over SR-TE Policies 523
Configuration to exclude network resources during Path Computation 525
Enable Strict Disjointness for SR-TE policies in the PCE 528
Configure Strict Disjointness in the PCE 529
Configure the Shortest Path for Disjoint Candidate Paths 530
Configure the shortest path for disjoint candidate paths 531
PCC-initiated Policies Delegated to PCE 532
Configure PCC-initiated Policies Delegated to PCE 532
SR-PCE IPv4 Unnumbered Interface Support 533
Configure SR-PCE IPv4 Unnumbered Interface 534
Inter-Domain Path Computation Using Redistributed SID 535
Example: Inter-Domain Path Computation Using Redistributed SID 537

CHAPTER 14 **Configure Performance Measurement** 539
Liveness Monitoring 540
IP Endpoint Liveness Monitoring 541
IP Endpoint Liveness Detection in an SR MPLS Network 545
IP Endpoint Liveness in an SRv6 Network 547
SR Policy Liveness Monitoring 550
Delay Measurement 563
Measurement Modes 563
Link Delay Measurement 567
Delay Normalization 577
Link Anomaly Detection with IGP Penalty 580
Delay Measurement for IP Endpoint 581

IP Endpoint Liveness Detection in an SR MPLS Network 583
IP Endpoint Delay Measurement over SRv6 Network Usecase 585
SR Policy End-to-End Delay Measurement 587
Path Tracing in SRv6 Network 591
Limitations and Guidelines 593
Configuration Steps 593
Two-Way Active Measurement Protocol Light Source Address Filtering 601
Usage Guidelines and Limitations 602
Configure IP address on querier and responder nodes 602
Synthetic Loss Measurement 605
Configure Synthetic Loss Measurement 606

CHAPTER 15 **Configure Topology-Independent Loop-Free Alternate (TI-LFA) 611**
Behaviors and Limitations of TI-LFA 613
Configuring TI-LFA for IS-IS 615
Configuring TI-LFA for OSPF 617
TI-LFA Node and SRLG Protection: Examples 618
Configuring Global Weighted SRLG Protection 619

CHAPTER 16 **Configure Segment Routing Microloop Avoidance 623**
About Segment Routing Microloop Avoidance 623
Configure Segment Routing Microloop Avoidance for IS-IS 625
Microloop Avoidance for IS-IS with Per-Prefix Filtering 627
Configure Segment Routing Microloop Avoidance for OSPF 630

CHAPTER 17 **Configure Segment Routing Mapping Server 633**
Segment Routing Mapping Server 633
Segment Routing Mapping Server Restrictions 634
Segment Routing and LDP Interoperability 634
Example: Segment Routing LDP Interoperability 634
Configuring Mapping Server 636
Enable Mapping Advertisement 638
Configure Mapping Advertisement for IS-IS 638
Configure Mapping Advertisement for OSPF 639

Enable Mapping Client 641

CHAPTER 18 Using Segment Routing OAM 643

MPLS Ping and Traceroute for BGP and IGP Prefix-SID 643

Examples: MPLS Ping, Traceroute, and Tree Trace for Prefix-SID 644

MPLS LSP Ping and Traceroute Nil FEC Target 646

Examples: LSP Ping and Traceroute for Nil_FEC Target 646

Segment Routing Ping and Traceroute 648

Segment Routing Ping 648

Segment Routing Traceroute 650

Segment Routing Policy Nil-FEC Ping and Traceroute 653

Segment Routing Data Plane Monitoring 655

Configure SR DPM 658

Data Plane Validation Support for SR-MPLS IPv6-based LSPs 660

Examples: SR-MPLS Data Plane Validation over IPv6-based LSPs 661

MPLS OAM support for SR-TE Policies using IPv6-based LSPs 662

MPLS OAM support for SR-TE Policies using MPLS IPv6-based LSPs 662

Examples: MPLS OAM support for SR-TE Policies with IPv6-based LSPs 663

Preface

The *Segment Routing Configuration Guide for Cisco 8000 Series Router* preface contains these sections:

- Communications, Services, and Additional Information, on page xiii
- Changes to This Document, on page xiii

Communications, Services, and Additional Information

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

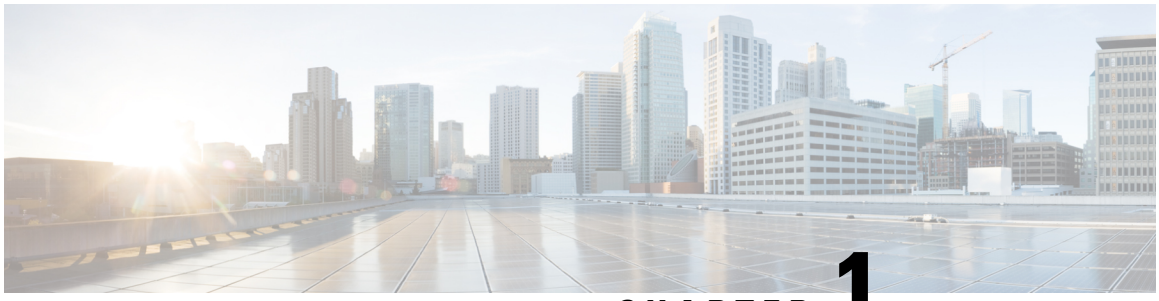
Cisco Bug Search Tool

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

Changes to This Document

None

Date	Change Summary
July 2024	Republished for Release 24.2.11.
March 2024	Initial release of this document.



CHAPTER 1

New and Changed Information for Segment Routing Features

This table summarizes the new and changed feature information for the *Segment Routing Configuration Guide for Cisco 8000 Series Routers*, and lists where they are documented.

• New and Changed Segment Routing Features-24.x.x, on page 1

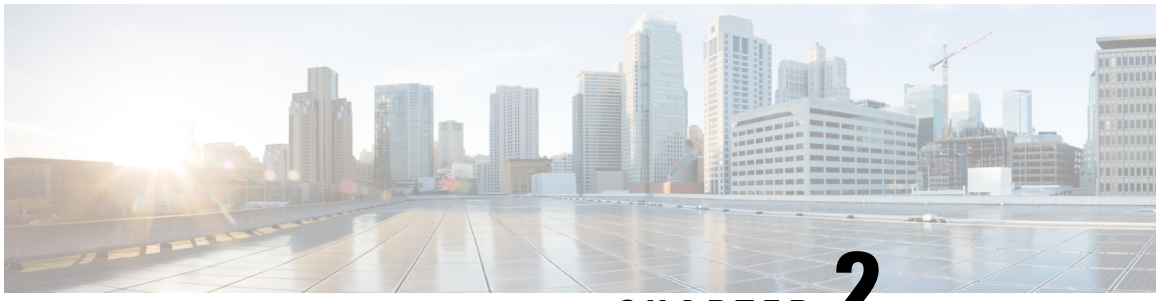
New and Changed Segment Routing Features-24.x.x

Segment Routing Features Added or Modified in IOS XR Release 24.x.x

Feature	Description	Changed in Release	Where Documented
BGP Signaling for co-existence of IP routes	This feature is introduced.	Release 24.3.1	BGP Signaling for co-existence of IP routes with or without SRv6 SID, on page 75
H.Insert.Red Headend Behavior for SRv6 on Cisco Silicon One P100-based Routers	This feature is introduced.	Release 24.3.1	SRv6 Head-End Behaviors, on page 11
SRv6 Services on EVPN E-Line	This feature is introduced.	Release 24.3.1	SRv6 Services on EVPN E-Line, on page 183
Data Plane Validation for SR-MPLS IPv6-based Controller Instantiated LSPs	This feature is introduced.	Release 24.2.11	Data Plane Validation Support for SR-MPLS IPv6-based LSPs, on page 660
Overriding MPLS Imposition (IP-to-MPLS) via Service Layer API (SL-API)	This feature is introduced.	Release 24.2.11	Overriding MPLS Imposition (IP-to-MPLS) via Service Layer API (SL-API), on page 255

Feature	Description	Changed in Release	Where Documented
MPLS OAM support for SR-TE Policies using MPLS IPv6-based LSPs	This feature is introduced.	Release 24.2.11	MPLS OAM support for SR-TE Policies using IPv6-based LSPs, on page 662
User-Defined Generic Metric Support for IS-IS Flex Algo	This feature is introduced.	Release 24.2.11	User-Defined Generic Metric Support for IS-IS Flex Algo, on page 508
Liveness Monitoring for IP Endpoint over SRv6 Network	This feature is introduced.	Release 24.2.11	IP Endpoint Liveness Monitoring, on page 541
Delay Measurement for IP Endpoint over SRv6 Network	This feature is introduced.	Release 24.2.11	Delay Measurement for IP Endpoint, on page 581
Reporting of SR-TE Policies Using BGP-Link State for SRv6	This feature is introduced.	Release 24.1.1	Reporting of SR-TE Policies Using BGP- Link State , on page 230
Path Maximum Transmission Unit (MTU) discovery for SRv6 Encapsulated Packets	This feature is introduced.	Release 24.1.1	Path Maximum Transmission Unit (MTU) Discovery for SRv6 Encapsulated Packets, on page 135
Synthetic Loss Measurement	This feature is introduced.	Release 24.1.1	Synthetic Loss Measurement, on page 605
Path Tracing Source and Sink Nodes	This feature is introduced.	Release 24.1.1	Path Tracing in SRv6 Network, on page 591
SR-MPLSv6 Traffic Engineering	This feature is introduced.	Release 24.1.1	SR Policy Path Computation for IPv6, on page 448
Exclude Network Resources during Path Computation over SR-TE Policies	This feature is introduced.	Release 24.1.1	Exclude Network Resources during Path Computation over SR-TE Policies, on page 523
Configure the Shortest Path for Disjoint Candidate Paths	This feature is introduced.	Release 24.1.1	Configure the Shortest Path for Disjoint Candidate Paths, on page 530
Enable Strict Disjointness for SR-TE policies in the PCE	This feature is introduced.	Release 24.1.1	Enable Strict Disjointness for SR-TE policies in the PCE, on page 528

Feature	Description	Changed in Release	Where Documented
Identical Route Distinguisher (RD) for Interworking Gateways between MPLS and SRv6 Domains	This feature was modified.	Release 24.1.1	SRv6/MPLS L3 Service Interworking Gateway, on page 96



CHAPTER 2

YANG Data Models for Segment Routing Features

This chapter provides information about the YANG data models for Segment Routing features.

- Using YANG Data Models, on page 5

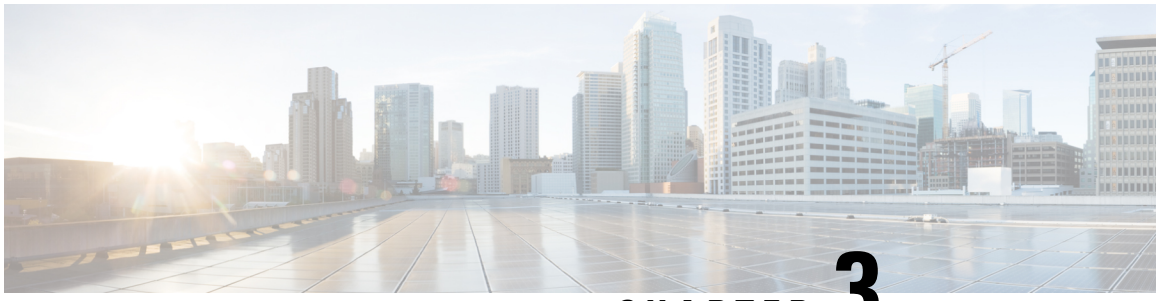
Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the **Available-Content.md** file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.



CHAPTER 3

Configure Segment Routing over IPv6 (SRv6) with Micro-SIDs

Table 1: Feature History Table

Feature Name	Release	Description
SRv6 with Micro-Segment (uSID)	Release 7.5.2	This release introduces support for Segment Routing over IPv6 data plane using Micro SIDs (uSIDs). This feature allows the source router to encode multiple SRv6 uSID instructions within a single 128-bit SID address. Such functionality allows for efficient and compact SRv6 SID representation with a low MTU overhead

Segment Routing for IPv6 (SRv6) is the implementation of Segment Routing over the IPv6 dataplane.

In a Segment Routing over IPv6 (SRv6) network, an IPv6 address serves as the segment identifier (SID). The source router can encode multiple SRv6 uSID instructions within a single 128-bit SID address. Such a SID address is called a uSID Carrier.

SRv6 uSIDs provides low MTU overhead; for example, 6 uSIDs per uSID carrier results in 18 source-routing waypoints in only 40 bytes of overhead (in SRH).

- Segment Routing over IPv6 Overview, on page 8
- SRv6 Micro-Segment (uSID), on page 10
- Usage Guidelines and Limitations, on page 25
- Configuring SRv6, on page 27
- Configuring SRv6 under IS-IS, on page 32
- Configuring SRv6 Flexible Algorithm under IS-IS, on page 32
- Configuring SRv6 Locator Prefix Summarization, on page 34
- Configuring TI-LFA with SRv6 IS-IS, on page 35
- Configuring SRv6 IS-IS Microloop Avoidance, on page 37
- Configuring SRv6 BGP-Based Services, on page 38
- SRv6/MPLS L3 Service Interworking Gateway, on page 96
- L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway, on page 102
- L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway, on page 105
- SRv6/MPLS Dual-Connected PE, on page 108

- SRv6 Provider Edge (PE) Lite Support, on page 110
- SRv6 SID Information in BGP-LS Reporting, on page 123
- Full-Replace Migration to SRv6 Micro-SID, on page 124
- SRv6 Traffic Accounting, on page 128
- Path Maximum Transmission Unit (MTU) Discovery for SRv6 Encapsulated Packets, on page 135

Segment Routing over IPv6 Overview

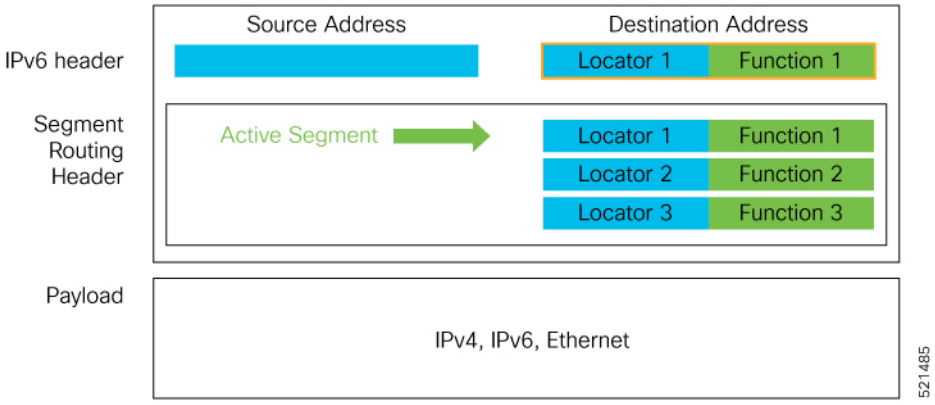
Segment Routing (SR) can be applied on both MPLS and IPv6 data planes. Segment Routing over IPv6 (SRv6) extends Segment Routing support with IPv6 data plane.

In an SR-MPLS enabled network, an MPLS label represents an instruction. The source nodes programs the path to a destination in the packet header as a stack of labels.

SRv6 introduces the Network Programming framework that enables a network operator or an application to specify a packet processing program by encoding a sequence of instructions in the IPv6 packet header. Each instruction is implemented on one or several nodes in the network and identified by an SRv6 Segment Identifier (SID) in the packet. The SRv6 Network Programming framework is defined in IETF RFC 8986 SRv6 Network Programming.

In SRv6, an IPv6 address represents an instruction. SRv6 uses a new type of IPv6 Routing Extension Header, called the Segment Routing Header (SRH), in order to encode an ordered list of instructions. The active segment is indicated by the destination address of the packet, and the next segment is indicated by a pointer in the SRH.

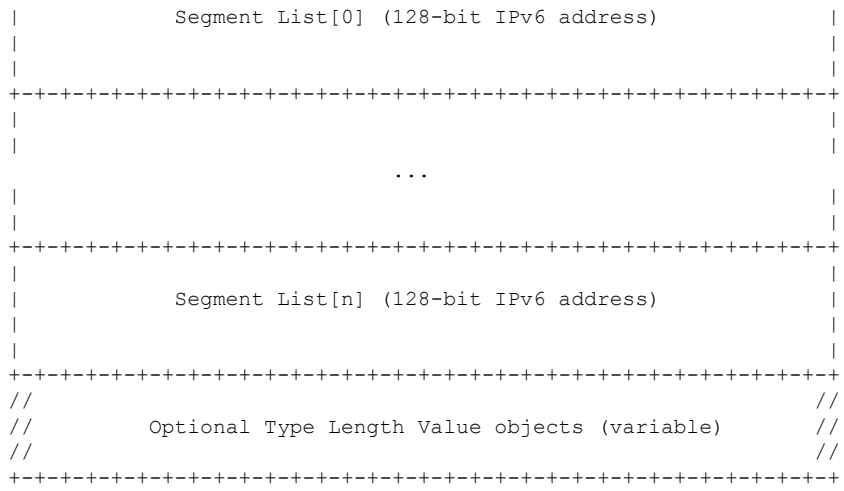
Figure 1: Network Program in the Packet Header



The SRv6 SRH is documented in IETF RFC IPv6 Segment Routing Header (SRH).

The SRH is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Last Entry   |     Flags     |              Tag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
```



The following list explains the fields in SRH:

- Next header—Identifies the type of header immediately following the SRH.
- Hdr Ext Len (header extension length)—The length of the SRH in 8-octet units, not including the first 8 octets.
- Segments left—Specifies the number of route segments remaining. That means, the number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
- Last Entry—Contains the index (zero based) of the last element of the segment list.
- Flags— Contains 8 bits of flags.
- Tag—Tag a packet as part of a class or group of packets like packets sharing the same set of properties.
- Segment list—128-bit IPv6 addresses representing the *n*th segment in the segment list. The segment list encoding starts from the last segment of the SR policy (path). That means the first element of the segment list (Segment list [0]) contains the last segment of the SR policy, the second element contains the penultimate segment of the SR policy and so on.

In SRv6, a SID represents a 128-bit value, consisting of the following three parts:

- Locator: This is the first part of the SID with most significant bits and represents an address of a specific SRv6 node.
- Function: This is the portion of the SID that is local to the owner node and designates a specific SRv6 function (network instruction) that is executed locally on a particular node, specified by the locator bits.
- Args: This field is optional and represents optional arguments to the function.

The locator part can be further divided into two parts:

- SID Block: This field is the SRv6 network designator and is a fixed or known address space for an SRv6 domain. This is the most significant bit (MSB) portion of a locator subnet.
- Node Id: This field is the node designator in an SRv6 network and is the least significant bit (LSB) portion of a locator subnet.

SRv6 Node Roles

Each node along the SRv6 packet path has a different functionality:

- Source node—A node that can generate an IPv6 packet with an SRH (an SRv6 packet), or an ingress node that can impose an SRH on an IPv6 packet.
- Transit node—A node along the path of the SRv6 packet (IPv6 packet and SRH). The transit node does not inspect the SRH. The destination address of the IPv6 packet does not correspond to the transit node.
- Endpoint node—A node in the SRv6 domain where the SRv6 segment is terminated. The destination address of the IPv6 packet with an SRH corresponds to the end point node. The segment endpoint node executes the function bound to the SID

SRv6 Micro-Segment (uSID)

The SRv6 micro-segment (uSID) is an extension of the SRv6 architecture. It leverages the SRv6 Network Programming architecture to encode several SRv6 Micro-SID (uSID) instructions within a single 128-bit SID address. Such a SID address is called a uSID Carrier.

SRv6 uSID is documented in the IETF drafts Network Programming extension: SRv6 uSID instruction and Compressed SRv6 Segment List Encoding in SRH.

Throughout this chapter, we will refer to SRv6 micro-segment as “uSID”.

The SRv6 uSID provides the following benefits:

- Leverages the SRv6 Network Programming with no change. SRv6 uSID is a new pseudo code in the existing SRv6 network programming framework.
- Leverages the SRv6 data plane (SRH) with no change. Any SID in the destination address or SRH can be an SRv6 uSID carrier.
- Leverages the SRv6 control plane with no change.
- Ultra-Scale—Scalable number of globally unique nodes in the domain, for example:
  - 16-bit uSID ID size: 65k uSIDs per domain block
  - 32-bit uSID ID size: 4.3M uSIDs per domain block
- Lowest MTU overhead
  - 6 uSIDs per uSID carrier
  - For example, 18 source-routing waypoints in only 40 bytes of overhead
    - + H.Encaps.Red with an SRH of 40 bytes (8 fixed + 2 * 16 bytes)
    - + 6 uSIDs in DA and 12 in SRH
- Hardware-friendliness:
  - Leverages mature hardware capabilities (inline IP Destination Address edit, IP Destination Address longest match).
  - Avoids any extra lookup in indexed mapping tables.

- A micro-program with 6 or fewer uSIDs requires only legacy IP-in-IP encapsulation behavior.

- Scalable Control Plane:
  - Summarization at area/domain boundary provides massive scaling advantage.
  - No routing extension is required, a simple prefix advertisement suffices.
- Seamless Deployment:
  - A uSID may be used as a SID (the carrier holds a single uSID).
  - The inner structure of an SR Policy can stay opaque to the source. A carrier with uSIDs is just seen as a SID by the policy headend Security.
  - Leverages SRv6's native SR domain security.

SRv6 Head-End Behaviors

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
H.Insert.Red Headend Behavior for SRv6 on Cisco Silicon One P100-based Line Cards	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) With H.Insert.Red head-end behavior, you can effectively steer traffic into an SR policy, allowing for fast rerouting, traffic optimization, and simplified path management without additional encapsulation. The H.Insert.Red head-end behavior enables the router to insert a Segment Routing Header (SRH) directly into an existing IPv6 packet. The feature is supported only on Cisco Silicon One P100-based line cards in Cisco 8800 modular routers operating in P100 compatibility mode. * This feature is supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

SR policies define the behavior of headend routers in managing and directing traffic through a network. The headend router is responsible for initiating and enforcing these policies. SR supports these headend behaviors.

- H.Encaps.Red—H.Encaps with Reduced Encapsulation
- H.Insert.Red—H.Insert with reduced insertion

The SR Headend with Encapsulation behaviors are documented in the IETF RFC 8986 SRv6 Network Programming.

The SR Headend with Insertion head-end behaviors are documented in the following IETF draft:

https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-net-pgm-insertion/

Default Headend Behaviour

Starting from Cisco IOS XR Release 24.3.1, the H.Insert.Red headend behavior is supported only on routers with P100 based line cards. Based on the available line cards, the default headend behavior varies. The table summarizes the default behavior.

If the router has ….	Then the default headend behavior is...
Cisco Silicon One P100-based line cards	H.Insert.Red
Cisco Silicon One Q200-based line cards	H.Encap.Red
a mix of Cisco Silicon One P100- and Q200-based line cards	H.Encap.Red When using a combination of Q200 and P100 line cards, enable the **hw-module profile npu-compatibility** command to switch the NPU operating mode to Q200. You need to manually reload the router to activate the NPU compatibility mode.

Comparision between H.Encaps.Red and H.Insert.Red Headend Behaviors

This table describes the difference between the H.Encaps and H.Insert.Red head-end behaviors.

Headend Behaviors	H.Encaps.Red	H.Insert.Red
Definition	The H.Encap.Red is a headend behavior that **encapsulates** the original packet into a new IPv6 packet with an Segment Routing Header (SRH).	The H.insert.Red is a headend behavior that **inserts** an SRH into the original IPv6 packet without encapsulating it into a new IPv6 packet.
Header Manipulation	A new IPv6 header with an SRH is added to the packet, encapsulating the original packet.	The SRH is inserted into the packet by modifying the existing IPv6 header.
Packet Size	The packet size increases as it includes both the SRH and an additional IPv6 header.	The packet size is smaller compared to H.encaps headend behavior. There is no extra IPv6 header and that helps maintain the packet size.
Processing at Intermediate Nodes	Intermediate nodes process the packet by examining the outer IPv6 header's destination address and the SRH.	Intermediate nodes process the packet by examining the inserted SRH and forwarding the packet based on the active segment.

Headend Behaviors	H.Encaps.Red	H.Insert.Red
Termination Process	Ultimate Segment Pop The termination process involves decapsulation, where the outer IPv6 header and SRH are removed to reveal the original packet.	Penultimate Segment Pop (PSP) The termination process involves the removal of the SRH when the packet reaches the end of the SR Policy.

SRv6 Endpoint Behaviors

SRv6 Endpoint Behaviors

The SRv6 endpoint behaviors are documented in the IETF RFC 8986 SRv6 Network Programming.

The following is a subset of defined SRv6 endpoint behaviors that can be associated with a SID.

- End—Endpoint function. The SRv6 instantiation of a Prefix SID [RFC8402].
- End.X—Endpoint with Layer-3 cross-connect. The SRv6 instantiation of an Adj SID [RFC8402].
- End.DX6—Endpoint with decapsulation and IPv6 cross-connect (IPv6-L3VPN - equivalent to per-CE VPN label).
- End.DX4—Endpoint with decapsulation and IPv4 cross-connect (IPv4-L3VPN - equivalent to per-CE VPN label).
- End.DT6—Endpoint with decapsulation and IPv6 table lookup (IPv6-L3VPN - equivalent to per-VRF VPN label).
- End.DT4—Endpoint with decapsulation and IPv4 table lookup (IPv4-L3VPN - equivalent to per-VRF VPN label).
- End.DT46—Endpoint with decapsulation and specific IP table lookup (IP-L3VPN - equivalent to per-VRF VPN label).
- End.DX2—Endpoint with decapsulation and L2 cross-connect (L2VPN use-case).
- End.B6.Encaps—Endpoint bound to an SRv6 policy with encapsulation. SRv6 instantiation of a Binding SID.
- End.B6.Encaps.RED—End.B6.Encaps with reduced SRH. SRv6 instantiation of a Binding SID.

SRv6 Endpoint Behavior Variants

Depending on how the SRH is handled, different behavior variants are defined for the End and End.X behaviors. The End and End.X behaviors can support these variants, either individually or in combinations.

- **Penultimate Segment Pop (PSP) of the SRH variant**—An SR Segment Endpoint Nodes receive the IPv6 packet with the Destination Address field of the IPv6 Header equal to its SID address.

  A penultimate SR Segment Endpoint Node is one that, as part of the SID processing, copies the last SID from the SRH into the IPv6 Destination Address and decrements the Segments Left value from one to zero.

The PSP operation takes place only at a penultimate SR Segment Endpoint Node and does not happen at non-penultimate endpoint nodes. When a SID of PSP-flavor is processed at a non-penultimate SR Segment Endpoint Node, the PSP behavior is not performed since Segments Left would not be zero.

The SR Segment Endpoint Nodes advertise the SIDs instantiated on them via control plane protocols. A PSP-flavored SID is used by the Source SR Node when it needs to instruct the penultimate SR Segment Endpoint Node listed in the SRH to remove the SRH from the IPv6 header.

- **Ultimate Segment Pop (USP) of the SRH variant**—The SRH processing of the End and End.X behaviors are modified as follows:

  If Segments Left is 0, then:

  1. Update the Next Header field in the preceding header to the Next Header value of the SRH
  2. Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)
  3. Remove the SRH from the IPv6 extension header chain
  4. Proceed to process the next header in the packet

  One of the applications of the USP flavor is when a packet with an SRH is destined to an application on hosts with smartNICs implementing SRv6. The USP flavor is used to remove the consumed SRH from the extension header chain before sending the packet to the host.

- **Ultimate Segment Decapsulation (USD) variant**—The Upper-layer header processing of the End and End.X behaviors are modified as follows:

  - **End** behavior: If the Upper-layer Header type is 41 (IPv6), then:

    1. Remove the outer IPv6 Header with all its extension headers
    2. Submit the packet to the egress IPv6 FIB lookup and transmission to the new destination
    3. Else, if the Upper-layer Header type is 4 (IPv4)
    4. Remove the outer IPv6 Header with all its extension headers
    5. Submit the packet to the egress IPv4 FIB lookup and transmission to the new destination
    6. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

  - **End.X** behavior: If the Upper-layer Header type is 41 (IPv6) or 4 (IPv4), then:

    1. Remove the outer IPv6 Header with all its extension headers
    2. Forward the exposed IP packet to the L3 adjacency J
    3. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

  One of the applications of the USD flavor is the case of TI-LFA in P routers with encapsulation with H.Encaps. The USD flavor allows the last Segment Endpoint Node in the repair path list to decapsulate the IPv6 header added at the TI-LFA Point of Local Repair and forward the inner packet.

SRv6 uSID Terminology

The SRv6 Network Programming is extended with the following terms:

- uSID—An identifier that specifies a micro-segment.

  A uSID has an associated behavior that is the SRv6 function (for example, a node SID or Adjacency SID) associated with the given ID. The node at which an uSID is instantiated is called the "Parent" node.

- uSID Carrier—A 128-bit IPv6 address (carried in either in the packet destination address or in the SRH) in the following format:

  `<uSID-Block><Active-uSID><Next-uSID>...<Last-uSID><End-of-Carrier>...<End-of-Carrier>`

  where:

  - uSID Block—An IPv6 prefix that defines a block of SRv6 uSIDs.
  - Active uSID—The first uSID that follows the uSID block.
  - Next uSID—The next uSID after the Active uSID.
  - Last uSID—The last uSID in the carrier before the End-of-Carrier uSID.
  - End-of-Carrier —A globally reserved uSID that marks the end of a uSID carrier. The End-of-Carrier ID is **0000**. All empty uSID carrier positions must be filled with the End-of-Carrier ID; therefore, a uSID carrier can have more than one End-of-Carrier.

  The following is an example of an SRH with 3 Micro-SID carriers for a total of up to 18 micro-instructions:

Micro-SID Carrier1: {uInstruction1, uInstruction2… uInstruction6}
Micro-SID Carrier2: {uInstruction7, uInstruction8… uInstruction12}
Micro-SID Carrier3: {uInstruction13, uInstruction14… uInstruction18}

SRv6 uSID Carrier Format

The uSID carrier format specifies the type of uSID carrier supported in an SRv6 network. The format specification includes Block size and ID size.

- **uSID Block**

  The uSID block is an IPv6 prefix that defines a block of SRv6 uSIDs. This can be an IPv6 prefix allocated to the provider (for example, /22, /24, and so on.), or it can be any well-known IPv6 address block generally available for private use, such as the ULA space FC/8, as defined in IETF draft RFC4193.

  An SRv6 network may support more than a single uSID block.

  The length of block [prefix] is defined in bits. From a hardware-friendliness perspective, it is expected to use sizes on byte boundaries (16, 24, 32, and so on).

- **uSID ID**

  The length of uSID ID is defined in bits. From a hardware-friendliness perspective, it is expected to use sizes on byte boundaries (8, 16, 24, 32, and so on).

The uSID carrier format is specified using the notation "F*bbuu*" , where "*bb*" is size of block and "*uu*" is size of ID. For example, "F3216" is a format with a 32-bit uSID block and 16-bit uSID IDs.

SRv6 uSID Allocation Within a uSID Block

Table 3: Feature History Table

Feature Name	Release	Description
Wide LIB uSID Allocation for End.DT46 SRv6 SIDs	Release 7.5.3	This feature introduces support for Wide Local ID block (W-LIB). W-LIB provides an extended set of IDs available for local uSID allocation that can be used when a PE with large-scale Pseudowire termination requires more local uSIDs than provided from the LIB. W-LIB uSID allocation is supported for End.DT46 SRv6 SIDs.

Key Concepts and Terminologies

The architecture for uSID specifies both globally scoped and locally scoped uSIDs.

- Global ID block (GIB): The set of IDs available for globally scoped uSID allocation.

  A globally scoped uSID is the type of uSID that provides reachability to a node. A globally scoped uSID typically identifies a shortest path to a node in the SR domain. An IP route (for example, /48) is advertised by the parent node to each of its globally scoped uSIDs, under the associated uSID block. The parent node executes a variant of the END behavior.

  The "nodal" uSID (uN) is an example of a globally scoped behavior defined in uSID architecture.

  A node can have multiple globally scoped uSIDs under the same uSID blocks (for example, one uSID per IGP flex-algorithm). Multiple nodes may share the same globally scoped uSID (Anycast).

- Local ID block (LIB): The set of IDs available for locally scoped uSID allocation.

  A locally scoped uSID is associated to a local (end-point) behavior, and therefore *must* be preceded by a globally scoped uSID of the parent node when relying on routing to forward the packet.

  A locally scoped uSID identifies a local micro-instruction on the parent node; for example, it may identify a cross-connect to a direct neighbor over a specific interface or a VPN context. Locally scoped uSIDs are not routeable.

  For example, if N1 and N2 are two different physical nodes of the uSID domain and *L* is a locally scoped uSID value, then N1 and N2 may bind two different behaviors to *L*.

- Wide LIB (W-LIB): The extended set of IDs available for local uSID allocation.

  The extended set of IDs is useful when a PE with large-scale Pseudowire termination requires more local uSIDs than provided from the LIB.

Example: uSID Allocation

The request to allocate locally scoped uSIDs comes from SRv6 clients (such as IS-IS or BGP). The request can be to allocate any available ID (dynamic allocation) or to allocate a specific ID (explicit allocation).

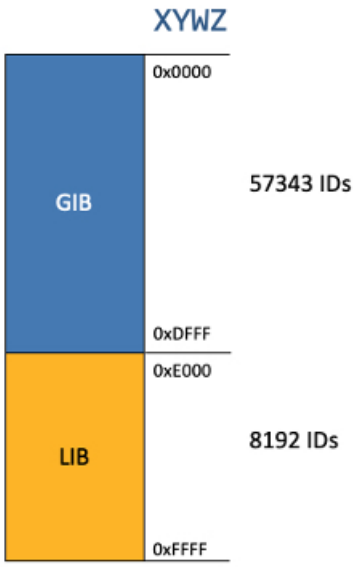
Consider the following example:

- uSID Locator Block length: 32 bits
- uSID Locator Block: FCBB:BB00::/32 (with B being a nibble value picked by operator)
- uSID length (Locator Node ID / Function ID): 16 bits
- uSID: FCBB:BB00:XYWZ::/48 (with XYWZ being variable nibbles)

A uSID FCBB:BB00:XYWZ::/48 is said to be allocated from its block (FCBB:BB00::/32).

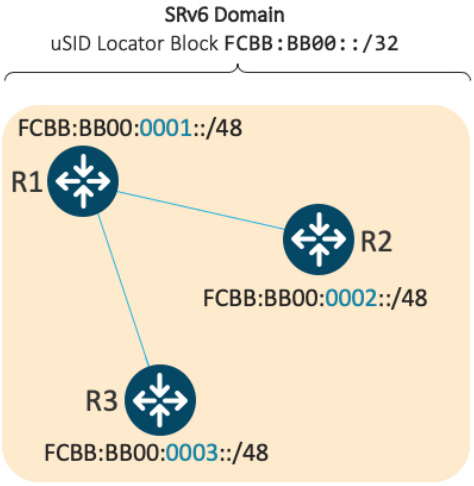
A uSID is allocated from the GIB or LIB of block FCBB:BB00::/32 depending on the value of the "**X**" nibble:

- GIB: nibble **X** from hex(0) to hex(D)
- LIB: nibble **X** hex(E) or hex(F)



With this allocation scheme, the uSID Block **FCBB:BB00::/32** supports up to 57343 global uSIDs (routers) with each router supporting up to 8192 local uSIDs.

For example, the following picture depicts the global uSIDs allocated for 3 nodes within the SRv6 domain.



Looking further into R1, this node also has Local uSIDs associated with uA end-point behaviors:

- Function ID 0xE000 – cross-connect to L3 neighbor R2
- Function ID 0xE001 – cross-connect to L3 neighbor R3

The underlay uSIDs present on R1 are:

- FCBB:BB00:0001::/48
- FCBB:BB00:0001:E000::/64
- FCBB:BB00:0001:E001::/64

GIB and LIB – IOS-XR Implementation

In Cisco IOS XR Release 7.5.2 and earlier, the following functionality is supported:

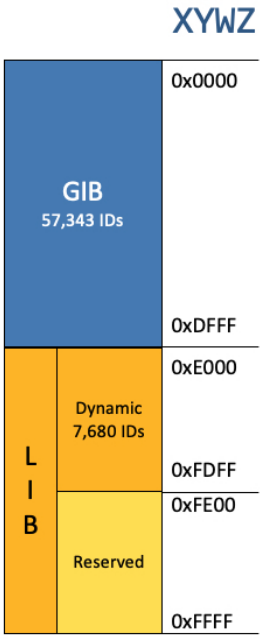
- GIB for user-assigned IDs of global segments (uNs)
- LIB for dynamically assigned IDs of local segments
  - uA end-point behavior
  - Service de-multiplexing end-point behaviors (for example, End.DT, End.DX, End.DX2)

A uSID FCBB:BB00:XYWZ::/48 is said to be allocated from its block FCBB:BB00::/32.

The range of IDs supported by the Cisco IOS XR 7.5.2 and earlier implementation are as follows:

- The range of IDs in the GIB is 0x000 to 0xDFFF.
- The range of IDs by default in the LIB is divided as follows:
  - Dynamic: 0xE000 to 0xFDFF
  - Reserved: 0xFE00 to 0xFFFF

Figure 2: GIB/LIB



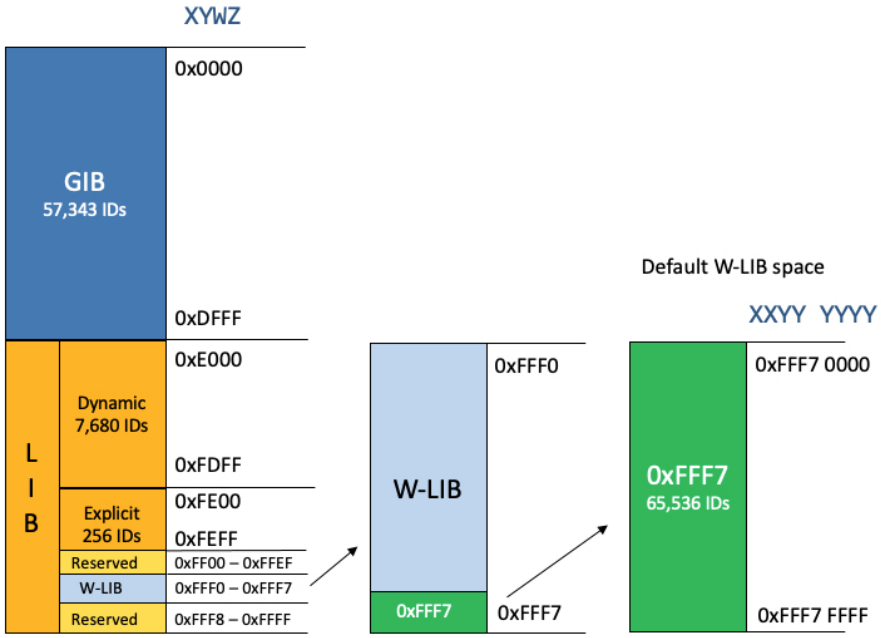
Starting with Cisco IOS XR Release 7.5.3, the following functionality is added:

- Configurable explicit LIB range
- Explicit LIB for user-assigned IDs of local segments
- Manual uDT46 from explicit LIB
- Wide LIB (W-LIB)
- Configurable explicit W-LIB range
- Explicit W-LIB for user-assigned IDs of local segments
- Manual uDT46 from explicit W-LIB

The range of IDs supported by the IOS XR implementation are as follows:

- The range of IDs in the GIB is 0x000 to 0xDFFF.
- The range of IDs by default in the LIB is divided as follows:
    - Dynamic: 0xE000 to 0xFDFF
    - Explicit: 0xFE00 to 0xFEFF
    - Reserved: 0xFF00 to 0xFFEF and 0xFFF8 to 0xFFFF
- The range of IDs by default in the W-LIB is divided as follows:
    - Reserved: 0xFFF0 to 0xFFF6
    - Explicit: 0xFFF7

Figure 3: GIB/LIB/W-LIB



SRv6 Endpoint Behaviors Associated with uSID

The SRv6 Network Programming is extended with new types of SRv6 SID endpoint behaviors:

- uN—A short notation for the NEXT-CSID (Compressed SID) End behavior with a pseudocode of shift-and-lookup, and PSP/USD flavors
- uA—A short notation for the NEXT-CSID End.X behavior with a pseudocode of shift-and-xconnect, and PSP/USD flavors
- uDT—A short notation for the NEXT-CSID End.DT behavior with the same pseudocode as End.DT4/End.DT6/End.DT46/End.DT2U/End.DT2M
- uDX—A short notation for the NEXT-CSID End.DX behavior with the same pseudocode as End.DX4/End.DX6/End.DX2

SRv6 uSID in Action - Example

This example highlights an integrated VPN and Traffic Engineering use-case leveraging SRv6 uSID.

VPNv4 site A connected to Node 1 sends packets to VPNv4 site B connected to Node 2 alongside a traffic engineered path via Node 8 and Node 7 using a single 128-bit SRv6 SID.

Node 1 is the ingress PE; Node 2 is the egress PE.

Nodes 3, 4, 5, and 6 are classic IPv6 nodes. Traffic received on these nodes use classic IP forwarding without changing the outer DA.

Nodes 1, 8, 7 and 2 are SRv6 capable configured with:

- 32-bit SRv6 block = fcbb:bb01

- 16-bit SRv6 ID

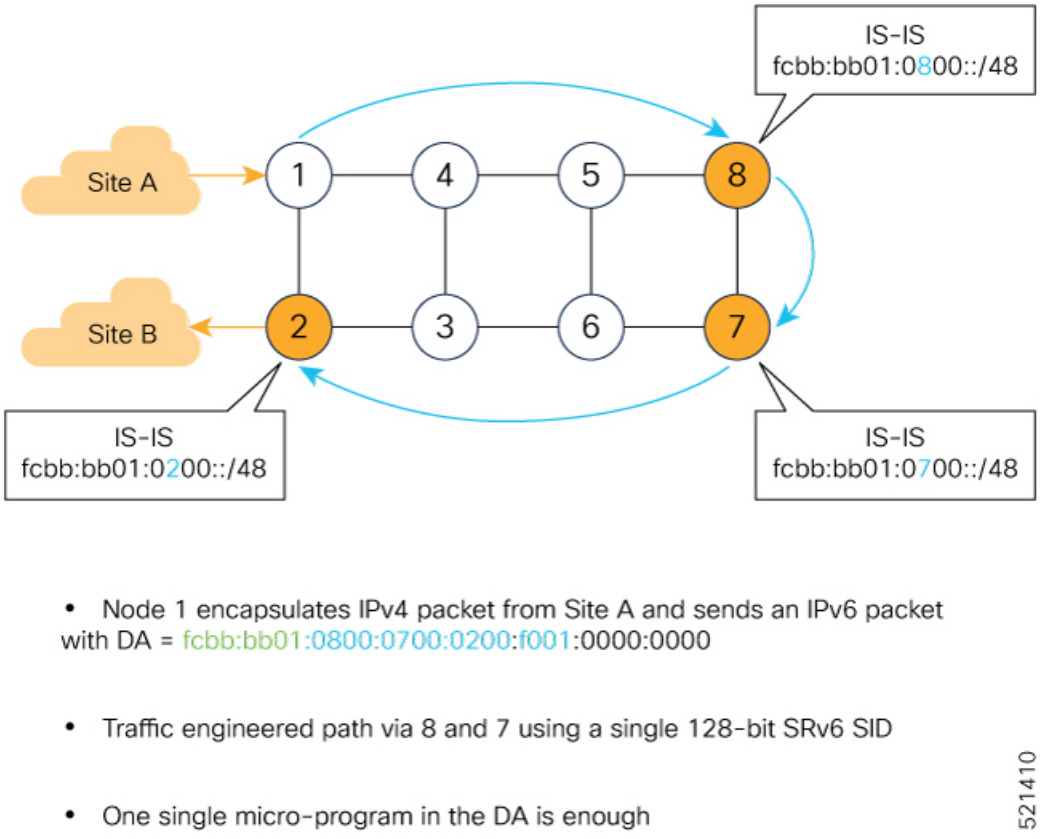
  For example:

  - Node 7 uN = fcbb:bb01:0700::/48
  - Node 8 uN = fcbb:bb01:0800::/48

The following IGP routes are advertised:

- Node 8 advertises the IGP route fcbb:bb01:**0800**::/48
- Node 7 advertises the IGP route fcbb:bb01:**0700**::/48
- Node 2 advertises the IGP route fcbb:bb01:**0200**::/48

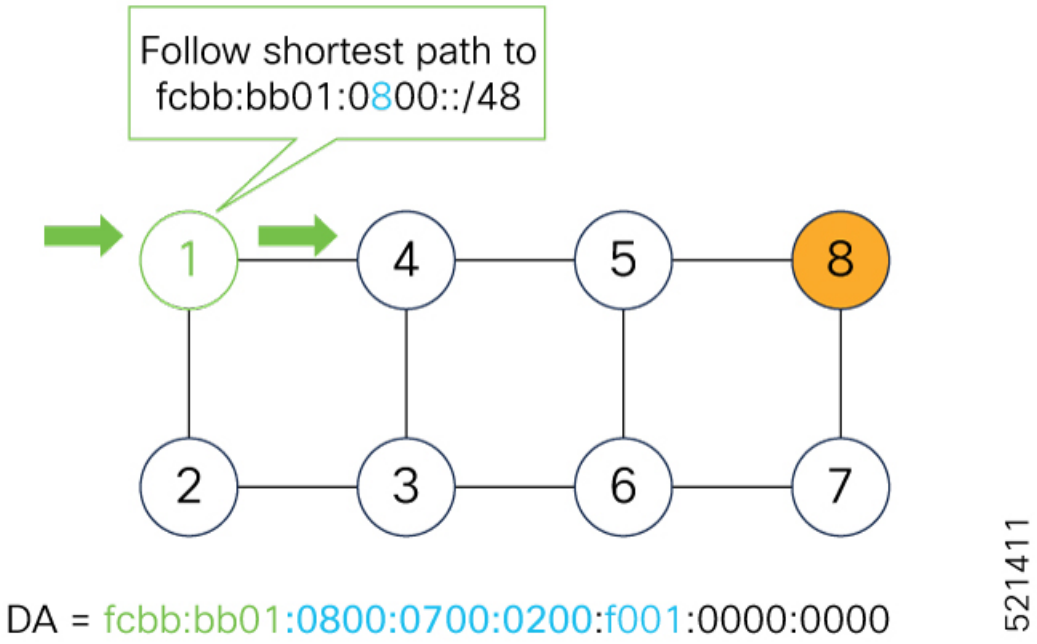
Figure 4: Integrated VPN and Traffic Engineering SRv6 uSID Use-case



Node 1 encapsulates an IPv4 packet from VPN Site A and sends an IPv6 packet with destination address fcbb:bb01:**0800**:**0700**:**0200**:f001:0000:0000. This is a uSID carrier, with a list of micro-instructions (uSIDs) (0800, 0700, 0200, f001, and 0000 – indicating the end of the instruction).

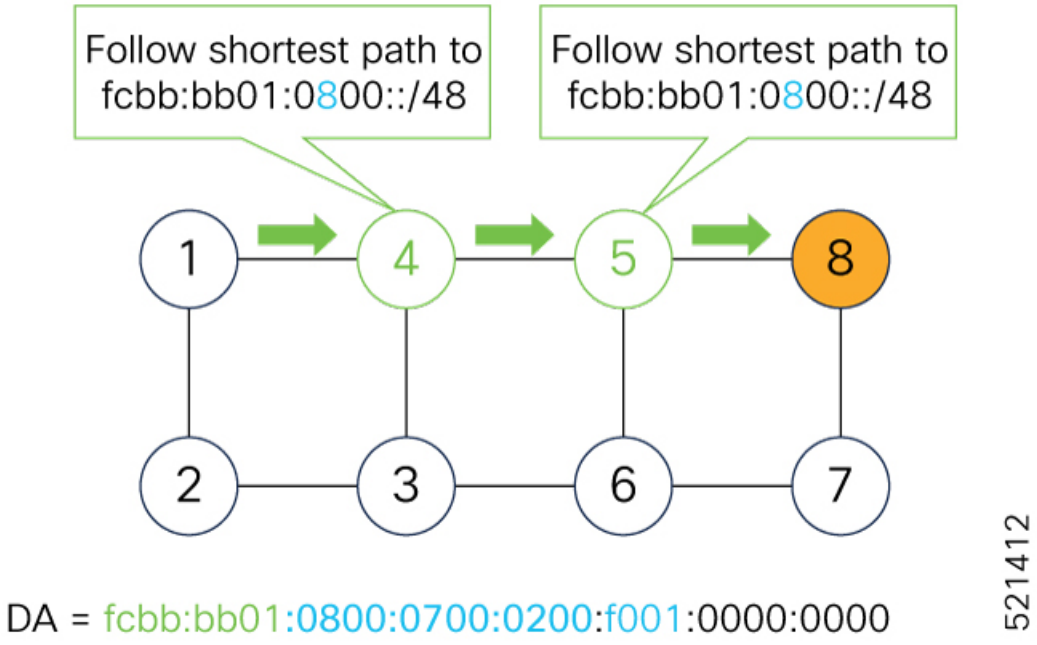
uSIDs (uNs) 0800, 0700, 0200 are used to realize the traffic engineering path to Node 2 with way points at Nodes 8 and 7. uSID f001 is the BGP-signalled instruction (uDT4) advertized by Node 2 for the VPNv4 service

Figure 5: Node 1: End.B6.Encaps Behavior



Nodes 4 and 5 simply forward the packet along the shortest path to Node 8, providing seamless deployment through classic IPv6 nodes.

Figure 6: Node 4 and Node 5: Classic IPv6 Nodes

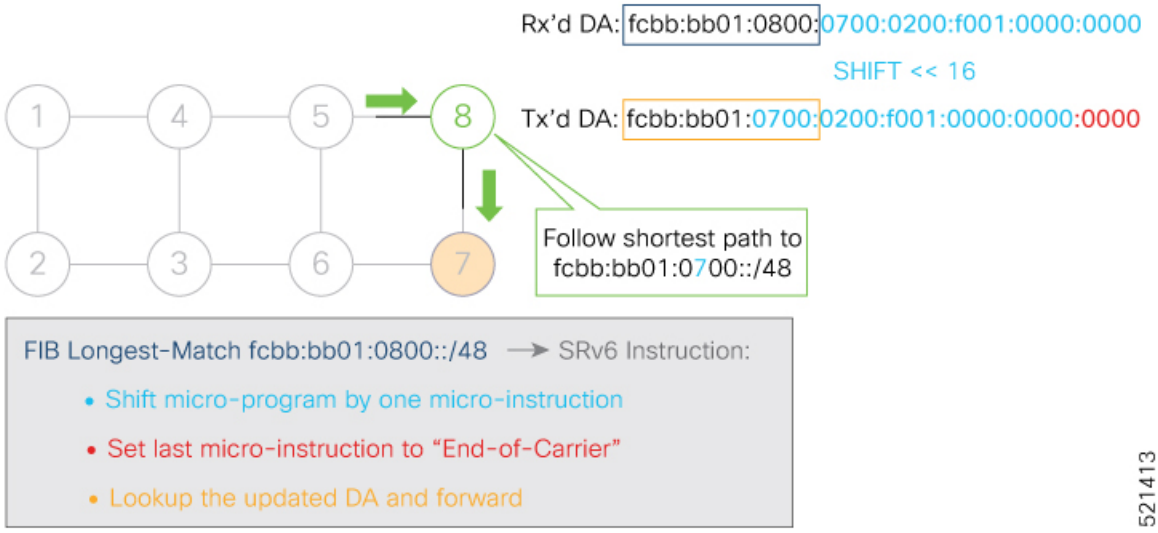


When Node 8 receives the packet, it performs SRv6 uN behavior (shift-and-lookup with PSP/USD). It removes its outer DA (0800) and advances the micro program to the next micro instruction by doing the following:

1. Pops its own uSID (0800)

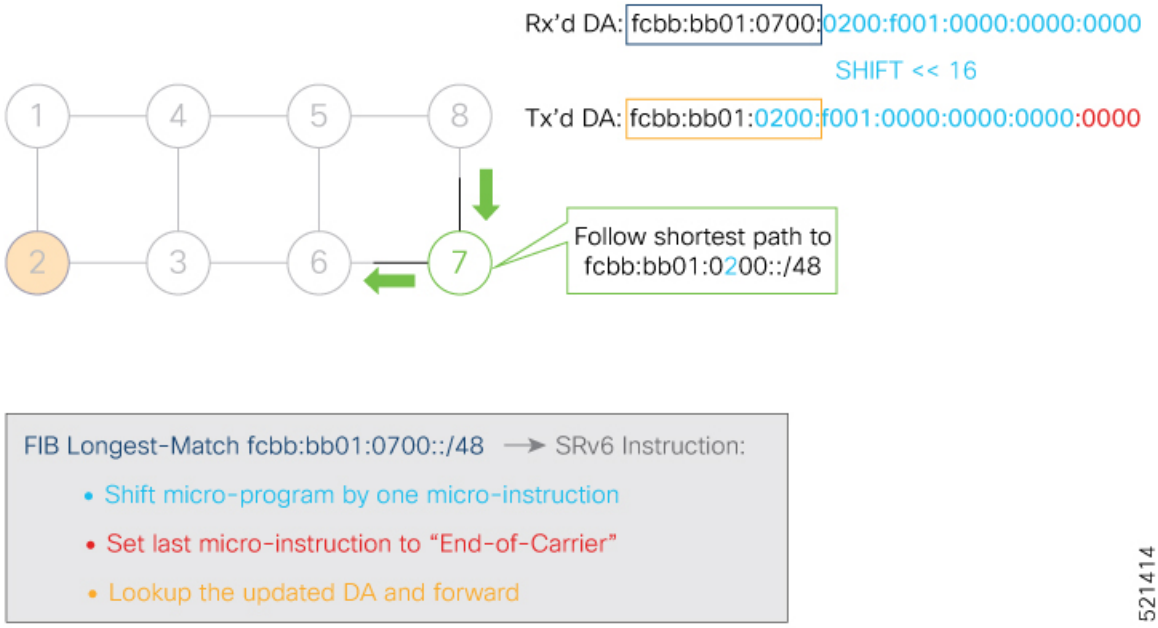
2. **Shifts** the remaining DA by 16-bits to the left
3. Fills the remaining bits with 0000 (End-of-Carrier)
4. Performs a **lookup** for the shortest path to the next DA (fcbb:bb01:**0700::/48**)
5. Forwards it using the new DA fcbb:bb01:**0700**:0200:f001:0000:0000:0000

Figure 7: Node 8: SRv6 uN Behavior (Shift and Forward)



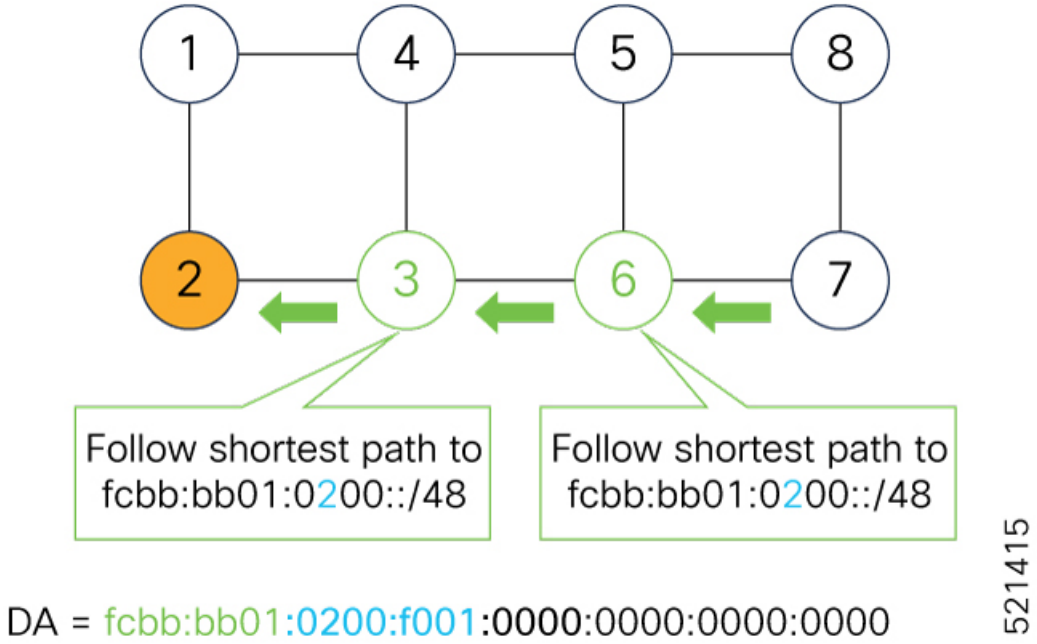
When Node 7 receives the packet, it performs the same SRv6 uN behavior (shift-and-lookup with PSP/USD), forwarding it using the new DA fcbb:bb01:**0200**:f001:0000:0000:0000:0000

Figure 8: Node 7: SRv6 uN Behavior (Shift and Forward)



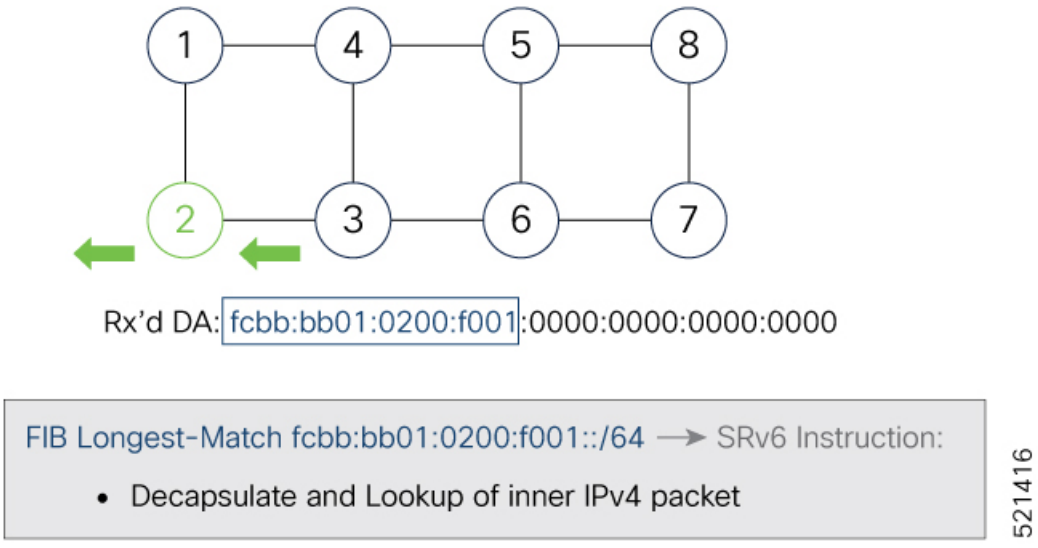
Nodes 6 and 3 simply forward the packet along the shortest path to Node 2, providing seamless deployment through classic IPv6 nodes.

Figure 9: Node 6 and Node 3: Classic IPv6 Nodes



When Node 2 receives the packet, it performs an SRv6 uDT4 behavior (End.DT4—Endpoint with decapsulation and IPv4 table lookup) to VPNv4 Site B.

Figure 10: Node 2: SRv6 uDT4 Behavior



To recap, this example showed an integrated VPN and Traffic Engineering use-case, where VPNv4 site A connected to Node 1 sent packets to VPNv4 site B connected to Node 2 alongside a traffic engineered path via Node 8 and Node 7 using a single 128-bit SRv6 SID:

- @1: inner packet P encapsulated with outer DA fcbb:bb01:**0800**:**0700**:**0200**:f001:0000:0000

- @4 & @5: classic IP forwarding, outer DA unchanged
- @8: SRv6 uN behavior: shift and lookup, outer DA becomes fcbb:bb01:**0700**:**0200**:f001:0000:0000:0000
- @7: SRv6 uN behavior: shift and lookup, outer DA becomes fcbb:bb01:**0200**:f001:0000:0000:0000
- @6 & @3: classic IP forwarding, outer DA unchanged
- @2: SRv6 End.DT4: Decapsulate and IPv4 table lookup

Usage Guidelines and Limitations

General Guidelines and Limitations

- Cisco IOS XR supports uSIDs with 32-bit uSID block and 16-bit uSID IDs (3216).

  A single UCF format must be used for uSID locators in a SRv6 uSID domain.

- Cisco IOS XR supports up to 16 uSID locator prefixes.

  Multiple locator prefixes are used when configuring Anycast locators or SRv6 Flexible Algorithm instances, for example.

- Cisco IOS XR supports uSID locator prefixes from different uSID blocks.

  Up to 256 uSID blocks can be used across all uSID locators in the network.

- SRv6 Underlay support includes:
  - IGP redistribution/leaking between levels
  - Prefix Summarization on ABR routers
  - IS-IS TI-LFA
  - Microloop Avoidance
  - Flex-algo
- SRv6 over GRE interface is not supported
- SRv6 over BVI interface is not supported

uSID Allocation Recommendation

We recommend allocating uSIDs from the private IPv6 space (IPv6 Unique Local Address [ULA] range). These addresses are not routable outside the domain and are therefore secure.

Allocation from the public IPv6 space (Global Unicast Addresses [GUA] range) is also possible but not recommended.

For example:

- Using /24 from FC::/8 ULA
- SRv6 Base Block = FC*BB*:*BB*::/24, with *B* indicating a nibble value picked by operator.

- SRv6 uSID Block = FC*BB*:*BBVV*/32, with *VV* indicating a nibble value picked by the operator.
  - 256 /32 uSID blocks possible from this allocation, from block 0 (FC*BB*:*BB**00***/32) to block 255(FC*BB*:*BB**FF***/32)
  - A network slice is assigned a /32 uSID block:
    - FCBB:BB**00**/32 for min-cost slice (shortest path based on minimum IS-IS cost)
    - FCBB:BB**08**/32 for min-delay slice (shortest path based on minimum latency using Flex Algo instance 128)

Platform-Specific Guidelines and Limitations

- SRv6 is supported on the following Cisco 8000 series Q200-based line cards and fixed-port routers:
  - Cisco 8800 with 88-LC0-36FH-M, 88-LC0-36FH, 88-LC0-34H14FH line cards
  - Cisco 8201-32FH
  - Cisco 8102-64H, 8101-32-FH
- SRv6 is not supported on Q100-based line cards and fixed-port routers.
- Egress marking on the outer header during SRv6 encapsulation operations (TI-LFA) is not supported.
- OAM: Ping and traceroute are supported.
- Cisco 8000 series routers support the following SRv6 uSID behaviors and variants:
  - **Endpoint behaviors**:
    - uN with PSP/USD
    - uA with PSP/USD
    - uDT4
    - uDT6
    - uDT46
  - **Head-end behaviors**:
    - H.Encap.Red (1 uSID carrier with up to 6 uSIDs)
    - H.Insert.Red is supported only on Cisco Silicon One P100-based routers.
- **Encapsulation Capabilities and Parameters**

  The following describes the Cisco 8000 series router capabilities for setting or propagating certain fields in the outer IPv6 header for SRv6 encapsulated packets:
  - **Source address**: Cisco 8000 series routers support a single source address (SA) for SRv6 encapsulated packets. The SA is derived from the SRv6 global configuration; if not configured, it is derived from the IPv6 Loopback address.
  - **Hop limit**:

- Overlay encapsulation

  Default: propagate=No

  The **hop-limit propagate** command enables propagation from inner header to outer header.

- Underlay encapsulation (TI-LFA) behavior is always in propagate mode, regardless of the CLI.

Manual configuration of the hop-limit value in the outer IPv6 header is not supported. See #unique_36 unique_36_Connect_42_section_mw5_4w5_qtb.

- **Traffic-class**:
  - Overlay encapsulation

    Default: propagate=No

    The **traffic-class propagate** command enables propagation from inner header to outer header.

  - Underlay encapsulation (TI-LFA) behavior is always in propagate mode, regardless of the CLI.

  Manual configuration of the traffic-class value in the outer IPv6 header is not supported. See #unique_36 unique_36_Connect_42_section_mw5_4w5_qtb.

- **Flow Label**:
  - Cisco 8000 series routers use the flow-label from the incoming IPv6 header. In case of USD operations, flow-label is used from the inner IPv6 header.
  - During H.Encap.Red operations, if the inner packet has a flow label (non-zero value), the Cisco 8000 series routers propagate it to the outer IPv6 header. If the flow label is not present (zero), it is computed.

- P role:
  - Underlay H-Encap: 6 sids (1 carrier with 6 sids per carrier)
- **PE role:**
  - Underlay H-Insert: 3 sids (1 carrier with 3 sids per carrier)
  - Overlay H-Encaps: 3 sids (1 carrier with 3 sids per carrier)

Configuring SRv6

Enabling SRv6 involves the following high-level configuration steps:

- Configure SRv6 locator(s)
- Enable SRv6 under IS-IS
- Enable SRv6 Services under BGP

Configure SRv6 Locator Name, Prefix, and uSID-Related Parameters

This section shows how to globally enable SRv6 and configure locator.

- **segment-routing srv6 locators locator** *locator*—Globally enable SRv6 and configure the locator.
- **segment-routing srv6 locators locator** *locator* **prefix** *ipv6_prefix*/*length*—Configure the locator prefix value.
- **segment-routing srv6 locators locator** *locator* **micro-segment behavior unode psp-usd**—Specifies the locator as a micro-segment (uSID) locator as well as specifies that IGP underlay uSID (uN/uA) variant is PSP-USD for this locator.

(Optional) Configure Algorithm Associated with Locator

- **segment-routing srv6 locators locator** *locator* **algorithm** *algo*—(Optional) Configure Algorithm associated with the locator. Valid values for *algo* are from 128 to 255.

  For additional information about SRv6 Flexible Algorithm, see Configuring SRv6 Flexible Algorithm under IS-IS, on page 32.

  For detailed information about Flexible Algorithm, see Enabling Segment Routing Flexible Algorithm, on page 489.

(Optional) Configure Anycast Locator

An SRv6 Anycast locator is a type of locator that identifies a set of nodes (uN SIDs). SRv6 Anycast Locators and their associated uN SIDs may be provisioned at multiple places in a topology.

The set of nodes (Anycast group) is configured to advertise a shared Anycast locator and uN SID. Anycast routing enables the steering of traffic toward multiple advertising nodes. Packets addressed to an Anycast address are forwarded to the topologically nearest nodes.

One use case is to advertise Anycast uN SIDs at exit points from an SRv6 network. Any of the nodes that advertise the common uN SID could be used to forward traffic out of the SRv6 portion of the network to the topologically nearest node.

The following behaviors apply to Anycast Locator:

- Unlike a normal locator, IS-IS does not program or advertise uA SIDs associated with an Anycast locator.
- uN SIDs allocated from Anycast locators will not be used in constructing TI-LFA backup paths or Microloop Avoidance primary paths. TI-LFA backup and Microloop Avoidance paths for an Anycast locator prefix may terminate on any node advertising that locator, which may be different from the node terminating the original primary path.
- SRv6 anycast locators may have non-zero algorithm (Flexible Algorithm) values.

Use the following commands to configure the Anycast locator and advertise Anycast prefixes associated with an interface.

- **segment-routing srv6 locators locator** *locator* **anycast**—Configure the Anycast locator
- **router isis** *instance-id* **interface Loopback** *instance* **prefix-attributes anycast level** *level*—Advertise the Anycast prefixes associated with an interface.

Example 1:

The following example shows how to globally enable SRv6 and configure a locator.

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc1
Router(config-srv6-locator)# micro-segment behavior unode psp-usd
Router(config-srv6-locator)# prefix 2001:0:8::/48
```

Example 2:

The following example shows how to configure Flexible Algorithm associated with locator.

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLocAlgo128
Router(config-srv6-locator)# algorithm 128
Router(config-srv6-locator)# micro-segment behavior unode psp-usd
Router(config-srv6-locator)# prefix 2001:0:88::/48
```

Example 3:

The following example shows how to configure Anycast locator.

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLocAnycast
Router(config-srv6-locator)# anycast
Router(config-srv6-locator)# micro-segment behavior unode psp-usd
Router(config-srv6-locator)# prefix 2001:0:100::/48
```

The following example shows how to advertise the Anycast prefixes associated with an interface.

```
Router(config)# router isis core
Router(config-isis)# interface Loopback100
Router(config-isis-if)# prefix-attributes anycast level 1
```

Example 4:

The following example shows how to configure SRv6-TE locator and binding SID (BSID) behavior.

```
Router#configure
Router(config)#segment-routing traffic-eng
Router(config-sr-te)#srv6 locator loc1 binding-sid dynamic behavior ub6-encaps-reduced
```

(Optional) Customize SRv6 Logging for Locator Status Changes

- **segment-routing srv6 logging locator status**—Enable the logging of locator status.

(Optional) Customize SRv6 SID Parameters

- **segment-routing srv6 sid holdtime** *minutes*—The holdtime for a stale or freed SID. The range of *minutes* is from 0 (disabled) to 60 minutes.

Example 4:

The following example shows how to configure optional SRv6 parameters:

```
RP/0/RSP0/CPU0:Node1(config)# segment-routing srv6
RP/0/RSP0/CPU0:Node1(config-srv6)# logging locator status
RP/0/RSP0/CPU0:Node1(config-srv6)# sid holdtime 10
```

```
RP/0/RSP0/CPU0:Node1(config-srv6)#
```

Verifying SRv6 Manager

This example shows how to verify the overall SRv6 state from SRv6 Manager point of view. The output displays parameters in use, summary information, and platform specific capabilities.

```
Router# SF-D#sh segment-routing srv6 manager
Parameters:
  SRv6 Enabled: No
  SRv6 Operational Mode: None
  Encapsulation:
    Source Address:
      Configured: ::
      Default: 77::77
    Hop-Limit: Default
    Traffic-class: Default
    SID Formats:
      f3216 <32B/16NFA> (2)
    uSID LIB Range:
    LIB Start : 0xe000
    ELIB Start : 0xfe00
    uSID WLIB Range:
    EWLIB Start : 0xfff7
Summary:
  Number of Locators: 0 (0 operational)
  Number of SIDs: 0 (0 stale)
  Max SID resources: 24000
  Number of free SID resources: 24000
  OOR:
    Thresholds (resources): Green 1200, Warning 720
    Status: Resource Available
      History: (0 cleared, 0 warnings, 0 full)
Platform Capabilities:
  SRv6: Yes
  TILFA: Yes
  Microloop-Avoidance: Yes
  Endpoint behaviors:
    End.DT6
    End.DT4
    End.DT46
    End (PSP/USD)
    End.X (PSP/USD)
    uN (PSP/USD)
    uA (PSP/USD)
    uDT6
    uDT4
    uDT46
Headend behaviors:
  H.Insert.Red
  H.Encaps.Red
Security rules:
  SEC-1
  SEC-2
  SEC-3
Counters:
  None
Signaled parameters:
  Max-SL : 3
  Max-End-Pop-SRH : 3
  Max-H-Insert : 0 sids
  Max-H-Encap : 2 sids
  Max-End-D : 5
```

```
Configurable parameters (under srv6):
  Ranges:
   LIB : Yes
   WLIB : Yes
  Encapsulation:
    Source Address: Yes
    Hop-Limit : value=No, propagate=Yes
    Traffic-class : value=No, propagate=Yes
    Default parameters (under srv6):
  Encapsulation:
    Hop-Limit : value=128, propagate=No
    Traffic-class : value=0, propagate=No
    Max Locators: 16
    Max SIDs: 24000
    SID Holdtime: 3 mins
Router# :SF-D#
```

Verifying SRv6 Locator

This example shows how to verify the locator configuration and its operational status.

```
Router# show segment-routing srv6 locator myLoc1 detail

Name                  ID       Algo  Prefix                    Status   Flags
--------------------  -------  ----  ------------------------  -------  --------
myLoc1                3        0     2001:0:8::/48             Up       U
  (U): Micro-segment (behavior: uN (PSP/USD))
  Interface:
    Name: srv6-myLoc1
    IFH : 0x02000120
    IPv6 address: 2001:0:8::/48
  Number of SIDs: 1
  Created: Dec 10 21:26:54.407 (02:52:26 ago)
```

Verifying SRv6 SIDs

This example shows how to verify the allocation of SRv6 local SIDs off locator(s).

```
Router# show segment-routing srv6 locator myLoc1 sid
SID                         Behavior          Context                           Owner
          State  RW
--------------------------  ----------------  ------------------------------
------------------  -----  --
2001:0:8::                  uN (PSP/USD)      'default':1                       sidmgr
          InUse  Y
```

The following example shows how to display detail information regarding an allocated SRv6 local SID.

```
Router# show segment-routing srv6 locator myLoc1 sid 2001:0:8:: detail
SID                         Behavior          Context                           Owner
          State  RW
--------------------------  ----------------  ------------------------------
------------------  -----  --
2001:0:8::                  uN (PSP/USD)      'default':8                       sidmgr
          InUse  Y
  SID Function: 0x8
  SID context: { table-id=0xe0800000 ('default':IPv6/Unicast), opaque-id=8 }
  Locator: 'myLoc1'
  Allocation type: Dynamic
  Created: Dec 10 22:10:51.596 (02:10:05 ago)
```

Similarly, you can display SID information across locators by using the **show segment-routing srv6 sid** command.

Configuring SRv6 under IS-IS

Intermediate System-to-Intermediate System (IS-IS) protocol already supports segment routing with MPLS dataplane (SR-MPLS). This feature enables extensions in IS-IS to support Segment Routing with IPv6 data plane (SRv6). The extensions include advertising the SRv6 capabilities of nodes and node and adjacency segments as SRv6 SIDs.

SRv6 IS-IS performs the following functionalities:

1. Interacts with SID Manager to learn local locator prefixes and announces the locator prefixes in the IGP domain.
2. Learns remote locator prefixes from other ISIS neighbor routers and installs the learned remote locator IPv6 prefix in RIB or FIB.
3. Allocate or learn prefix SID and adjacency SIDs, create local SID entries, and advertise them in the IGP domain.

Usage Guidelines and Restrictions

The following usage guidelines and restrictions apply for SRv6 IS-IS:

- An IS-IS address-family can support either SR-MPLS or SRv6, but both at the same time is not supported.

Configuring SRv6 IS-IS

To configure SRv6 IS-IS, you should enable SRv6 under the IS-IS IPv6 address-family. The following example shows how to configure SRv6 IS-IS.

```
Router(config)# router isis core
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc1
Router(config-isis-srv6-loc)# exit
```

Configuring SRv6 Flexible Algorithm under IS-IS

This feature introduces support for implementing Flexible Algorithm using IS-IS SRv6.

SRv6 Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

For detailed information about Flexible Algorithm, see Enabling Segment Routing Flexible Algorithm, on page 489.

Usage Guidelines and Restrictions

Observe the following usage guidelines and restrictions:

- You can configure up to 8 locators to support SRv6 Flexible Algorithm.
- The Flexible Algorithm locator prefix follows the same usage guidelines and restrictions of algo-0 locator prefixes. See Usage Guidelines and Limitations, on page 25.
- The Locator Algorithm value range is 128 to 255.

Configuring SRv6 Flexible Algorithm under IS-IS

The following sections show you the steps to enable SRv6 Flexible Algorithm. The example highlights a delay-based Flexible Algorithm instance.

1. Configure SRv6 locators
2. Assign SRv6 locators under IS-IS
3. Configure Flexible Algorithm definition and associated metric (for example, delay)
4. Configure the delay probe under the interface. For more information on SR performance measurement, see Configure Performance Measurement, on page 539.

The following section shows how to configure two SRv6 locators: one associated with Algo 0, and the other associated with Algo 128.

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLocBestEffort // best-effort locator
Router(config-srv6-locator)# micro-segment behavior unode psp-usd
Router(config-srv6-locator)# prefix 2001:0:1::/48
Router(config-srv6-locator)# exit

Router(config-srv6-locators)# locator myLocLowLat // low-latency (flex algo 128) locator
Router(config-srv6-locator)# micro-segment behavior unode psp-usd
Router(config-srv6-locator)# prefix 2001:0:2::/48
Router(config-srv6-locator)# algorithm 128
Router(config-srv6-locator)# exit
Router(config-srv6)# exit
```

The following section shows how to assign multiple SRv6 locators under IS-IS.

```
Router(config)# router isis core
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLocBestEffort
Router(config-isis-srv6-loc)# exit
Router(config-isis-srv6)# locator myLocLowLat
Router(config-isis-srv6-loc)# exit
```

The following section shows how to configure the Flexible Algorithm definition.

```
Router(config)# router isis core
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# metric-type delay
```

```
Router(config-isis-flex-algo)# exit
Router(config-isis)# interface GigabitEthernet0/0/0/0
Router(config-isis-if)# address-family ipv6 unicast
```

The following section shows how to configure the delay probe under the interface.

```
Router(config)# performance-measurement
Router(config-perf-meas)# interface GigabitEthernet0/0/0/0
Router(config-pm-intf)# delay-measurement
Router(config-pm-intf-dm)# commit
```

Verification

```
Router# show segment-routing srv6 locator

Name                  ID       Algo  Prefix                    Status   Flags
--------------------  -------  ----  ------------------------  -------  --------
myLoc1                3        0     2001:0:8::/48             Up       U
myLocBestEffort       5        0     2001:0:1::/48             Up       U
myLocLowLat           4        128   2001:0:2::/48             Up       U


Router# show isis flex-algo 128

IS-IS core Flex-Algo Database

Flex-Algo 128:

Level-2:
        Definition Priority: 128
        Definition Source: Router.00, (Local)
        Definition Equal to Local: Yes
        Disabled: No


Level-1:
        Definition Priority: 128
        Definition Source: Router.00, (Local)
        Definition Equal to Local: Yes
        Disabled: No


    Local Priority: 128
    FRR Disabled: No
    Microloop Avoidance Disabled: No
```

Configuring SRv6 Locator Prefix Summarization

SRv6 leverages longest-prefix-match IP forwarding. Massive-scale reachability can be achieved by summarizing locators at ABRs and ASBRs.

Use the **summary-prefix** *locator* [**algorithm** *algo*] [**explicit**] command in IS-IS address-family configuration mode to specify that only locators from the specified algorithm contribute to the summary. The **explicit** keyword limits the contributing prefixes to only those belonging to the same algorithm.

The following example shows how to configure SRv6 IS-IS Algorithm Summarization for regular algorithm and Flexible Algorithm (128).

```
Router(config)# router isis core
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# summary-prefix 2001:0:1::/48
Router(config-isis-af)# summary-prefix 2001:0:2::/48 algorithm 128 explicit
```

Configuring TI-LFA with SRv6 IS-IS

This feature introduces support for implementing Topology-Independent Loop-Free Alternate (TI-LFA) using SRv6 IS-IS.

TI-LFA provides link 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. TI-LFA leverages the post-convergence path which is planned to carry the traffic and ensures link and node protection within 50 milliseconds. TI-LFA with IS-IS SR-MPLS is already supported.

TI-LFA provides link, node, and Shared Risk Link Groups (SRLG) protection in any topology.

For more information, see Configure Topology-Independent Loop-Free Alternate (TI-LFA), on page 611.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- TI-LFA provides link protection by default. Additional tiebreaker configuration is required to enable node or SRLG protection.
- Usage guidelines for node and SRLG protection:
  - TI-LFA node protection functionality 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.
  - 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.

Configuring SRv6 IS-IS TI-LFA

The following example shows how to configure different types of TI-LFA protection for SRv6 IS-IS.

```
Router(config)# router isis core
Router(config-isis)# interface bundle-ether 1201
Router(config-isis-if)# address-family ipv6 unicast
```

```
Router(config-isis-if-af)# fast-reroute per-prefix
Router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
Router(config-isis)# interface bundle-ether 1301
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# fast-reroute per-prefix
Router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Router(config-isis-if-af)# fast-reroute per-prefix tiebreaker node-protecting index 100
Router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint index 200
Router(config-isis-if-af)# exit
```

Configuring SRv6 IS-IS TI-LFA with Flexible Algorithm

TI-LFA backup paths for particular Flexible Algorithm are computed using the same constraints as the calculation of the primary paths for such Flexible Algorithm. These paths use the locator prefix advertised specifically for such Flexible Algorithm in order to enforce a backup path.

By default, LFA/TI-LFA for SRv6 Flexible Algorithm uses the LFA/TI-LFA configuration of Algo 0.

Use the **fast-reroute disable** command to disable the LFA/TI-LFA calculation on a per-algorithm basis:

```
Router(config)# router isis core
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# fast-reroute disable
```

Verification

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show isis ipv6 fast-reroute** *ipv6-prefix* **detail** command.

```
Router# show isis ipv6 fast-reroute cafe:0:2::2/128 detail

L2 cafe:0:2::2/128 [20/115] Label: None, medium priority
     via fe80::e00:ff:fe3a:c700, HundredGigE0/0/0/0, Node2, Weight: 0
       Backup path: TI-LFA (link), via fe80::1600:ff:feec:fe00, HundredGigE0/0/0/1 Node3,
Weight: 0, Metric: 40
         P node: Node4.00 [cafe:0:4::4], SRv6 SID: cafe:0:4:: uN (PSP/USD)
         Backup-src: Node2.00
       P: No, TM: 40, LC: No, NP: No, D: No, SRLG: Yes
     src Node2.00-00, cafe:0:2::2
```

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show route ipv6** *ipv6-prefix* **detail** command.

```
Router# show route ipv6 cafe:0:2::2/128 detail
Tue Feb 23 23:08:48.151 UTC

Routing entry for cafe:0:2::2/128
  Known via "isis 1", distance 115, metric 20, type level-2
  Installed Feb 23 22:57:38.900 for 00:11:09
  Routing Descriptor Blocks
    fe80::1600:ff:feec:fe00, from cafe:0:2::2, via HundredGigE0/0/0/1, Backup (TI-LFA)
      Repair Node(s): cafe:0:4::4
      Route metric is 40
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:65              Path ref count:1
      NHID:0x20002(Ref:19)
```

```
      SRv6 Headend: H.Encaps.Red, SID-list {cafe:0:4::}
    fe80::e00:ff:fe3a:c700, from cafe:0:2::2, via HundredGigE0/0/0/0, Protected
      Route metric is 20
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:0
      NHID:0x20001(Ref:19)
      Backup path id:65
  Route version is 0x4 (4)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_NON_RECURSIVE_MEDIUM (7) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 1, Download Version 66
  No advertising protos.
```

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show cef ipv6** *ipv6-prefix* **detail location** *location* command.

```
Router# show cef ipv6 cafe:0:2::2/128 detail location 0/0/cpu0
Tue Feb 23 23:09:07.719 UTC
cafe:0:2::2/128, version 66, SRv6 Headend, internal 0x1000001 0x210 (ptr 0x8e96fd2c) [1],
0x0 (0x8e93fae0), 0x0 (0x8f7510a8)
 Updated Feb 23 22:57:38.904
 local adjacency to HundredGigE0/0/0/0

 Prefix Len 128, traffic index 0, precedence n/a, priority 1
  gateway array (0x8e7b5c78) reference count 1, flags 0x500000, source rib (7), 0 backups
                [2 type 3 flags 0x8401 (0x8e86ea40) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8e93fae0, sh-ldi=0x8e86ea40]
  gateway array update type-time 1 Feb 23 22:57:38.904
 LDI Update time Feb 23 22:57:38.913
 LW-LDI-TS Feb 23 22:57:38.913
   via fe80::1600:ff:feec:fe00/128, HundredGigE0/0/0/1, 9 dependencies, weight 0, class 0,
 backup (TI-LFA) [flags 0xb00]
    path-idx 0 NHID 0x20002 [0x8f5850b0 0x0]
    next hop fe80::1600:ff:feec:fe00/128, Repair Node(s): cafe:0:4::4
    local adjacency
    SRv6 H.Encaps.Red SID-list {cafe:0:4::}
   via fe80::e00:ff:fe3a:c700/128, HundredGigE0/0/0/0, 6 dependencies, weight 0, class 0,
protected [flags 0x400]
    path-idx 1 bkup-idx 0 NHID 0x20001 [0x8f8420b0 0x0]
    next hop fe80::e00:ff:fe3a:c700/128

    Load distribution: 0 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/0        fe80::e00:ff:fe3a:c700
```

Configuring SRv6 IS-IS Microloop Avoidance

This feature introduces support for implementing microloop avoidance using IS-IS SRv6.

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 SRv6 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.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- The Routing Information Base (RIB) update delay value specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The *delay-time* range is from 1 to 60000 milliseconds; the default value is 5000.

Configuring SRv6 IS-IS Microloop Avoidance

The following example shows how to configure SRv6 IS-IS Microloop Avoidance and set the Routing Information Base (RIB) update delay value.

Note Complete the Configuring SRv6, on page 27 before performing these steps.

```
Router(config)# router isis test-igp
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# microloop avoidance segment-routing
Router(config-isis-af)# microloop avoidance rib-update-delay 2000
Router(config-isis-af)# commit
```

Configuring SRv6 BGP-Based Services

Table 4: Feature History Table

Feature Name	Release	Description
Support for uDT46 SRv6 Endpoint Behavior	Release 7.11.1 Release 7.5.3	This feature adds support for the “Endpoint with decapsulation and specific IP table lookup” SRv6 end-point behavior (uDT46). The End.DT46 behavior is used for dual-stack L3VPNs. This behavior is equivalent to the single per-VRF VPN label (for IPv4 and IPv6) in MPLS.

Feature Name	Release	Description
VRF Allocation Mode for uDT46 Endpoint Behavior	Release 7.11.1	This feature introduces a new VRF allocation mode for uDT46 SIDs for the following BGP-based services: • IPv4 Layer-3 VPNs • IPv6 Layer-3 VPNs • IPv4 BGP global • IPv6 BGP global • L3 EVPN This allocation mode allows for both IPv4/IPv6 address families, and VPNv4/v6 address families, to use a single SID. When this allocation mode is configured under an address family, CE-learned routes, redistributed routes, aggregated routes, local routes, and imported routes will use uDT46 SID when advertised to remote peers. The feature introduces these changes: **CLI:** • The **per-vrf-46** allocation mode is introduced in the following commands: • **router bgp** *as-number* **address-family** {**ipv4** | **ipv6**} **unicast segment-routing srv6 alloc mode per-vrf-46** • **router bgp** *as-number* **address-family** {**vpnv4** | **vpnv6**} **unicast vrf all segment-routing srv6 alloc mode per-vrf-46** • **router bgp** *as-number* **vrf** *WORD* **address-family** {**ipv4** | **ipv6**} **unicast segment-routing srv6 alloc mode per-vrf-46**

Building on the messages and procedures defined in IETF draft "BGP/MPLS IP Virtual Private Networks (VPNs)", BGP has been extended to provide services over an SRv6 network, such as:

- IPv4 Layer-3 VPNs
- IPv6 Layer-3 VPNs
- IPv4 BGP global
- IPv6 BGP global

- Layer-2 VPNs - Ethernet VPNs (EVPN)

For more information about BGP, refer to the "Implementing BGP" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

In SRv6-based services, the egress PE signals an SRv6 Service SID with the BGP service route. The ingress PE encapsulates the payload in an outer IPv6 header where the destination address is the SRv6 Service SID advertised by the egress PE. BGP messages between PEs carry SRv6 Service SIDs as a means to interconnect PEs and form VPNs. SRv6 Service SID refers to a segment identifier associated with one of the SRv6 service-specific behaviors advertised by the egress PE router, such as:

- uDT4 (Endpoint with decapsulation and IPv4 table lookup)
- uDT6 (Endpoint with decapsulation and IPv6 table lookup)
- uDT46 (Endpoint with decapsulation and specific IP table lookup)
- uDX4 (Endpoint with decapsulation and IPv4 cross-connect)
- uDX6 (Endpoint with decapsulation and IPv6 cross-connect)

Based on the messages and procedures defined in IETF draft "SRv6 BGP based Overlay services", BGP encodes the SRv6 Service SID in the prefix-SID attribute of the corresponding BGP Network Layer Reachability Information (NLRI) and advertises it to its IPv6 BGP peers.

Usage Guidelines and Restrictions

- The following SRv6 BGP-based services are supported:
  - IPv4 Layer-3 VPNs
  - IPv6 Layer-3 VPNs
  - IPv4 BGP global
  - IPv6 BGP global
- uDT4, uDT6, and uDT46 for L3VPN and BGP global are supported.
- Dual-Stack L3 Services (IPv4 L3VPN, IPv6 L3VPN, IPv4 BGP global, IPv6 BGP global) are supported.

SRv6 Locator Inheritance Rules

SRv6 locators can be assigned at different levels inside the BGP routing process. BGP allocates SRv6 Service SIDs from configured locator spaces according to the following inheritance rules:

1. Use the locator as defined under the service.

   If not defined under the specific service, then:

2. Use the locator as defined under the corresponding address-family.

   If not defined under the corresponding address-family, then:

3. Use the locator as defined globally under BGP.

Enabling SRv6 Globally under BGP

Use the **router bgp** *as-number* **segment-routing srv6** command to enable SRv6 globally under the BGP routing process. The *as-number* is from 1-65535.

```
RP/0/0/CPU0:Node1(config)# router bgp 100 segment-routing srv6
```

Assigning SRv6 Locator Globally under BGP

Use the **router bgp** *as-number* **segment-routing srv6 locator** *WORD* command to assign an SRv6 locator globally under the BGP routing process. The *as-number* is from 1-65535.

This example shows how to assign a locator:

```
RP/0/0/CPU0:Node1(config)# router bgp 100 segment-routing srv6 locator Node1-locator
```

For more information on how to configure an SRv6 locator, see Configuring SRv6, on page 27.

For more information on how to assign an SRv6 locator under the BGP service or BGP address-family, see the following SRv6 Services sections.

SRv6 Services: IPv4 L3VPN

Table 5: Feature History Table

Feature Name	Release	Description
Dual-Stack L3VPN Services (IPv4, IPv6) (SRv6 Micro-SID)	Release 7.8.1	This feature introduces support for Dual-stack (VPNv4/VPNv6) VRFs. VPNv4/VPNv6 Dual-stack supports both IPv4 (uDT4) and IPv6 (uDT6) based SRv6 L3VPN service on the same interface, sub-interface, or VRF. Dual stacking allows operators to access both IPv4 and IPv6 simultaneously and independent of each other. It avoids the need to translate between two protocol stacks. This results in high processing efficiency and zero information loss.

Feature Name	Release	Description
Per-Prefix SRv6 Locator Assignment	Release 7.8.1	This feature allows you to assign a specific SRv6 locator for a given prefix or a set of prefixes (IPv4/IPv6 GRT, IPv4/IPv6 VPN). The egress PE advertises the prefix with the specified locator. This allows for per-prefix steering into desired transport behaviors, such as Flex Algo.
Support for iBGP as PE-CE protocol	Release 7.8.1	This feature introduces support for iBGP as PE-CE protocol.
SRv6 VPN BGP Route Leaking	Release 7.8.1	This feature supports SRv6 VPN Route-leaking between Global Routing Table (GRT) and Virtual Routing and Forwarding (VRF). This enables Enterprise IPv4 internet connectivity.

This feature provides IPv4 L3VPNs (VPNv4) over an SRv6 network.

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF or per-prefix.
- Per-VRF allocation mode is supported (uDT4 behavior)
- Per-VRF-46 allocation mode is supported (uDT46 behavior)
- Dual-Stack L3VPN Services (IPv4, IPv6) are supported
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- eBGP, OSPF, Static are supported as PE-CE protocol.
- BGP (iBGP, eBGP), OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *BGP Configuration Guide for Cisco 8000 Series Routers*.
- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.
- Per-CE allocation mode is not supported (uDX4 behavior)

Per-VRF-46 Allocation Mode

In traditional routing protocols, each route is typically identified by its unique IP address. This means that routers must maintain separate entries in their routing tables for each route. As the number of routes increases, the routing tables become larger and more complex, which can impact the efficiency and scalability of the network.

Starting Cisco IOS XR 7.11.1, when the "per-vrf-46" allocation mode is configured under an address family, or both address families, in BGP, several types of routes (CE learned route, redistributed route, aggregated

route, Local route, and imported route) use the specific identifier "uDT46 SID" when they are advertised to remote peers.

By using the same uDT46 SID for multiple routes, these routes can be aggregated and treated as a single entity during forwarding decisions. This aggregation is based on common characteristics or attributes shared by those routes, such as the same VRF or the same address family.

When BGP requests the SID, the SID manager provides information such as Locator, behavior (uDT46), WLIB/LIB indication, and VRF name. The SID manager also determines whether to return an explicitly configured SID or a dynamic SID.

If all the provided information (Locator/behavior/WLIB/LIB/VRF name) matches with a configured explicit SID, that explicit SID is returned. However, if there is no match, a dynamic SID is returned instead.

Configuring SRv6 based IPv4 L3VPN

To enable SRv6-based L3VPN, you need to enable SRv6 under BGP, specify the locator, and configure the SID allocation mode. The assignment of the locator can be done in different places under the **router bgp** configuration. See #concept_f11_rmx_lvb_8k.

Use Case 1: Assigning SRv6 Locator Globally

This example shows how to enable SRv6 and configure the SRv6 locator name under BGP Global:

```
Node1(config)# router bgp 100
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1-locator
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family vpnv4 unicast
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv4 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust1
Node1(config-bgp-vrf)# rd 100:1
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# commit
```

Running Config

```
router bgp 100
 segment-routing srv6
  locator Node1-locator
 !
 address-family vpnv4 unicast
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv4 unicast
  !
 !
 vrf vrf_cust1
  rd 100:1
  address-family ipv4 unicast
  !
 !
!
end
```

Use Case 2: Assigning SRv6 Locator for All VRFs

To configure the SRv6 locator for all VRFs under VPNv4 Address Family and specify the allocation mode, use the following commands:

- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6 alloc mode** {**per-vrf** | **per-vrf-46**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT4 behavior) be used for all the routes advertised from a unique VRF.
  - Use the **per-vrf-46** keyword to specify that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF. BGP will program two paths for this SID route: one for VPNv4 table and one for VPNv6 table.
- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6 locator** *WORD*: Specify the locator

This example shows how to enable SRv6 and configure the SRv6 locator for all VRFs under VPNv4 Address Family, with per-VRF label allocation mode:

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family vpnv4 unicast
Node1(config-bgp-af)# vrf all
Node1(config-bgp-af-vrfall)# segment-routing srv6
Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf
Node1(config-bgp-af-vrfall-srv6)# exit
Node1(config-bgp-af-vrfall)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv4 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust1
Node1(config-bgp-vrf)# rd 100:1
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# commit
```

Running Config

```
router bgp 100
 address-family vpnv4 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf
   !
  !
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv4 unicast
  !
 !
 vrf vrf_cust1
  rd 100:1
```

```
  address-family ipv4 unicast
  !
 !
!
end
```

This example shows how to enable SRv6 and configure the SRv6 locator for all VRFs under VPNv4/v6 Address Family, with per-VRF-46 label allocation mode:

```
Node1(config)# router bgp 200
Node1(config-bgp)# address-family vpnv4 unicast
Node1(config-bgp-af)# vrf all
Node1(config-bgp-af-vrfall)# segment-routing srv6
Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-vrfall-srv6)# exit
Node1(config-bgp-af-vrfall)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# address-family vpnv6 unicast
Node1(config-bgp-af)# vrf all
Node1(config-bgp-af-vrfall)# segment-routing srv6
Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-vrfall-srv6)# exit
Node1(config-bgp-af-vrfall)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv4 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust1
Node1(config-bgp-vrf)# rd 100:1
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# commit
```

Running Config

```
router bgp 200
 address-family vpnv4 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
 !
 address-family vpnv6 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv4 unicast
  !
 !
 vrf vrf_cust1
  rd 100:1
  address-family ipv4 unicast
```

```
  !
 !
!
end
```

Use Case 3: Assigning SRv6 Locator for a specific VRF

To configure the SRv6 locator for a specific VRF under IPv4 Address Family and specify the allocation mode, use the following commands:

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6 alloc mode** { **per-vrf** | **per-vrf-46**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT4 behavior) be used for all the routes advertised from a unique VRF.
  - Use the **per-vrf-46** keyword to specify that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF. BGP will program two paths for this SID route: one for VPNv4 table and one for VPNv6 table.
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6 locator** *WORD*: Specify the locator

This example shows how to configure the SRv6 locator for an individual VRF, with per-VRF label allocation mode:

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family vpnv4 unicast
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv4 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust1
Node1(config-bgp-vrf)# rd 100:1
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf
Node1(config-bgp-vrf-af-srv6)# commit
```

Running Config

```
router bgp 100
 address-family vpnv4 unicast
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv4 unicast
  !
 !
 vrf vrf_cust1
  rd 100:1
  address-family ipv4 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf
   !
```

```
  !
 !
!
end
```

This example shows how to configure the SRv6 locator for an individual VRF, for both IPv4 and IPv6 address families, with per-VRF-46 label allocation mode:

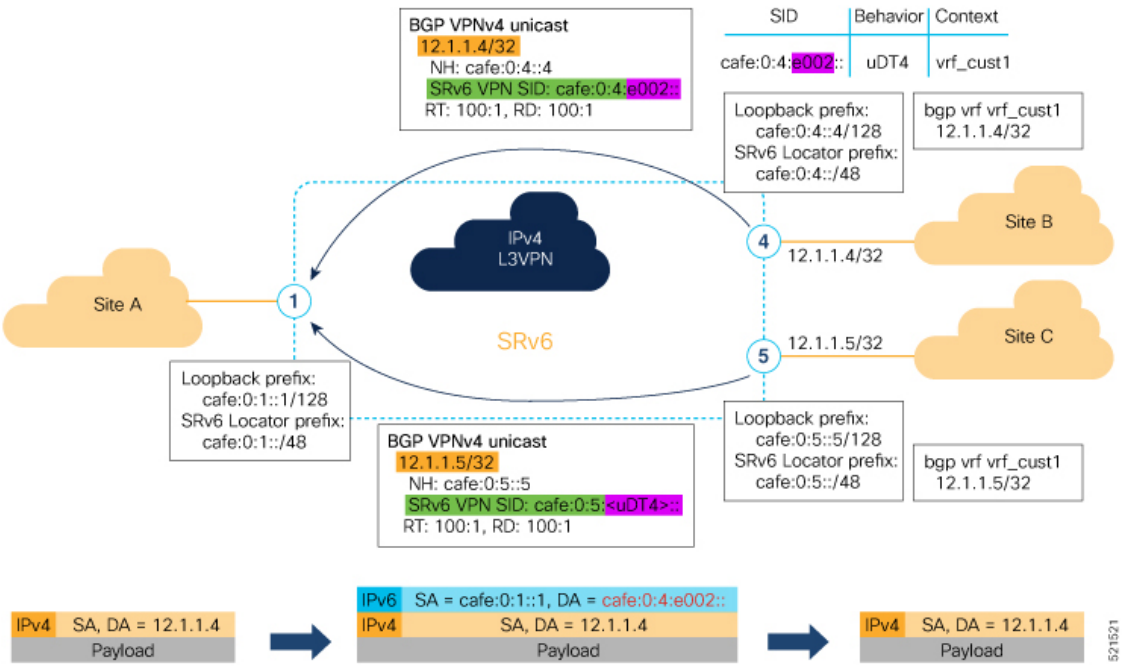
```
Node1(config)# router bgp 200
Node1(config-bgp)# address-family vpnv4 unicast
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv4 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust1
Node1(config-bgp-vrf)# rd 100:1
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-vrf-af-srv6)# exit
Node1(config-bgp-vrf-af)# exit
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-vrf-af-srv6)# commit
```

Running Config

```
router bgp 200
 address-family vpnv4 unicast
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv4 unicast
  !
 !
 vrf vrf_cust1
  rd 100:1
  address-family ipv4 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
  address-family ipv6 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
 !
!
end
```

Verification

The following figure shows a VPNv4 scenario. The sequence of commands included correspond to router Node1 acting as Ingress PE, and routers Node4 and Node5 acting as Egress PEs.



The following example shows how to verify the SRv6 based L3VPN configuration using the **show segment-routing srv6 sid** command.

In this example, we can observe the uDT4 SIDs associated with the IPv4 L3VPN; where uDT4 behavior represents Endpoint with decapsulation and IPv4 table lookup.

```
Node1# show segment-routing srv6 sid

*** Locator: 'Node1-locator' ***

SID                         Behavior          Context                           Owner
          State  RW
--------------------------  ----------------  ------------------------------
------------------  -----  --
cafe:0:1::                  uN (PSP/USD)      'default':1                       sidmgr
          InUse  Y
cafe:0:1:e000::             uA (PSP/USD)      [Hu0/0/0/0, Link-Local]:0         isis-1
          InUse  Y
cafe:0:1:e001::             uA (PSP/USD)      [Hu0/0/0/1, Link-Local]:0         isis-1
          InUse  Y
cafe:0:1:e002::             uDT4              'vrf_cust1'                       bgp-100
          InUse  Y
cafe:0:1:e003::             uDT4              'vrf_cust2'                       bgp-100
          InUse  Y
cafe:0:1:e004::             uDT4              'vrf_cust3'                       bgp-100
          InUse  Y
cafe:0:1:e005::             uDT4              'vrf_cust4'                       bgp-100
          InUse  Y
cafe:0:1:e006::             uDT4              'vrf_cust5'                       bgp-100
          InUse  Y
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show segment-routing srv6***SID-prefix***detail** command.

```
Node1# show segment-routing srv6 sid cafe:0:1:e002:: detail
Tue Feb  9 17:50:40.621 UTC
```

```
*** Locator: 'Node1-locator' ***

SID                         Behavior          Context                                 Owner
               State  RW
--------------------------  ----------------  ------------------------------
------------------  -----  --
cafe:0:1:e002::             uDT4              'vrf_cust1'                             bgp-100
             InUse  Y
  SID Function: 0xe002
  SID context: { table-id=0xe0000011 ('vrf_cust1':IPv4/Unicast) }
  Locator: 'Node1-locator'
  Allocation type: Dynamic
  Created: Feb  9 17:41:07.475 (00:09:33 ago)
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show bgp vpnv4 unicast** commands on Egress PE.

```
Node1# show bgp vpnv4 unicast summary

BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 36
BGP NSR Initial initsync version 16 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker              36         36         36         36          36           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
cafe:0:4::4       0   100      47      48       36    0    0 00:40:05          5
cafe:0:5::5       0   100      47      47       36    0    0 00:39:56          5


Node1# show bgp vpnv4 unicast rd 100:1

BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 36
BGP NSR Initial initsync version 16 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf vrf_cust1)
*> 12.1.1.1/32        0.0.0.0                  0         32768 ?
*>i12.4.4.4/32        cafe:0:4::4              0    100      0 ?
*>i12.5.5.5/32        cafe:0:5::5              0    100      0 ?

Processed 3 prefixes, 3 paths
```

```
Node1# show bgp vpnv4 unicast rd 100:1 12.4.4.4/32

BGP routing table entry for 12.4.4.4/32, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                22          22
Last Modified: Feb 23 22:57:56.756 for 00:40:08
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (received & used)
    cafe:0:4::4 (metric 30) from cafe:0:4::4 (1.1.1.4)
      Received Label 0xe00400
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 22
      Extended community: RT:1:1 RT:100:1
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:63, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv4 Unicast, Source VRF: vrf_cust1, Source Route Distinguisher: 100:1
```

The following examples show how to verify the BGP prefix information for VRF instances using the **show bgp vrf** commands:

```
Node1# show bgp vrf vrf_cust1 ipv4 unicast

BGP VRF vrf_cust1, state: Active
BGP Route Distinguisher: 100:1
VRF ID: 0x60000002
BGP router identifier 1.1.1.1, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 32
BGP main routing table version 36
BGP NSR Initial initsync version 16 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf vrf_cust1)
*> 12.1.1.1/32        0.0.0.0                  0         32768 ?
*>i12.4.4.4/32        cafe:0:4::4              0    100      0 ?
*>i12.5.5.5/32        cafe:0:5::5              0    100      0 ?

Processed 3 prefixes, 3 paths


Node1# show bgp vrf vrf_cust1 ipv4 unicast 12.4.4.4/32
Tue Feb 23 23:39:57.499 UTC
BGP routing table entry for 12.4.4.4/32, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                22          22
Last Modified: Feb 23 22:57:56.756 for 00:42:01
Paths: (1 available, best #1)
```

```
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (received & used)
    cafe:0:4::4 (metric 30) from cafe:0:4::4 (1.1.1.4)
      Received Label 0xe00400
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 22
      Extended community: RT:1:1 RT:100:1
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:63, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv4 Unicast, Source VRF: vrf_cust1, Source Route Distinguisher: 100:1
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show route vrf** commands.

```
Node1# show route vrf vrf_cust1

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is not set

L    12.1.1.1/32 is directly connected, 00:44:43, Loopback100
B    12.4.4.4/32 [200/0] via cafe:0:4::4 (nexthop in vrf default), 00:42:45
B    12.5.5.5/32 [200/0] via cafe:0:5::5 (nexthop in vrf default), 00:42:45


Node1# show route vrf vrf_cust1 12.4.4.4/32

Routing entry for 12.4.4.4/32
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Feb 23 22:57:56.746 for 00:43:12
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
  No advertising protos.


Node1# show route vrf vrf_cust1 12.4.4.4/32 detail

Routing entry for 12.4.4.4/32
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Feb 23 22:57:56.746 for 00:43:37
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
```

```
    Extended communities count: 0
    Source RD attributes: 0x0000:100:1
    NHID:0x0(Ref:0)
    SRv6 Headend: H.Encaps.Red [f3216], SID-list {cafe:0:4:e004::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_REMOTE
  Download Priority 3, Download Version 3
  No advertising protos.
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show cef vrf** commands.

```
Node1# show cef vrf vrf_cust1

Prefix              Next Hop            Interface
------------------- ------------------- ------------------
0.0.0.0/0           drop                default handler
0.0.0.0/32          broadcast
12.1.1.1/32         receive             Loopback100
12.4.4.4/32         cafe:0:4::/128      <recursive>
12.5.5.5/32         cafe:0:5::/128      <recursive>
224.0.0.0/4         0.0.0.0/32
224.0.0.0/24        receive
255.255.255.255/32  broadcast


Node1# show cef vrf vrf_cust1 12.4.4.4/32

12.4.4.4/32, version 3, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x78b9a61c) [1], 0x0
(0x0), 0x0 (0x88873720)
 Updated Feb 23 22:57:56.749
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
   via cafe:0:4::/128, 3 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78e2da14 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:4::/128 via cafe:0:4::/48
    SRv6 H.Encaps.Red SID-list {cafe:0:4:e004::}


Node1# show cef vrf vrf_cust1 12.4.4.4/32 detail

12.4.4.4/32, version 3, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x78b9a61c) [1], 0x0
(0x0), 0x0 (0x88873720)
 Updated Feb 23 22:57:56.749
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0x88a740a8) reference count 5, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x789cbcc8) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Feb 23 22:57:56.749
 LDI Update time Feb 23 22:57:56.754

  Level 1 - Load distribution: 0
  [0] via cafe:0:4::/128, recursive

   via cafe:0:4::/128, 3 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78e2da14 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:4::/128 via cafe:0:4::/48
```

```
SRv6 H.Encaps.Red SID-list {cafe:0:4:e004::}

Load distribution: 0 1 (refcount 1)

Hash  OK  Interface                 Address
0     Y   HundredGigE0/0/0/1        remote
1     Y   HundredGigE0/0/0/0        remote
```

SRv6 Services: IPv6 L3VPN

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: IPv6 L3VPN	Release 7.8.1	With this feature, the egress PE can signal an SRv6 Service SID with the BGP overlay service route. The ingress PE encapsulates the IPv4/IPv6 payload in an outer IPv6 header where the destination address is the SRv6 Service SID provided by the egress PE. BGP messages between PEs carry SRv6 Service SIDs to interconnect PEs and form VPNs.

This feature provides IPv6 L3VPNs (VPNv6) over an SRv6 network.

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF, or per-prefix.
- Per-VRF allocation mode is supported (uDT6 behavior)
- Per-VRF-46 allocation mode is supported (uDT46 behavior) - see Per-VRF-46 Allocation Mode
- Dual-Stack L3VPN Services (IPv4, IPv6) are supported
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP (iBGP, eBGP), OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers* .
- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.
- Per-CE allocation mode is not supported (uDX6 behavior)

Configuring SRv6-based IPv6 L3VPN

To enable SRv6-based L3VPN, you need to enable SRv6 under BGP, specify the locator, and configure the SID allocation mode. The assignment of the locator can be done in different places under the **router bgp** configuration. See SRv6 Locator Inheritance Rules, on page 40.

Use Case 1: Assigning SRv6 Locator Globally

This example shows how to configure the SRv6 locator name under BGP Global:

```
Node1(config)# router bgp 100
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1-locator
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family vpnv6 unicast
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::12:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv6 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust6
Node1(config-bgp-vrf)# rd 100:6
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# commit
```

Running Configuration

```
router bgp 100
 segment-routing srv6
  locator Node1-locator
 !
 address-family vpnv6 unicast
 !
 neighbor 3001::12:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
  !
 !
!
end
```

Use Case 2: Assigning SRv6 Locator for All VRFs

To configure the SRv6 locator for all VRFs under VPNv6 Address Family and specify the allocation mode, use the following commands:

- **router bgp** *as-number* **address-family vpnv6 unicast vrf all segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family vpnv6 unicast vrf all segment-routing srv6 alloc mode** {**per-vrf** | **per-vrf-46**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT6 behavior) be used for all the routes advertised from a unique VRF.
  - Use the **per-vrf-46** keyword to specify that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF. BGP will program two paths for this SID route: one for VPNv4 table and one for VPNv6 table.
- **router bgp** *as-number* **address-family vpnv6 unicast vrf all segment-routing srv6 locator** *WORD*: Specify the locator

This example shows how to configure the SRv6 locator for all VRFs under VPNv6 Address Family, with per-VRF label allocation mode:

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family vpnv6 unicast
Node1(config-bgp-af)# vrf all
Node1(config-bgp-af-vrfall)# segment-routing srv6
Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf
Node1(config-bgp-af-vrfall-srv6)# exit
Node1(config-bgp-af-vrfall)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::12:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv6 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust6
Node1(config-bgp-vrf)# rd 100:6
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# commit
```

Running Configuration

```
router bgp 100
 address-family vpnv6 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf
   !
  !
 !
 neighbor 3001::12:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
  !
 !
!
end
```

This example shows how to configure the SRv6 locator for all VRFs under VPNv4/v6 Address Family, with per-VRF-46 label allocation mode:

```
Node1(config)# router bgp 200
Node1(config-bgp)# address-family vpnv4 unicast
Node1(config-bgp-af)# vrf all
Node1(config-bgp-af-vrfall)# segment-routing srv6
Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-vrfall-srv6)# exit
Node1(config-bgp-af-vrfall)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# address-family vpnv6 unicast
Node1(config-bgp-af)# vrf all
Node1(config-bgp-af-vrfall)# segment-routing srv6
Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf-46
```

```
Node1(config-bgp-af-vrfall-srv6)# exit
Node1(config-bgp-af-vrfall)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::12:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv6 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust6
Node1(config-bgp-vrf)# rd 100:6
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# commit
```

Running Configuration

```
router bgp 200
 address-family vpnv4 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
 !
 address-family vpnv6 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
 !
 neighbor 3001::12:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
  !
 !
!
end
```

Use Case 3: Assigning SRv6 Locator for a specific VRF

To configure the SRv6 locator for a specific VRF under IPv6 Address Family and specify the allocation mode, use the following commands:

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6 alloc mode** { **per-vrf** | **per-vrf-46**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT6 behavior) be used for all the routes advertised from a unique VRF.
  - Use the **per-vrf-46** keyword to specify that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF. BGP will program two paths for this SID route: one for VPNv4 table and one for VPNv6 table.

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6 locator** *WORD*: Specify the locator

This example shows how to configure the SRv6 locator for an individual VRF, with per-VRF label allocation mode:

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family vpnv6 unicast
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::12:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv6 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust6
Node1(config-bgp-vrf)# rd 100:6
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf
Node1(config-bgp-vrf-af-srv6)# commit
```

Running Configuration

```
router bgp 100
 address-family vpnv6 unicast
 !
 neighbor 3001::12:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf
   !
  !
 !
!
end
```

This example shows how to configure the SRv6 locator for an individual VRF, for both IPv4 and IPv6 address families, with per-VRF-46 label allocation mode:

```
Node1(config)# router bgp 200
Node1(config-bgp)# address-family vpnv6 unicast
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::12:1:1:4
Node1(config-bgp-nbr)# remote-as 100
Node1(config-bgp-nbr)# address-family vpnv6 unicast
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# vrf vrf_cust6
Node1(config-bgp-vrf)# rd 100:6
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-vrf-af-srv6)# exit
```

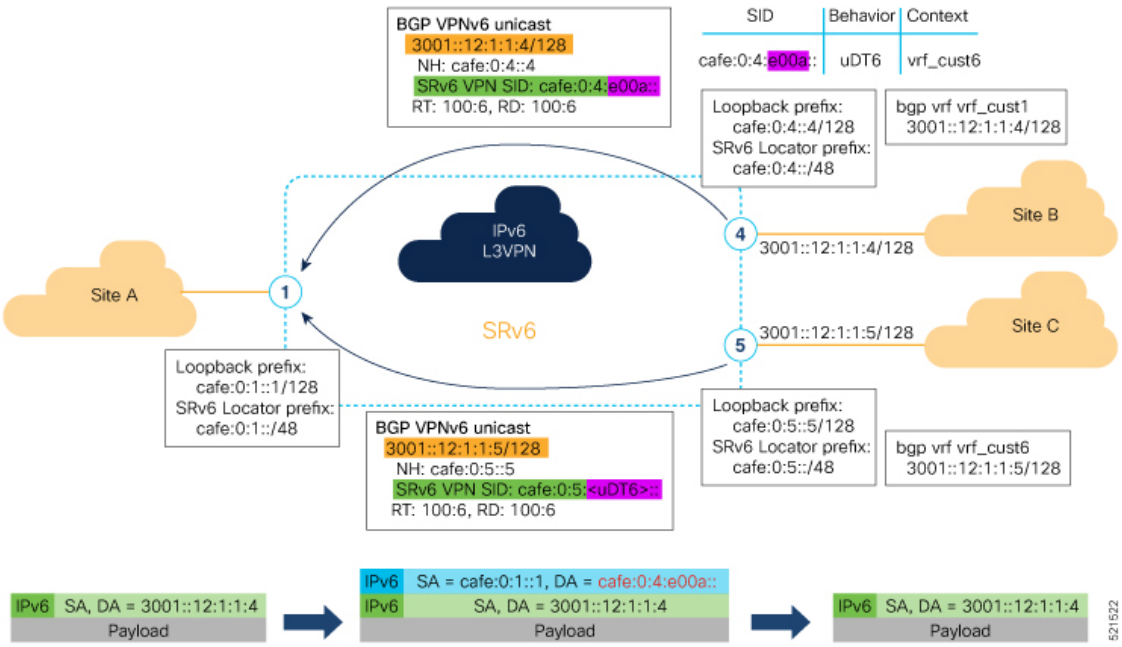
```
Node1(config-bgp-vrf-af)# exit
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-vrf-af-srv6)# commit
```

Running Configuration

```
router bgp 200
 address-family vpnv6 unicast
 !
 neighbor 3001::12:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv4 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
  address-family ipv6 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf-46
   !
  !
 !
!
end
```

Verification

The following figure shows a VPNv6 scenario. The sequence of commands included correspond to router Node1 acting as Ingress PE, and routers Node4 and Node5 acting as Egress PEs.



The following examples shows how to verify the SRv6 based L3VPN configurations for an Individual VRF with per VRF label allocation mode.

In this example, we can observe the uDT6 SID associated with the IPv6 L3VPN, where uDT6 behavior represents Endpoint with decapsulation and IPv6 table lookup.

```
Node1# show segment-routing srv6 sid
Fri Jan 29 19:31:53.293 UTC

*** Locator: 'Node1-locator' ***

SID                         Behavior          Context                          Owner
          State  RW
--------------------------  ----------------  -------------------------------
------------------  -----  --
cafe:0:1::                  uN (PSP/USD)      'default':1                      sidmgr
          InUse  Y
cafe:0:1:e000::             uA (PSP/USD)      [Hu0/0/0/0, Link-Local]:0        isis-1
          InUse  Y
cafe:0:1:e001::             uA (PSP/USD)      [Hu0/0/0/1, Link-Local]:0        isis-1
          InUse  Y
cafe:0:1:e002::             uDT4              'vrf_cust1'                      bgp-100
          InUse  Y
cafe:0:1:e003::             uDT4              'vrf_cust2'                      bgp-100
          InUse  Y
cafe:0:1:e004::             uDT4              'vrf_cust3'                      bgp-100
          InUse  Y
cafe:0:1:e005::             uDT4              'vrf_cust4'                      bgp-100
          InUse  Y
cafe:0:1:e006::             uDT4              'vrf_cust5'                      bgp-100
          InUse  Y
cafe:0:1:e007::             uA (PSP/USD)      [Hu0/0/0/0, Link-Local]:0:P      isis-1
          InUse  Y
cafe:0:1:e008::             uA (PSP/USD)      [Hu0/0/0/1, Link-Local]:0:P      isis-1
          InUse  Y
cafe:0:1:e009::             uDT6              'default'                        bgp-100
          InUse  Y
cafe:0:1:e00a::             uDT6              'vrf_cust6'                      bgp-100
```

```
          InUse  Y
```

The following examples show how to verify the SRv6 based L3VPN configuration using the **show bgp vpnv6 unicast** commands on the Ingress PE.

```
Node1# show bgp vpnv6 unicast summary
Fri Jan 29 19:33:01.177 UTC
BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 6
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               6          6          6          6           6           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
cafe:0:4::4       0   100     122     123        6    0    0 00:20:05          1
cafe:0:5::5       0   100     111     111        0    0    0 00:49:46          1

Node1# show bgp vpnv6 unicast rd 100:6
Fri Jan 29 19:41:01.334 UTC
BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 8
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::12:1:1:1/128 ::                       0         32768 ?
*>i3001::12:1:1:4/128 cafe:0:4::4              0    100      0 ?
*>i3001::12:1:1:5/128 cafe:0:5::5              0    100      0 ?

Processed 3 prefixes, 3 paths

Node1# show bgp vpnv6 unicast rd 100:6 3001::12:1:1:4/128
Fri Jan 29 19:41:42.008 UTC
BGP routing table entry for 3001::12:1:1:4/128, Route Distinguisher: 100:6
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                  6           6
Last Modified: Jan 29 19:29:35.858 for 00:12:06
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (received & used)
    cafe:0:4::4 (metric 30) from cafe:0:4::4 (1.1.1.4)
```

```
      Received Label 0xe00a00
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 6
      Extended community: RT:100:6
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:62, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv6 Unicast, Source VRF: vrf_cust6, Source Route Distinguisher: 100:6
```

The following examples show how to verify the BGP prefix information for VRF instances:

```
Node1# show bgp vrf vrf_cust6 ipv6 unicast
Fri Jan 29 19:42:05.675 UTC
BGP VRF vrf_cust6, state: Active
BGP Route Distinguisher: 100:6
VRF ID: 0x60000007
BGP router identifier 1.1.1.1, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800016   RD version: 8
BGP main routing table version 8
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::12:1:1:1/128 ::                        0         32768 ?
*>i3001::12:1:1:4/128 cafe:0:4::4               0    100      0 ?
*>i3001::12:1:1:5/128 cafe:0:5::5               0    100      0 ?

Processed 3 prefixes, 3 paths

Node1# show bgp vrf vrf_cust6 ipv6 unicast 3001::12:1:1:4/128

BGP routing table entry for 3001::12:1:1:4/128, Route Distinguisher: 100:6
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 17          17
Last Modified: Jan 15 16:50:44.032 for 01:48:21
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (received & used)
    cafe:0:4::4 (metric 30) from cafe:0:4::4 (1.1.1.4)
      Received Label 0xe00a00
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 17
      Extended community: RT:100:6
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:62, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv6 Unicast, Source VRF: vrf_cust6, Source Route Distinguisher: 100:6
```

The following examples show how to verify the current routes in the Routing Information Base (RIB):

```
Node1# show route vrf vrf_cust6 ipv6 unicast
Fri Jan 29 19:43:28.067 UTC

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is not set

L    3001::12:1:1:1/128 is directly connected,
      01:01:23, Loopback105
B    3001::12:1:1:4/128
      [200/0] via cafe:0:4::4 (nexthop in vrf default), 00:13:52
B    3001::12:1:1:5/128
      [200/0] via cafe:0:5::5 (nexthop in vrf default), 00:05:53

Node1# show route vrf vrf_cust6 ipv6 unicast 3001::12:1:1:4/128
Fri Jan 29 19:43:55.645 UTC

Routing entry for 3001::12:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 29 19:29:35.696 for 00:14:20
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
  No advertising protos.

Node1# show route vrf vrf_cust6 ipv6 unicast 3001::12:1:1:4/128 detail
Fri Jan 29 19:44:17.914 UTC

Routing entry for 3001::12:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 29 19:29:35.696 for 00:14:42
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Source RD attributes: 0x0000:100:6
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [f3216], SID-list {cafe:0:4:e00a::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_REMOTE
  Download Priority 3, Download Version 3
  No advertising protos.
```

The following examples show how to verify the current IPv6 Cisco Express Forwarding (CEF) table:

```
Node1# show cef vrf vrf_cust6 ipv6
Fri Jan 29 19:44:56.888 UTC

::/0
  drop       default handler
3001::12:1:1:1/128
  receive    Loopback105
3001::12:1:1:4/128
  recursive    cafe:0:4::/128
3001::12:1:1:5/128
  recursive    cafe:0:5::/128
fe80::/10
  receive
ff02::/16
  receive
ff02::2/128
  receive
ff02::1:ff00:0/104
  receive
ff05::/16
  receive
ff12::/16
  receive

Node1# show cef vrf vrf_cust6 ipv6 3001::12:1:1:4/128
Fri Jan 29 19:45:23.607 UTC
3001::12:1:1:4/128, version 3, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x78f2e0e0) [1],
 0x0 (0x0), 0x0 (0x888a3ac8)
 Updated Jan 29 19:29:35.700
 Prefix Len 128, traffic index 0, precedence n/a, priority 3
   via cafe:0:4::/128, 7 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78cd2a14 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:4::/128 via cafe:0:4::/48
    SRv6 H.Encaps.Red SID-list {cafe:0:4:e00a::}

Node1# show cef vrf vrf_cust6 ipv6 3001::12:1:1:4/128 detail
Fri Jan 29 19:45:55.847 UTC
3001::12:1:1:4/128, version 3, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x78f2e0e0) [1],
 0x0 (0x0), 0x0 (0x888a3ac8)
 Updated Jan 29 19:29:35.700
 Prefix Len 128, traffic index 0, precedence n/a, priority 3
  gateway array (0x78afe238) reference count 1, flags 0x2010, source rib (7), 0 backups
               [1 type 3 flags 0x48441 (0x78ba9a60) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Jan 29 19:29:35.699
 LDI Update time Jan 29 19:29:35.701

  Level 1 - Load distribution: 0
  [0] via cafe:0:4::/128, recursive

   via cafe:0:4::/128, 7 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78cd2a14 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:4::/128 via cafe:0:4::/48
    SRv6 H.Encaps.Red SID-list {cafe:0:4:e00a::}

    Load distribution: 0 1 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/0        remote
```

```
  1      Y    HundredGigE0/0/0/1         remote
```

SRv6 Services: IPv4 BGP Global

This feature extends support of SRv6-based BGP services to include IPv4 global BGP by implementing uDX4, uDT4, and uDT46 SRv6 functions at the PE node (draft-ietf-bess-srv6-services).

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv4 unicast address family
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to theImplementing BGP chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*
- Dual-Stack L3 Services (IPv4 BGP global, IPv6 BGP global) are supported.

BGP Global IPv4 Over SRv6 with Per-CE SID Allocation Mode (End.DX4)

The following example shows how to configure BGP global IPv4 over SRv6 with per-CE SID allocation.

```
router bgp 1
 bgp router-id 10.1.0.1
 address-family ipv4 unicast
  segment-routing srv6
   alloc mode per-ce
  !
 !
 neighbor 60::2
  remote-as 1
  update-source Loopback1
  address-family ipv4 unicast
   route-policy passall in
   encapsulation-type srv6
   route-policy passall out
  !
 !
 neighbor 52.52.52.1
  remote-as 3
  address-family ipv4 unicast
   route-policy passall in
   route-policy passall out
  !
 !
!
```

BGP Global IPv4 Over SRv6 with Per-AFI SID Allocation Mode (uDT4/uDT46)

To configure BGP global IPv4 over SRv6, use the following commands:

- **router bgp** *as-number* **address-family ipv4 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family ipv4 unicast segment-routing srv6 alloc mode** {**per-vrf** | **per-vrf-46** | **route-policy** *policy_name*}: Specify the SID behavior (allocation mode).

- **per-vrf**: Specifies that the same label is be used for all the routes advertised from a unique VRF.
- **per-vrf-46**: Specifies that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF. See Per-VRF-46 Allocation Mode.
- **route-policy** *policy_name*: Uses a route policy to determine the SID allocation mode and locator (if provided) for given prefix.

- **router bgp** *as-number* **address-family ipv4 unicast segment-routing srv6 locator** *WORD*: Specify the locator
- **router bgp** *as-number* {**af-group** *WORD*| **neighbor-group** *WORD* | **neighbor** *ipv6-addr*} **address-family ipv4 unicast encapsulation-type srv6**: Specify the encapuslation type for SRv6.
  - Use **af-group** *WORD* to apply the SRv6 encapsulation type to the address family group for BGP neighbors.
  - Use **neighbor-group** *WORD*to apply the SRv6 encapsulation type to the neighbor group for BGP neighbors.
  - Use **neighbor** *ipv6-addr* to apply the SRv6 encapsulation type to the specific BGP neighbor.

Use Case 1: BGP Global IPv4 over SRv6 with Per-AFI SID Allocation

The following example shows how to configure BGP global IPv4 over SRv6 with per-AFI SID allocation.

```
Node1(config)# router bgp 1
Node1(config-bgp)# bgp router-id 10.1.0.1
Node1(config-bgp)# address-family ipv4 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 60::2
Node1(config-bgp-nbr)# remote-as 1
Node1(config-bgp-nbr)# update-source Loopback1
Node1(config-bgp-nbr)# address-family ipv4 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# neighbor 52.52.52.1
Node1(config-bgp-nbr)# remote-as 3
Node1(config-bgp-nbr)# address-family ipv4 unicast
Node1(config-bgp-nbr-af)# route-policy passall in
Node1(config-bgp-nbr-af)# route-policy passall out
Node1(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 1
 bgp router-id 10.1.0.1
 address-family ipv4 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf
  !
 !
 neighbor 60::2
  remote-as 1
```

```
  update-source Loopback1
  address-family ipv4 unicast
   encapsulation-type srv6
  !
 !
 neighbor 52.52.52.1
  remote-as 3
  address-family ipv4 unicast
   route-policy passall in
   route-policy passall out
  !
 !
!
```

The following example shows how to configure BGP global IPv4/IPv6 over SRv6 with uDT46 SID allocation using per-VRF-46 alloction mode (uDT46 behavior).

```
Node1(config)# router bgp 1
Node1(config-bgp)# bgp router-id 10.1.0.1
Node1(config-bgp)# address-family ipv4 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 60::2
Node1(config-bgp-nbr)# remote-as 1
Node1(config-bgp-nbr)# update-source Loopback1
Node1(config-bgp-nbr)# address-family ipv4 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# neighbor 52.52.52.1
Node1(config-bgp-nbr)# remote-as 3
Node1(config-bgp-nbr)# address-family ipv4 unicast
Node1(config-bgp-nbr-af)# route-policy passall in
Node1(config-bgp-nbr-af)# route-policy passall out
Node1(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 1
 bgp router-id 10.1.0.1
 address-family ipv4 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf-46
  !
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf-46
  !
 !
 neighbor 60::2
```

```
  remote-as 1
  update-source Loopback1
  address-family ipv4 unicast
   encapsulation-type srv6
  !
 !
 neighbor 52.52.52.1
  remote-as 3
  address-family ipv4 unicast
   route-policy passall in
   route-policy passall out
  !
 !
!
```

Use Case 2: BGP Global IPv4 over SRv6 with Per-Prefix SID Allocation

This use case provides the ability to assign a specific SRv6 locator for a given prefix or a set of prefixes. The egress PE advertises the prefix with the specified locator. This allows for per-prefix steering into desired transport behaviors, such as Flex Algo.

To assign an SRv6 locator for a specific prefix, configure a route policy to specify the SID allocation mode based on match criteria. Examples of match criteria are destination-based match or community-based match.

- Supported SID allocation modes are per-VRF and per-CE.
- For per-VRF allocation mode, you can also specify the SRv6 locator.
  - If an SRv6 locator is specified in the route policy, BGP will use that to allocate per-VRF SID. If the specified locator is invalid, the SID will not be allocated.
  - If an SRv6 locator is not specified in the route policy, the default locator is used to allocate the SID. If the default locator is not configured in BGP, then the SID will not be allocated.
- Per-CE allocation mode always uses the default locator to allocate the SID.

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

The following example shows a route policy specifying the SID allocation mode with destination-based match:

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (1.1.1.0/24) then
Node1(config-rpl-if)# set srv6-alloc-mode per-vrf locator locator1
Node1(config-rpl-if)# elseif destination in (2.2.2.0/24) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf locator locator2
Node1(config-rpl-elseif)# elseif destination in (3.3.3.0/24) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf
Node1(config-rpl-elseif)# elseif destination in (4.4.4.0/24) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-ce
Node1(config-rpl-elseif)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
Node1(config)#
```

The following example shows how to configure BGP global IPv4 over SRv6 with a route policy to determine the SID allocation mode for given prefix.

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family ipv4 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# alloc mode route-policy set_per_prefix_locator_rpl
```

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (1.1.1.0/24) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (2.2.2.0/24) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3.3.3.0/24) then
    set srv6-alloc-mode per-vrf
  elseif destination in (4.4.4.0/24) then
    set srv6-alloc-mode per-ce
  else
    drop
  endif
end-policy
!
router bgp 100
 address-family ipv4 unicast
  segment-routing srv6
   alloc mode route-policy set_per_prefix_locator_rpl
  !
 !
!
```

Verify that the local and received SIDs have been correctly allocated under BGP IPv4 address family:

```
Node1# show bgp ipv4 unicast local-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Local Sid                                Alloc mode   Locator
*> 1.1.1.0/24         fc00:8:1:41::                            per-vrf      locator2
*> 2.2.2.0/24         fc00:0:1:41::                            per-vrf      locator1
*> 3.3.3.0/24         fc00:9:1:42::                            per-vrf      locator4
*> 4.4.4.0/24         fc00:9:1:43::                            per-ce       locator4
*> 1.1.1.5/32         NO SRv6 Sid                              -            -
* i8.8.8.8/32         NO SRv6 Sid                              -            -


Node1# show bgp ipv4 unicast received-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop                            Received Sid
*> 1.1.1.0/24         66.2.2.2                            NO SRv6 Sid
*> 2.2.2.0/24         66.2.2.2                            NO SRv6 Sid
*> 3.3.3.0/24         66.2.2.2                            NO SRv6 Sid
*> 4.4.4.0/24         66.2.2.2                            NO SRv6 Sid
*> 1.1.1.5/32         66.2.2.2                            NO SRv6 Sid
* i8.8.8.8/32         77.1.1.2                            fc00:0:2:41::
```

SRv6 Services: IPv6 BGP Global

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: BGP Global IPv6	Release 7.8.1	With this feature, the egress PE can signal an SRv6 Service SID with the BGP global route. The ingress PE encapsulates the IPv4/IPv6 payload in an outer IPv6 header where the destination address is the SRv6 Service SID provided by the egress PE. BGP messages between PEs carry SRv6 Service SIDs to interconnect PEs.

This feature extends support of SRv6-based BGP services to include IPv6 global BGP by implementing uDT6 and uDT46 SRv6 functions at the PE node (draft-ietf-bess-srv6-services).

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv6 unicast address family
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.
- Dual-Stack L3 Services (IPv4 BGP global, IPv6 BGP global) are supported.

BGP Global IPv6 Over SRv6 with Per-AFI SID Allocation Mode (uDT6)

To configure BGP global IPv6 over SRv6, use the following commands:

- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 alloc mode** {**per-vrf** | **per-vrf-46** | **route-policy** *policy_name*}: Specify the SID behavior (allocation mode).
  - **per-vrf**: Specifies that the same label is be used for all the routes advertised from a unique VRF.
  - **per-vrf-46**: Specifies that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF. See Per-VRF-46 Allocation Mode.
  - **route-policy** *policy_name*: Uses a route policy to determine the SID allocation mode and locator (if provided) for given prefix.
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 locator** *WORD*: Specify the locator
- **router bgp** *as-number* {**af-group** *WORD*| **neighbor-group** *WORD* | **neighbor** *ipv6-addr*} **address-family ipv6 unicast encapsulation-type srv6**: Specify the encapuslation type for SRv6.
  - Use **af-group** *WORD* to apply the SRv6 encapsulation type to the address family group for BGP neighbors.

- Use **neighbor-group** *WORD*to apply the SRv6 encapsulation type to the neighbor group for Border Gateway Protocol (BGP) neighbors.
- Use **neighbor** *ipv6-addr* to apply the SRv6 encapsulation type to the specific BGP neighbor.

Use Case 1: BGP Global IPv6 over SRv6 with Per-AFI SID Allocation

The following example shows how to configure BGP global IPv6 over SRv6 with per-AFI SID allocation.

```
Node1(config)# router bgp 100
Node1(config-bgp)# bgp router-id 1.1.1.1
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor cafe:0:4::4
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# neighbor cafe:0:5::5
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 100
 bgp router-id 1.1.1.1
 segment-routing srv6
  locator Node1
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf
  !
 !
 neighbor cafe:0:4::4
  address-family ipv6 unicast
   encapsulation-type srv6
  !
 !
 neighbor cafe:0:5::5
  address-family ipv6 unicast
   encapsulation-type srv6
```

The following example shows how to configure BGP global IPv4/IPv6 over SRv6 with uDT46 SID allocation using per-VRF-46 allocation mode (uDT46 behavior).

```
Node1(config)# router bgp 200
Node1(config-bgp)# bgp router-id 1.1.1.1
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family ipv4 unicast
```

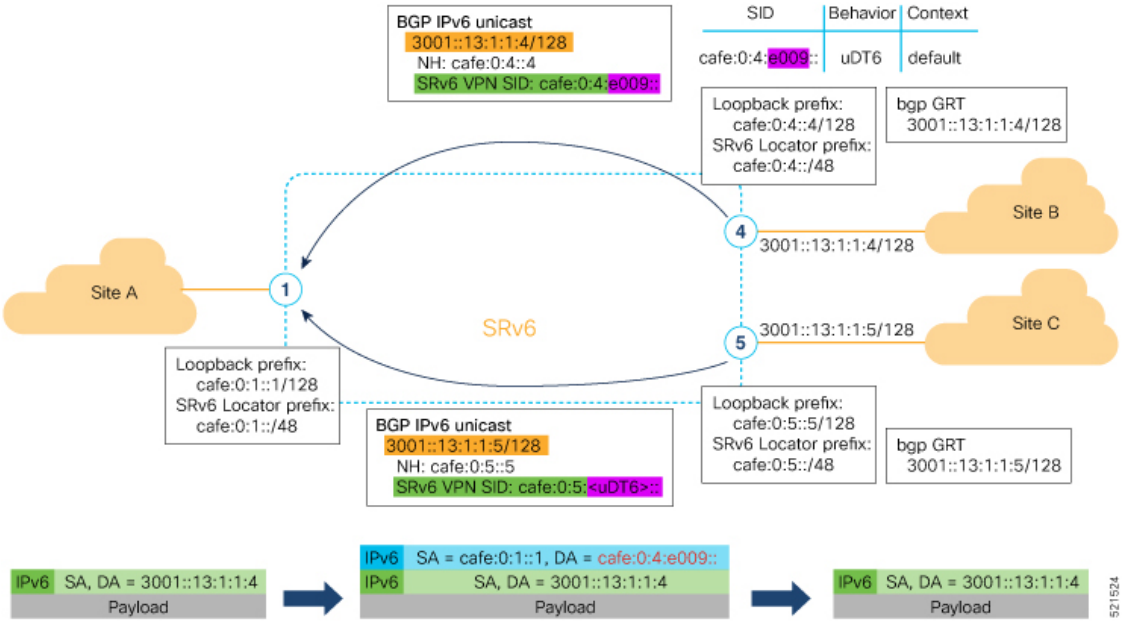
```
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf-46
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor cafe:0:4::4
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# neighbor cafe:0:5::5
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 200
 bgp router-id 1.1.1.1
 segment-routing srv6
  locator Node1
 !
 address-family ipv4 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf-46
  !
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf-46
  !
 !
 neighbor cafe:0:4::4
  address-family ipv6 unicast
   encapsulation-type srv6
  !
 !
 neighbor cafe:0:5::5
  address-family ipv6 unicast
   encapsulation-type srv6
```

Verification

The following figure shows a IPv6 BGP global scenario. The sequence of commands included correspond to router Node1 acting as Ingress PE, and routers Node4 and Node5 acting as Egress PEs.



The following examples show how to verify the BGP global IPv6 configuration using the **show bgp ipv6 unicast** commands.

```
Node1# show bgp ipv6 unicast summary
Fri Jan 29 19:48:23.255 UTC
BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               4          4          4          4           4           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
cafe:0:4::4       0   100     137     138        4    0    0 00:35:27           1
cafe:0:5::5       0   100     138     137        4    0    0 00:10:54           1

Node1# show bgp ipv6 unicast
Fri Jan 29 19:49:05.688 UTC
BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 3001::13:1:1:1/128 ::                       0         32768 i
*>i3001::13:1:1:4/128 cafe:0:4::4              0    100      0 i
*>i3001::13:1:1:5/128 cafe:0:5::5              0    100      0 i

Processed 3 prefixes, 3 paths

Node1# show bgp ipv6 unicast 3001::13:1:1:4/128
Fri Jan 29 19:49:22.067 UTC
BGP routing table entry for 3001::13:1:1:4/128
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 3           3
Last Modified: Jan 29 19:14:13.858 for 00:35:08
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    cafe:0:4::4 (metric 30) from cafe:0:4::4 (1.1.1.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 3
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4:e009::, Behavior:62, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
```

The following examples show how to verify the current routes in the Routing Information Base (RIB):

```
Node1# show route ipv6 3001::13:1:1:4/128
Fri Jan 29 19:53:26.839 UTC

Routing entry for 3001::13:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 29 19:14:13.397 for 00:35:28
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Route metric is 0
  No advertising protos.

Node1# show route ipv6 3001::13:1:1:4/128 detail
Fri Jan 29 19:50:08.601 UTC

Routing entry for 3001::13:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 29 19:14:13.397 for 00:35:55
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [f3216], SID-list {cafe:0:4:e009::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
```

```
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 4, Download Version 106
  No advertising protos.
```

The following examples show how to verify the current IPv6 Cisco Express Forwarding (CEF) table:

```
Node1# show cef ipv6 3001::13:1:1:4/128
Fri Jan 29 19:50:29.149 UTC
3001::13:1:1:4/128, version 106, SRv6 Headend, internal 0x5000001 0x40 (ptr 0x78    cd3944)
 [1], 0x0 (0x0), 0x0 (0x888a3a80)
 Updated Jan 29 19:14:13.401
 Prefix Len 128, traffic index 0, precedence n/a, priority 4
   via cafe:0:4::/128, 7 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78cd2a14 0x0]
    next hop cafe:0:4::/128 via cafe:0:4::/48
    SRv6 H.Encaps.Red SID-list {cafe:0:4:e009::}

Node1# show cef ipv6 3001::13:1:1:4/128 detail
Fri Jan 29 19:51:00.920 UTC
3001::13:1:1:4/128, version 106, SRv6 Headend, internal 0x5000001 0x40 (ptr 0x78cd3944)
[1], 0x0 (0x0), 0x0 (0x888a3a80)
 Updated Jan 29 19:14:13.401
 Prefix Len 128, traffic index 0, precedence n/a, priority 4
  gateway array (0x78afe150) reference count 1, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x78ba99e8) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Jan 29 19:14:13.401
 LDI Update time Jan 29 19:14:13.401

  Level 1 - Load distribution: 0
  [0] via cafe:0:4::/128, recursive

   via cafe:0:4::/128, 7 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78cd2a14 0x0]
    next hop cafe:0:4::/128 via cafe:0:4::/48
    SRv6 H.Encaps.Red SID-list {cafe:0:4:e009::}

    Load distribution: 0 1 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/0        remote
    1     Y   HundredGigE0/0/0/1        remote
```

BGP Signaling for co-existence of IP routes with or without SRv6 SID

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
BGP Signaling for co-existence of IP routes	Release 24.3.1	Introduced in this release on: Fixed Systems (8200); Modular Systems (8800 [LC ASIC: Q100, Q200]) SRv6 with BGP supports the coexistence of IP routes with or without SRv6 SID over an SRv6-enabled core network. This support enables integrating SRv6 capabilities into existing network infrastructures without replacing IP routing completely. This feature enables flexibility and scalability, transition to new technologies, and enhanced network efficiency, making it easier to migrate from MPLS to SRV6. The feature introduces these changes: CLI: • **encapsulation-type srv6 relax-sid**

Need for BGP Signaling Over SRv6 core

BGP now supports sending internet service over an SRv6 core, assuming that all Global Routing Table (GRT) routes are advertised with an SRv6-SID.

To differentiate between the SRv6 core and non-SRv6 core sides, an "encapsulation-type SRv6" was introduced under the IPv6 BGP peer for the IPv4 unicast address-family. When the "encapsulation-type srv6" is enabled, routes without an SRv6-SID are not sent to the neighbor sessions during update generation. For more information, see Configuring SRv6 BGP-Based Services and https://datatracker.ietf.org/doc/rfc9252/.

However, in some networks, there may be a mix of GRT routes with SRv6 SID encapsulation and without SRv6 encapsulation. Hence, there is a need for BGP to allow SRv6-enabled GRT to support the co-existence and signaling of IP routes with or without an SRv6-SID on the same IPv6 neighbor session.

Co-existence of IP routes with or without SRv6 SID

This feature adds a new BGP encapsulation type called **SRv6 relax-SID**, which allows the advertisement of prefixes with or without SRv6 SID over the same BGP session. This is in contrast to the existing encapsulation type "srv6", which did not advertise prefixes without an SRv6 SID. The configuration allows for the specification of route policies that set the SRv6 allocation mode based on the destination prefix, enabling the coexistence of IP routes with or without SRv6 SID.

Benefits

The benefits of the co-existence of IP routes with or without SRv6 SID over an SRv6 core are numerous and significant for network operations as listed.

- **Enhanced Network Efficiency**: Allows seamless integration of SRv6 capabilities into existing network infrastructures, which can lead to more efficient routing and resource utilization.

- **Simplified Operations**: By supporting the coexistence of IP routes with or without SRv6 SID, network operators can manage their networks better without maintaining separate BGP peer sessions to support advertising both type of routes.
- **Future-Proofing the Network**: As networks evolve, the ability to support IP routes with or without SRv6 SID ensures that the network is prepared to enable customer to support use cases such as overlay and underlay route separation in a GRT table.
- **Cost Savings**: Reduce operations cost by streamlining network efficiency by optimizing BGP session management.
- **Flexibility and Scalability**: The feature provides the flexibility to apply SRv6 where it is needed while maintaining IP routing, allowing the network to scale efficiently.
- **Transition to New Technologies**: It facilitates a smoother transition to newer routing technologies like SRv6, which is designed to meet the demands of modern network applications and services.

These benefits contribute to a more robust, agile, and cost-effective network that can adapt to the changing needs of service providers and their customers.

Configure BGP Signaling over SRv6 Core

The purpose of this task is to enable SRv6 with BGP to support the co-existence of IP routes with or without SRv6 SID.

Follow these steps to configure BGP signaling over SRv6 Core.

Procedure

Step 1 Execute the **encapsulation-type srv6 relax-sid** command on neighbor to configure the neighbor.

Summary of this configuration: Set up BGP to use SRv6 for IPv4 unicast routes, with specific rules for SID allocation based on the destination prefixes. It also configures a BGP neighbor and specifies how SRv6 encapsulation should be handled for that neighbor.

Example:

```
Router(config)# route-policy alloc-sid-policy
Router(config-rpl)# if destination in prefix-set-1 then
Router(config-rpl-if)# set srv6-alloc-mode per-vrf locator LOC2
Router(config-rpl-if)# else if destination is prefix-set-2 then
Router(config-rpl-else)# drop
Router(config-rpl-if)# else
Router(config-rpl-else)# set srv6-alloc-mode per-vrf
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy

Router(config)# router bgp 2
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# segment-routing srv6
Router(config-bgp-af-srv6)# locator LOC1
Router(config-bgp-af-srv6)# alloc mode route-policy alloc-sid-policy
Router(config-bgp-af-srv6)# exit
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 12:100::1
Router(config-bgp-nbr)# address-family ipv4 unicast
```

```
Router(config-bgp-nbr-af)# encapsulation-type srv6 relax-sid
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
```

Step 2 Execute the **encapsulation-type srv6 relax-sid** command on the neighbor group to configure the neighbor-group.

Example:

```
Router(config-bgp)# neighbor-group srv6-core-relax
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# encapsulation-type srv6 relax-sid
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# neighbor 12:100::1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use neighbor-group srv6-core-relax
Router(config-bgp-nbr)# exit
```

Step 3 Execute the **encapsulation-type srv6 relax-sid** command, on the address family group to configure the Address- Family Group .

Example:

```
Router(config-bgp)# af-group srv6-core-af address-family ipv4 unicast
Router(config-bgp-nbr-af)# encapsulation-type srv6 relax-sid
Router(config-bgp-nbr)# exit
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# neighbor 12:100::1
Router(config-bgp-nbr-af))# remote-as 1
Router(config-bgp-nbr-af))# address-family ipv4 unicast
Router(config-bgp-nbr-af))# use af-group srv6-core-af
Router(config-bgp-nbr)# exit
```

Step 4 Run the show commands to verify the encapsulation type is updated to SRv6 Relax-SID in all neighbor sessions.

You can see that 192::4 has **encapsulation-type srv6 relax-sid** configured.

Example:

```
Router#show bgp neighbor 192::4
For Address Family: IPv4 Unicast
  BGP neighbor version 155
  Update group: 0.1 Filter-group: 0.3  No Refresh request being processed
  Encapsulation type SRv6 Relax-SID
  NEXT_HOP is always this router
  Default information originate: default sent
  AF-dependent capabilities:
    Graceful Restart capability advertised
      Local restart time is 120, RIB purge time is 600 seconds
      Maximum stalepath time is 360 seconds
    Graceful Restart capability received
      Remote Restart time is 120 seconds
      Neighbor preserved the forwarding state during latest restart
    Extended Nexthop Encoding: advertised and received
  Route refresh request: received 0, sent 0
  3 accepted prefixes, 3 are bestpaths
….

Router#show bgp update-group neighbor 192::4
Update group for IPv4 Unicast, index 0.1:
  Attributes:
```

```
    Neighbor sessions are IPv6
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    Next-hop-self enabled
    4-byte AS capable
    Advertise routes with local-label via Unicast SAFI
    Send AIGP
    Encapsulation type SRv6 Relax-SID
    Send multicast attributes
    Extended Nexthop Encoding
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 7, replicated: 7
  All neighbor are assigned to sub-group(s)
    Neighbors in sub-group: 0.3, Filter-Groups num:1
     Neighbors in filter-group: 0.3(RT num: 0)
      192::4
```

In the following example, 158.158.58.1/32 is without SRv6 SID but advertised to 192::4 and 157.157.57.1/32 with SRv6 SID, which is also advertised to 192::4. To allow IP route without SRv6 SID, you must include it in **prefix-set-2**.

Example:

```
Router#show bgp 158.158.58.1/32
BGP routing table entry for 158.158.58.1/32
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                175          175
Last Modified: Dec 13 11:38:31.000 for 00:00:04
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  60
    16.16.16.3 from 16.16.16.3 (16.16.16.3)
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 175
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Not advertised to any peer
  70
    17.17.17.3 from 17.17.17.3 (17.17.17.3)
      Origin IGP, localpref 100, valid, external, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)

Note that both Prefix 157 with SID and Prefix 158 without SID are advertised to neighbor 192::4.


Router#show bgp 157.157.57.1/32
BGP routing table entry for 157.157.57.1/32
Versions:
```

```
  Process           bRIB/RIB   SendTblVer
  Speaker                172          172
    SRv6-VPN SID: cafe:1:1:2:42::/128
    Format: base
Last Modified: Dec 13 11:38:31.000 for 00:02:09
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  50
    15.15.15.3 from 15.15.15.3 (15.15.15.3)
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 172
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Not advertised to any peer
  60
    16.16.16.3 from 16.16.16.3 (16.16.16.3)
      Origin IGP, localpref 100, valid, external, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)
```

Step 5 Run these commands to view the flag details and path-elements, if needed.

Example:

```
Router#show bgp 157.157.57.1/32 detail
BGP routing table entry for 157.157.57.1/32
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                172          172
    SRv6-VPN SID: cafe:1:1:2:42::/128
    Format: base
    Alloc Mode/Locator ID: per-vrf/2
    Flags: 0x00123201+0x61010000+0x00000000; multipath;
Last Modified: Dec 13 11:38:31.000 for 00:04:22
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Path #1: Received by speaker 0
  Flags: 0x3000000001050003+0x00, import: 0x020
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  50
    15.15.15.3 from 15.15.15.3 (15.15.15.3), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 172
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Flags: 0x3000000000010003+0x00, import: 0x020
  Not advertised to any peer
  60
    16.16.16.3 from 16.16.16.3 (16.16.16.3), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, multipath
```

```
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)


Router#show bgp 158.158.58.1/32 path-elements
BGP routing table entry for 158.158.58.1/32
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker               175          175
    Flags: 0x00123201+0x20010000+0x00000002; multipath;
Last Modified: Dec 13 11:38:31.000 for 00:05:50
Paths: (2 available, best #1)
Path count: 2
Path-elements: 1
  Path ID: 1
    Gateway metric 0, Version 175
    Path: Nexthop 16.16.16.3, flags 0x3000000001050003
          Neighbor 16.16.16.3, Received Path ID 0
    Flags: 0x00000001
           status: valid
           path type: bestpath
           add-path action:
    Opaque: pelem=0x7f7948026d88
            net=0x7f794d2fd968,       tblattr=0x22cc208 (ver 177)
            path=0x7f794d2dd0c8, path-tblattr=0x22cc208 (ver 177)
                       nobestpath-tblattr=0x22cd6c0 (ver 0)
                        noaddpath-tblattr=0x22cd638 (ver 0)
            bitfields=0x7f79481ce538 (val=0xc, size=1)
            pe-bitfields=0x0 (val=0x0, size=0)
            orr-bitfields=0x0 (val=0x0, size=0)
            orr-ap-bitfields=0x0 (val=0x0, size=0)
            net-next=0x0, tblattr-prev=0x7f7948026d18, tblattr-next=0x0
    Radix: rn_parent=0x7f794d2fdd88, rn_left=0x7f794d2fdf98, rn_right=0x7f794d2fd758,
           rn_version=180, rn_bit=6, rn_flags=0x0
Active Paths: (0 available)
Active Path-elements: 0
```

SRv6 Services: IPv4 L3VPN Active-Standby Redundancy using Port-Active Mode

The Segment Routing IPv6 (SRv6) Services: IPv4 L3VPN Active-Standby Redundancy using Port-Active Mode feature provides all-active per-port load balancing for multihoming. The forwarding of traffic is determined based on a specific interface rather than per-flow across multiple Provider Edge routers. This feature enables efficient load-balancing and provides faster convergence. In an active-standby scenario, the active PE router is detected using designated forwarder (DF) election by modulo calculation and the interface of the standby PE router brought down. For Modulo calculation, byte 10 of the Ethernet Segment Identifier (ESI) is used.

Usage Guidelines and Restrictions

- This feature can only be configured for bundle interfaces.
- When an EVPN Ethernet Segment (ES) is configured with port-active load-balancing mode, you cannot configure ACs of that bundle on bridge-domains with a configured EVPN instance (EVI). EVPN Layer 2 bridging service is not compatible with port-active load-balancing.

SRv6 Services for L3VPN Active-Standby Redundancy using Port-Active Mode: Operation

Under port-active operational mode, EVPN Ethernet Segment (ES) routes are exchanged across BGP for the routers servicing the multihomed ES. Each PE router then builds an ordered list of the IP addresses of all PEs connected to the ES, including itself, and assigns itself an ordinal for its position in the list. The ordinals are used with the modulo calculation to determine which PE will be the Designated Forwarder (DF) for a given ES. All non-DF PEs will take the respective bundles out of service.

In the case of link or port failure, the active DF PE withdraws its ES route. This re-triggers DF election for all PEs that service the ES and a new PE is elected as DF.

Configure SRv6 Services L3VPN Active-Standby Redundancy using Port-Active Mode

This section describes how you can configure SRv6 services L3VPN active-standby redundancy using port-active mode under an Ethernet Segment (ES).

Configuration Example

```
/* Configure Ethernet Link Bundles */
Router# configure
Router(config)# interface Bundle-Ether10
Router(config-if)# ipv4 address 10.0.0.2 255.255.255.0
Router(config-if)# ipv6 address 2001:DB8::1
Router(config-if)# lacp period short
Router(config-if)# mac-address 1.2.3
Router(config-if)# bundle wait-while 0
Router(config-if)# exit
Router(config)# interface GigabitEthernet 0/2/0/5
Router(config-if)# bundle id 14 mode active
Router(config-if)# commit

/* Configure load balancing. */
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 11.11.11.11.11.11.11.11.14
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# commit
!
/* Configure address family session in BGP. */
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 192.168.0.2
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 192.168.0.3
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr)# commit
```

Running Configuration

```
interface Bundle-Ether14
 ipv4 address 14.0.0.2 255.255.255.0
 ipv6 address 14::2/64
 lacp period short
 mac-address 1.2.3
```

```
 bundle wait-while 0
!
interface GigabitEthernet0/2/0/5
 bundle id 14 mode active
!
evpn
 interface Bundle-Ether14
  ethernet-segment
   identifier type 0 11.11.11.11.11.11.11.11.14
   load-balancing-mode port-active
  !
 !
!
router bgp 100
 bgp router-id 192.168.0.2
 address-family l2vpn evpn
 !
 neighbor 192.168.0.3
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
```

Verification

Verify the SRv6 services L3VPN active-standby redundancy using port-active mode configuration.

```
/* Verify ethernet-segment details on active DF router */
Router# show evpn ethernet-segment interface Bundle-Ether14 detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0011.1111.1111.1111.1114 BE14                              192.168.0.2
                                                           192.168.0.3
    ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether14
     Interface MAC  : 0001.0002.0003
     IfHandle       : 0x000041d0
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.1111.1114
  ES Import RT      : 1111.1111.1111 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Peering Details   :
     192.168.0.2 [MOD:P:00]
     192.168.0.3 [MOD:P:00]

  Service Carving Results:
     Forwarders     : 0
     Permanent      : 0
     Elected        : 0
     Not Elected    : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
```

```
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : None
  Remote SHG labels : 0


/* Verify bundle Ethernet configuration on active DF router */
Router# show bundle bundle-ether 14
Bundle-Ether14
  Status:                                  Up
  Local links <active/standby/configured>:   1 / 0 / 1
  Local bandwidth <effective/available>:     1000000 (1000000) kbps
  MAC address (source):                    0001.0002.0003 (Configured)
  Inter-chassis link:                      No
  Minimum active links / bandwidth:        1 / 1 kbps
  Maximum active links:                    64
  Wait while timer:                        Off
  Load balancing:
    Link order signaling:                  Not configured
    Hash type:                             Default
    Locality threshold:                    None
  LACP:                                    Operational
    Flap suppression timer:                Off
    Cisco extensions:                      Disabled
    Non-revertive:                         Disabled
  mLACP:                                   Not configured
  IPv4 BFD:                                Not configured
  IPv6 BFD:                                Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Gi0/2/0/5             Local            Active       0x8000, 0x0003     1000000
      Link is Active



/*  Verify ethernet-segment details on standby DF router */
Router# show evpn ethernet-segment interface bundle-ether 10 detail

Router# show evpn ethernet-segment interface Bundle-Ether24 detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0011.1111.1111.1111.1114 BE24                              192.168.0.2
                                                           192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether24
     Interface MAC  : 0001.0002.0003
     IfHandle       : 0x000041b0
     State          : Standby
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.1111.1114
  ES Import RT      : 1111.1111.1111 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Peering Details   :
     192.168.0.2 [MOD:P:00]
     192.168.0.3 [MOD:P:00]
```

```
  Service Carving Results:
     Forwarders     : 0
     Permanent      : 0
     Elected        : 0
     Not Elected    : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : None
  Remote SHG labels : 0

/* Verify bundle configuration on standby DF router */
Router# show bundle bundle-ether 24

Bundle-Ether24
  Status:                                  LACP OOS (out of service)
  Local links <active/standby/configured>:   0 / 1 / 1
  Local bandwidth <effective/available>:     0 (0) kbps
  MAC address (source):                    0001.0002.0003 (Configured)
  Inter-chassis link:                      No
  Minimum active links / bandwidth:        1 / 1 kbps
  Maximum active links:                    64
  Wait while timer:                        Off
  Load balancing:
    Link order signaling:                  Not configured
    Hash type:                             Default
    Locality threshold:                    None
  LACP:                                    Operational
    Flap suppression timer:                Off
    Cisco extensions:                      Disabled
    Non-revertive:                         Disabled
  mLACP:                                   Not configured
  IPv4 BFD:                                Not configured
  IPv6 BFD:                                Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Gi0/0/0/4             Local            Standby      0x8000, 0x0002     1000000
      Link is in standby due to bundle out of service state
```

SRv6 Services: IPv4 L3VPN Active-Active Redundancy

This feature provides active-active connectivity to a CE device in a L3VPN deployment. The CE device can be Layer-2 or Layer-3 device connecting to the redundant PEs over a single LACP LAG port.

Depending on the bundle hashing, an ARP or IPv6 Network Discovery (ND) packet can be sent to any of the redundant routers. As a result, not all entries will exist on a given PE. In order to provide complete awareness, Layer-3 local route learning is augmented with remote route-synchronization programming.

Route synchronization between service PEs is required in order to provide minimum interruption to unicast and multicast services after failure on a redundant service PE. The following EVPN route-types are used for Layer-3 route synchronization:

- EVPN route-type 2 for synchronizing ARP tables
- EVPN route-type 7/8 for synchronizing IGMP JOINS/LEAVES

In a Layer-3 CE scenario, the router that connects to the redundant PEs may establish an IGP adjacency on the bundle port. In this case, the adjacency will be formed to one of the redundant PEs, and IGP customer

routes will only be present on that PE. To synchronize Layer-3 customer subnet routes (IP Prefixes), the EVPN route-type 5 is used to carry the ESI and ETAG as well as the gateway address (prefix next-hop address).

Note Gratuitous ARP (GARP) or IPv6 Network Advertisement (NA) replay is not needed for CEs connected to the redundant PEs over a single LAG port.

The below configuration enables Layer-3 route synchronization for routes learned on the Ethernet-segment sub-interfaces.

```
evpn
 route-sync vrf default
!
vrf RED
 evi route-sync 10
!
vrf BLUE
 evi route-sync 20
!
```

Note EVPN does not support untagged interfaces.

SRv6 Services: L3 EVPN

EVPN Route Type 5 (RT5) is used for the advertisement of EVPN routes using IP prefixes (refer to IETF RFC 9136 - IP Prefix Advertisement in Ethernet VPN (EVPN)) to provide end-to-end L3 connectivity

This feature adds support for carrying L3VPN routes in L2VPN EVPN EVPN RT5 address family instead of VPNv4 unicast and/or VPNv6 unicast address-family across SRv6 core (EVPN over SRv6 underlay).

Usage Guidelines and Limitations

Interworking between EVPN RT5 over SRv6 core and EVPN RT5 over MPLS core is supported. See L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway, on page 102.

Interworking between EVPN RT5 over SRv6 core and L3VPN over MPLS core is supported. See L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway, on page 105.

BGP does not support dual VPNv4/v6 address family and EVPN RT5 address family on the same BGP session. For the route reflector (RR) to receive both Type-5 EVPN route and VPNv4/v6 address family, we recommend that you configure two pairs of loopback interfaces and configure two BGP loopback sessions between the RR and the PE: one session for VPNv4/v6 address family and one session for EVPN address family.

BGP sends all VRF routes via either VPNv4/v6 or EVPN address family. We recommend that you mark the VRF route via export route-policy and use neighbor out policy to either drop or pass the route for an address family to achieve the same net effect.

The following behaviors are supported:

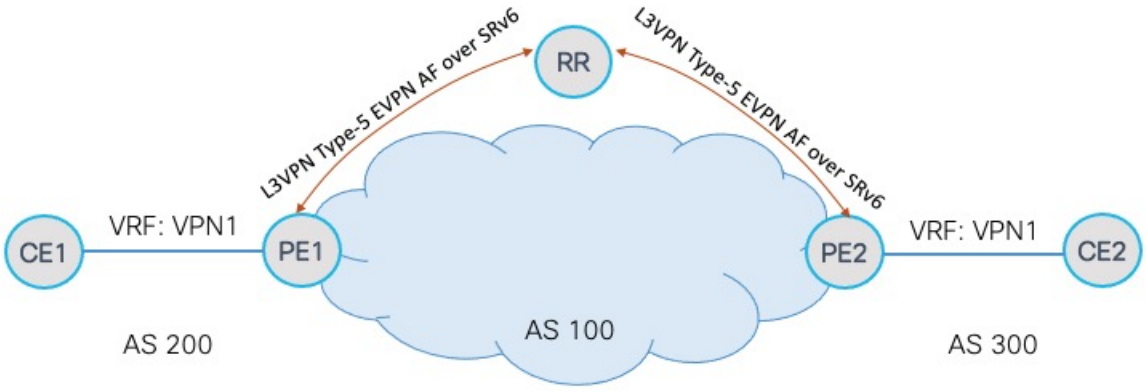
- IPv4, IPv6, and IPv4/IPv6 (dual stack) L3 EVPN over SRv6
- uDT4

- uDT6
- uDT46
- Automated Steering to Flex-Algo (BGP per-VRF locator Flex-Algo (per-prefix))
- Automated Steering to SRv6 Policy (ODN/AS)

Configuring SRv6-based L3 EVPN

To enable SRv6-based L3 EVPN, you must enable SRv6 under BGP, specify the locator, and configure the SID allocation mode. The assignment of the locator can be done in multiple ways under the **router bgp** configuration. See SRv6 Locator Inheritance Rules, on page 40

Figure 11: Configuration Example: Dual Stack L3 EVPN over SRv6



Configure the VRF (Dual-Stack IPv4/IPv6)

```
Router(config)# vrf VPN1
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 1:1
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export route-target
Router(config-vrf-export-rt)# 1:1
Router(config-vrf-export-rt)# exit
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# import route-target
Router(config-vrf-import-rt)# 1:1
Router(config-vrf-import-rt)# exit
Router(config-vrf-af)# export route-target
Router(config-vrf-export-rt)# 1:1
Router(config-vrf-export-rt)# exit
Router(config-vrf-af)#
```

Configure the SRv6 Locator for an Individual VRF, with Per-VRF Label Allocation Mode

To configure the SRv6 locator for a specific VRF under IPv4 or IPv6 Address Family and specify the allocation mode, use the following commands:

- **router bgp** *as-number* **vrf** *WORD* **address-family** {**ipv4** | **ipv6**} **unicast segment-routing srv6**: Enables SRv6

- **router bgp** *as-number* **vrf** *WORD* **address-family** {**ipv4** | **ipv6**} **unicast segment-routing srv6 alloc mode** {**per-vrf** | **per-vrf-46** }: Specifies the SID behavior (allocation mode).
  - **per-vrf**: Specifies that the same service SID (uDT6 behavior) is used for all the routes advertised from a unique VRF.
  - **per-vrf-46**: Specifies that the same service SID (uDT46 behavior) be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **vrf** *WORD* **address-family** {**ipv4** | **ipv6**} **unicast segment-routing srv6 locator** *WORD*: Specifies the locator

```
Router(config)# router bgp 100
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# additional-paths receive
Router(config-bgp-af)# additional-paths send
Router(config-bgp-af)# additional-paths selection route-policy add-path
Router(config-bgp-af)# exit
Router(config-bgp)# address-family vpnv6 unicast
Router(config-bgp-af)# additional-paths receive
Router(config-bgp-af)# additional-paths send
Router(config-bgp-af)# additional-paths selection route-policy add-path
Router(config-bgp-af)# exit
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# additional-paths receive
Router(config-bgp-af)# additional-paths send
Router(config-bgp-af)# additional-paths selection route-policy add-path
Router(config-bgp-af)# exit

Router(config-bgp)# neighbor 1111::1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# advertise vpnv4 unicast
Router(config-bgp-nbr-af)# advertise vpnv6 unicast
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit

Router(config-bgp)# vrf VPN1
Router(config-bgp-vrf)# rd 100:1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# segment-routing srv6
Router(config-bgp-vrf-af-srv6)# locator LOC1
Router(config-bgp-vrf-af-srv6)# alloc mode per-vrf
Router(config-bgp-vrf-af-srv6)# exit
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# segment-routing srv6
Router(config-bgp-vrf-af-srv6)# locator LOC1
Router(config-bgp-vrf-af-srv6)# alloc mode per-vrf-46
Router(config-bgp-vrf-af-srv6)# exit
Router(config-bgp-vrf-af)# exit

Router(config-bgp-vrf)# neighbor 1.1.1.1
Router(config-bgp-vrf-nbr)# remote-as 200
Router(config-bgp-vrf-nbr)# address-family ipv4 unicast
Router(config-bgp-vrf-nbr-af)# exit
Router(config-bgp-vrf-nbr)# exit
Router(config-bgp-vrf)# neighbor 3333::3
Router(config-bgp-vrf-nbr)# remote-as 200
Router(config-bgp-vrf-nbr)# address-family ipv6 unicast
```

Running Configuration

```
vrf VPN1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
router bgp 100
 address-family vpnv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy add-path
 !
 address-family vpnv6 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy add-path
 !
 address-family l2vpn evpn
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy add-path
 !
 neighbor 1111::1
  remote-as 100
  address-family l2vpn evpn
   advertise vpnv4 unicast
   advertise vpnv6 unicast
  !
 !
 vrf VPN1
  rd 100:1
  address-family ipv4 unicast
   segment-routing srv6
    locator LOC1
    alloc mode per-vrf
   !
  !
  address-family ipv6 unicast
   segment-routing srv6
    locator LOC1
    alloc mode per-vrf-46
   !
  !
  neighbor 1.1.1.1
   remote-as 200
   address-family ipv4 unicast
   !
  !
  neighbor 3333::3
   remote-as 200
```

```
   address-family ipv6 unicast
   !
  !
 !
!
```

SRv6 Services: L2 and L3 Services with Remote SIDs from W-LIB

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: L2 and L3 Services with Remote SIDs from Wide Local ID Block	Release 7.9.1	This feature enables an SRv6 headend node to receive and install remote SIDs with Wide (32-bit) functions (Remote W-LIB). The Remote W-LIB is supported for Layer 3 (VPN/BGP global) and Layer 2 EVPN services (ELINE/ELAN). This capability is enabled by default.

This capability is enabled by default; there is no CLI to configure this capability at the ingress PE.

An SRv6 Service SID is used to identify a specific service function. This Service SID inserted into the packet header by the source node is used to steer the packet along a specific path that includes the service function.

The Service SID signaled by transposing a variable part of the SRv6 SID value (function, argument, or both) and carrying them in the existing label fields to achieve more efficient compression of those service prefix NLRIs in BGP update messages. The SRv6 SID Structure Sub-Sub-TLV (SSTLV) contains appropriate length fields when the SRv6 Service SID is signaled in split parts to enable the receiver to put together the SID accurately.

The Transposition Offset indicates the bit position. The Transposition Length indicates the number of bits that are being taken out of the SRv6 SID value and put into high order bits of label field.

For example, a remote W-LIB uSID **fcbb:bb00:0200:fff0:0001::** with a SRv6 SID SSTLV of **BL=32; NL=16; FL=32; AL=0, TPOS len/offset=16/64** is defined as follows:

- Block length (BL) of 32 bits = fcbb:bb00
- Node length (NL) of 16 bits = 0200
- Function length (FL) of 32 bits = fff0:0001
- Argument length (AL) of 0
- Transposition length (TPOS len) of 16 bits = 0001
- Transposition offset (TPOS offset) of 64 bits = fcbb:bb00:0200:fff0:

This results in a SID value of **fcbb:bb00:0200:fff0::** and Label value of **0x0001**.

Example

The following example shows output of a BGP route table for a VPNv4 prefix learned from three egress PEs:

- BGP Path 1 from next-hop 7::1 and a 32-bit uDT4 function (0xfff0 4002) allocated from W-LIB

- BGP Path 2 from next-hop 9::1 and a 16-bit uDT4 function (0x4002) allocated from LIB
- BGP Path 3 from next-hop 8::1 and a 16-bit uDT4 function (0x4002) allocated from LIB

Note the following fields in the output:

- Function length of 16 bits for LIB and 32 bits for W-LIB
- Transposition offset (Tpose-offset) value of 48 bits for LIB and 64 bits for W-LIB
- Transposition length (Tpose-len) value of 16 bits for LIB/W-LIB

```
Router# show bgp vpnv4 unicast rd 100:2 2.2.0.1/32 detail

BGP routing table entry for 2.2.0.1/32, Route Distinguisher: 100:2
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               5314         5314
    Flags: 0x20061292+0x00060000; multipath; backup available;
Last Modified: Jan 20 14:37:59.189 for 00:00:19
Paths: (3 available, best #1)
  Not advertised to any peer
 Path #1: Received by speaker 0
  Flags: 0x2000000085070005+0x00, import: 0x39f
  Not advertised to any peer
  Local
    7::1 (metric 20) from 2::1 (192.0.0.1), if-handle 0x00000000
      Received Label 0x40020
      Origin IGP, localpref 150, valid, internal, best, group-best, multipath,
import-candidate, imported
      Received Path ID 1, Local Path ID 1, version 5314
      Extended community: RT:100:2
      Originator: 192.0.0.1, Cluster list: 2.0.0.1
      PSID-Type:L3, SubTLV Count:1, R:0x00,
       SubTLV:
        T:1(Sid information), Sid:fccc:cc00:7001:fff0::, F:0x00, R2:0x00, Behavior:63,
R3:0x00, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
            Length [Loc-blk,Loc-node,Func,Arg]:[32,16,32,0], Tpose-len:16, Tpose-offset:64
      Source AFI: VPNv4 Unicast, Source VRF: VRF_2, Source Route Distinguisher: 100:2
  Path #2: Received by speaker 0
  Flags: 0x2000000084060005+0x00, import: 0x096
  Not advertised to any peer
  Local
    9::1 (metric 20) from 2::1 (192.0.0.3), if-handle 0x00000000
      Received Label 0x40020
      Origin IGP, localpref 100, valid, internal, backup(protect multipath), add-path,
import-candidate, imported
      Received Path ID 2, Local Path ID 5, version 5314
      Extended community: RT:100:2
      Originator: 192.0.0.3, Cluster list: 2.0.0.1
      PSID-Type:L3, SubTLV Count:1, R:0x00,
       SubTLV:
        T:1(Sid information), Sid:fccc:cc00:9001::, F:0x00, R2:0x00, Behavior:63, R3:0x00,
 SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
            Length [Loc-blk,Loc-node,Func,Arg]:[32,16,16,0], Tpose-len:16, Tpose-offset:48
      Source AFI: VPNv4 Unicast, Source VRF: VRF_2, Source Route Distinguisher: 100:2
  Path #3: Received by speaker 0
  Flags: 0x2000000084070005+0x00, import: 0x296
  Not advertised to any peer
```

```
  Local
    8::1 (metric 20) from 2::1 (192.0.0.2), if-handle 0x00000000
      Received Label 0x400020
      Origin IGP, localpref 150, valid, internal, multipath, backup, add-path,
import-candidate, imported
      Received Path ID 3, Local Path ID 4, version 5314
      Extended community: RT:100:2
      Originator: 192.0.0.2, Cluster list: 2.0.0.1
      PSID-Type:L3, SubTLV Count:1, R:0x00,
       SubTLV:
        T:1(Sid information), Sid:fccc:cc00:8001::, F:0x00, R2:0x00, Behavior:63, R3:0x00,
 SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
           Length [Loc-blk,Loc-node,Func,Arg]:[32,16,16,0], Tpose-len:16, Tpose-offset:48
      Source AFI: VPNv4 Unicast, Source VRF: VRF_2, Source Route Distinguisher: 100:2
```

SRv6-Services: L3 Services with Local SIDs from W-LIB

Table 10: Feature History Table

Feature Name	Release	Description
SRv6-Services: L3 Services with Local SIDs from W-LIB	Release 7.11.1	This feature enables an SRv6 headend node to allocate and advertise local SIDs with Wide (32-bit) functions (Local W-LIB). The headend router utilizes the local W-LIB functionality to define and implement SR policies using SRv6 SIDs. The Local W-LIB is supported for Layer 3 (VPNv4/VPNv6/BGPv4/BGPv6 global) services. This feature introduces the **usid allocation wide-local-id-block** command.

An SRv6 Service SID is used to identify a specific service function. This Service SID inserted into the packet header by the source node is used to steer the packet along a specific path that includes the service function. This capability enhances flexibility and control over how packets are processed and enables efficient delivery of services within the network.

Note See *SRv6 uSID Allocation Within a uSID Block* for more information about W-LIB.

By default, BGP specifies to SID-Manager that allocation of uSIDs is from LIB space only. With this feature enabled, BGP can indicate to the SID-Manager that uSID allocation is to be enforced from W-LIB.

BGP performs transposition when encoding the service SID for VPN services to the label part of the NLRI, as described in IETF RFC 9252. In the current LIB implementation, BGP transposes the 16-bit function to the label field in the NLRI.

For W-LIB, BGP transposes the last 16-bits of the W-LIB 32-bit function to the label part of the NLRI for VPNv4 and VPNv6 routes. For more information on transposition, see the *SRv6 Services: L2 and L3 Services with Remote SIDs from W-LIB* section.

Note There is no transposition for BGPv4/BGPv6 global.

Usage Guidelines and Limitations

This feature is supported on Cisco 8000 Series Routers and Line Cards with Cisco Silicon One Q200 and P100 ASICs.

This feature is not supported on Cisco 8000 Series Routers and Line Cards with Cisco Silicon One Q100 ASICs.

Configuration

Use the **usid allocation wide-local-id-block** command to enable the allocation and advertisement of an SRv6 Service SID with wide function (W-LIB) for L3 services.

The precedence rules for the W-LIB allocation mode are applied at different levels:

- W-LIB uSID Allocation Applied Globally under BGP:

```
router bgp 1
  segment-routing srv6
    usid allocation wide-local-id-block
  !
```

- W-LIB uSID Allocation Applied at the IPv4/v6 Address Family under BGP:

```
router bgp 1
  address-family ipv4 unicast
    segment-routing srv6
      usid allocation wide-local-id-block
   !
  address-family ipv6 unicast
    segment-routing srv6
      usid allocation wide-local-id-block
```

- W-LIB uSID Allocation Applied for all VPNv4/v6 Address Family:

```
router bgp 1
  address-family vpnv4 unicast
    vrf all
      segment-routing srv6
        usid allocation wide-local-id-block
    !
  address-family vpnv6 unicast
    vrf all
      segment-routing srv6
        usid allocation wide-local-id-block
```

- W-LIB uSID Allocation Applied at the VRF IPv4/v6 Address Family:

```
router bgp 1
  vrf foo
    address-family ipv4 unicast
      segment-routing srv6
        usid allocation wide-local-id-block
    !
    address-family ipv6 unicast
```

```
segment-routing srv6
  usid allocation wide-local-id-block
```

Verification

The following ouput shows the W-LIB uSID allocation:

```
RP/0/0/CPU0:PE1# show bgp ipv4 unicast process

BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 100
Router ID: 192.168.0.1
Default Cluster ID: 192.168.0.1
Active Cluster IDs:  192.168.0.1
Fast external fallover enabled
Platform Loadbalance paths max: 16
Platform RLIMIT max: 2147483648 bytes
Maximum limit for BMP buffer size: 409 MB
Default value for BMP buffer size: 307 MB
Current limit for BMP buffer size: 307 MB
Current utilization of BMP buffer limit: 0 B
Neighbor logging is enabled
Enforce first AS enabled
AS Path multipath-relax is enabled
Use SR-Policy admin/metric of color-extcomm Nexthop during path comparison: disabled
Default local preference: 100
Default keepalive: 60
Graceful restart enabled
Restart time: 120
Stale path timeout time: 360
RIB purge timeout time: 600
Non-stop routing is enabled
ExtComm Color Nexthop validation: RIB

Update delay: 120
Generic scan interval: 15
Configured Segment-routing Local Block: [0, 0]
In use Segment-routing Local Block: [15000, 15999]
Platform support mix of sr-policy and native nexthop: No
Segment Routing SRv6 Locator Name: LOC2
Segment Routing SRv6 uSID WLIB allocation: Enforced

Address family: IPv4 Unicast
Dampening is enabled
Client reflection is enabled in global config
Dynamic MED is Disabled
Dynamic MED interval : 10 minutes
Dynamic MED Timer : Running, will expire in 342 seconds
Dynamic MED Periodic Timer : Running, will expire in 42 seconds
Scan interval: 60
Total prefixes scanned: 42
Prefixes scanned per segment: 100000
Number of scan segments: 1
Nexthop resolution minimum prefix-length: 0 (not configured)
IPv6 Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 44
Table version synced to RIB: 44
Table version acked by RIB: 44
IGP notification: IGPs notified
RIB has converged: version 0
```

```
RIB table prefix-limit reached ?  [No], version 0
Permanent Network Unconfigured
Segment Routing SRv6 Alloc Mode: 0
Segment Routing SRv6 uSID WLIB allocation: Enforced


RP/0/0/CPU0:PE1# show bgp vrf all ipv4 unicast process

VRF: foo
-------

BGP Process Information: VRF foo
BGP Route Distinguisher: 23:1

BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 100
Router ID: 192.168.0.1
Default Cluster ID: 192.168.0.1
Active Cluster IDs:  192.168.0.1
Fast external fallover enabled
Platform Loadbalance paths max: 16
Platform RLIMIT max: 2147483648 bytes
Maximum limit for BMP buffer size: 409 MB
Default value for BMP buffer size: 307 MB
Current limit for BMP buffer size: 307 MB
Current utilization of BMP buffer limit: 0 B
Neighbor logging is enabled
Enforce first AS enabled
iBGP to IGP redistribution enabled
AS Path multipath-relax is enabled
Use SR-Policy admin/metric of color-extcomm Nexthop during path comparison: disabled
Default local preference: 100
Default keepalive: 60
Graceful restart enabled
Restart time: 120
Stale path timeout time: 360
RIB purge timeout time: 600
Non-stop routing is enabled
ExtComm Color Nexthop validation: RIB

Update delay: 120
Generic scan interval: 15
Configured Segment-routing Local Block: [0, 0]
In use Segment-routing Local Block: [15000, 15999]
Platform support mix of sr-policy and native nexthop: No
Segment Routing SRv6 Locator Name: LOC2 (WLIB allocation enforced)
Segment Routing SRv6 uSID WLIB allocation: Enforced

VRF foo Address family: IPv4 Unicast
Dampening is enabled
Client reflection is not enabled in global config
Dynamic MED is Disabled
Dynamic MED interval : 10 minutes
Dynamic MED Timer : Not Running
Dynamic MED Periodic Timer : Not Running
Scan interval: 60
Total prefixes scanned: 85
Prefixes scanned per segment: 100000
Number of scan segments: 1
Nexthop resolution minimum prefix-length: 0 (not configured)
IPv6 Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 152
Table version synced to RIB: 152
```

```
Table version acked by RIB: 152
IGP notification: IGPs notified
RIB has converged: version 1
RIB table prefix-limit reached ?  [No], version 0
Permanent Network Unconfigured
Segment Routing SRv6 uSID WLIB allocation: Enforced
```

The following output shows the advertized SRv6 W-LIB uSID for the default VRF:

```
RP/0/0/CPU0:PE1# show bgp ipv4 unicast 192.168.4.1/32

BGP routing table entry for 192.168.4.1/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 419          419
    SRv6-VPN SID: fccc:cccc:a:fff0:4::/80
Last Modified: Apr  3 10:35:41.000 for 136y10w
Paths: (1 available, best #1)
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Local
    0.0.0.0 from 0.0.0.0 (192.168.0.1)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, redistributed, best,
 group-best
      Received Path ID 0, Local Path ID 1, version 419
```

The following output shows the advertized SRv6 W-LIB uSID for a specific VRF (foo):

```
RP/0/0/CPU0:PE1# show bgp vrf foo 192.168.7.1/32

BGP routing table entry for 192.168.7.1/32, Route Distinguisher: 23:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 439          439
    SRv6-VPN SID: fccc:cccc:a:fff0:4::/80
Last Modified: Apr  3 10:31:00.000 for 00:00:44
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    192::4
  Advertised to CE peers (in unique update groups):
    10.10.10.2
  Path #1: Received by speaker 0
  Advertised to PE peers (in unique update groups):
    192::4
  Advertised to CE peers (in unique update groups):
    10.10.10.2
  Local
    0.0.0.0 from 0.0.0.0 (192.168.0.1)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, redistributed, best,
 group-best, import-candidate
      Received Path ID 0, Local Path ID 1, version 439
      Extended community: RT:23:23
```

SRv6/MPLS L3 Service Interworking Gateway

Table 11: Feature History Table

Feature Name	Release	Description
Identical Route Distinguisher (RD) for Interworking Gateways between MPLS and SRv6 Domains	Release 24.1.1	You can now configure the same Route Distinguisher (RD) for interworking gateways catering to both MPLS and SRv6 domains that help conserve hardware resources, reduce the BGP table scale and minimize the processing load on routers. At the same time, it ensures seamless connectivity across SRv6 and MPLS L3 EVPN domains, thus promoting interoperability and efficiency in modern network environments. Previously, a unique RD was required to extend L3 services between MPLS and SRv6 domains resulting in higher router load and resource consumption, which could have affected performance.
SRv6/MPLS L3 Service Interworking Gateway (SRv6 Micro-SID)	Release 7.8.1	This feature enables you to extend L3 services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane. This feature allows for SRv6 L3VPN domains to interwork with existing MPLS L3VPN domains. The feature also allows migration from MPLS L3VPN to SRv6 L3VPN.

SRv6/MPLS L3 Service Interworking Gateway enables you to extend L3 services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane.

This feature allows for SRv6 L3VPN domains to interwork with existing MPLS L3VPN domains. The feature also allows a way to migrate from MPLS L3VPN to SRv6 L3VPN.

The SRv6/MPLS L3 Service Interworking Gateway provides both transport and service termination at the gateway node. The gateway generates both SRv6 VPN SIDs and MPLS VPN labels for all prefixes under the VRF configured for re-origination. The gateway supports traffic forwarding from MPLS domain to SRv6 domain by popping the MPLS VPN label, looking up the destination prefix, and pushing the appropriate SRv6 encapsulation. From SRv6 domain to MPLS domain, the gateway removes the outer IPv6 header, looks up the destination prefix, and pushes the VPN and next-hop MPLS labels.

VRFs on the gateway node are configured with 2 sets of route targets (RTs):

- MPLS L3VPN RTs
- SRv6 L3VPN RTs (called *stitching RTs*)

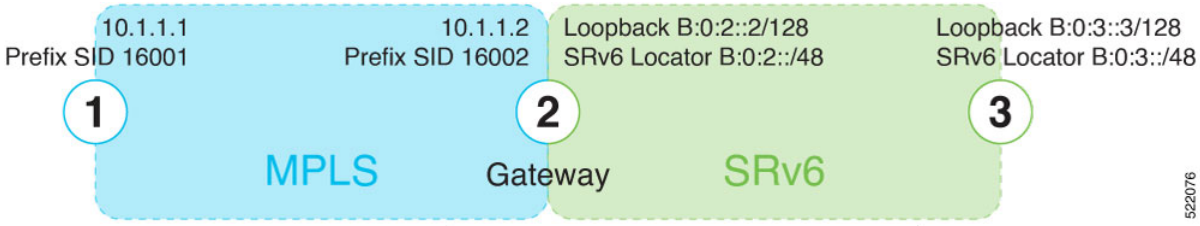
The gateway performs the following actions:

- Imports service routes received from one domain (MPLS or SRv6)
- Re-advertises exported service routes to the other domain (next-hop-self)
- Stitches the service on the data plane (uDT4/H.Encaps.Red ↔ service label)

SRv6/MPLS L3 Service Interworking Gateway Scenarios

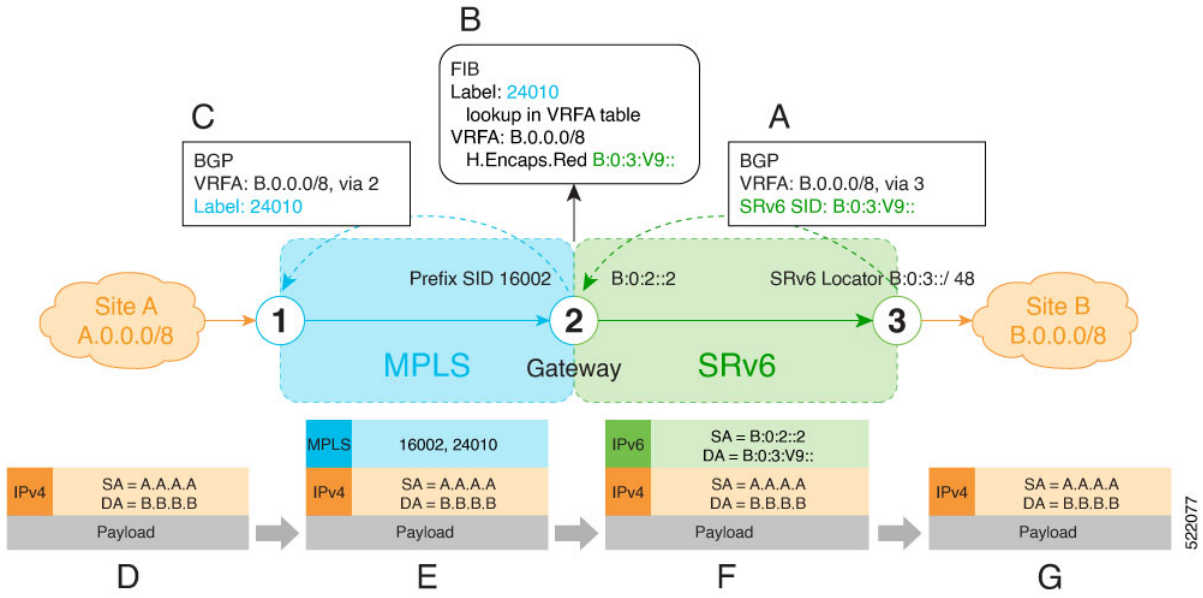
The following scenario is used to describe the gateway functionality:

- Node 1 is an L3VPN PE in the MPLS domain with an SR prefix SID label of 16001 for its Loopback interface 1.1.1.1/32.
- Node 2 is the SRv6/MPLS L3 Service Interworking Gateway. In the MPLS domain, it has an SR prefix SID label of 16002 for its Loopback interface 1.1.1.2/32. In the SRv6 domain, it has an SRv6 locator of B:0:2::/48 and Loopback interface B:0:2::2/128.
- Node 3 is an L3VPN PE in the SRv6 domain with SRv6 locator of B:0:3::/48 and Loopback interface B:0:3::3/128.



Scenario 1: SRv6-to-MPLS Control-Plane Direction/MPLS-to-SRv6 Data-Plane Direction

The figure below describes the associated control-plane behaviors in the SRv6-to-MPLS direction for traffic in the MPLS-to-SRv6 data-plane direction.



A. Node 3 advertises a BGP L3VPN update for prefix B.0.0.0/8 with RD corresponding to VRFA, including the SRv6 VPN SID (B:0:3:V9::) assigned to this VRF, in the SRv6 domain.

Note SRv6 uDT4 function value "V9" is not a valid hex number, however it is used for illustration purposes to remind you of its connection to a VRF.

B. Node 2 (gateway) imports the BGP L3VPN update and programs its FIB:

- MPLS label 24010 is allocated for VRFA
- Prefix B.0.0.0/8 is programmed with an "SR Headend Behavior with Reduced Encapsulation in an SR Policy" function (H.Encaps.Red) of B:0:3:V9::

Note The gateway follows per-VRF label and per-VRF SID allocation methods.

C. Node 2 re-originates a BGP L3VPN update for the same prefix, including the MPLS VPN label (24010) allocated for the VRF, in the MPLS domain.

D. Site A sends traffic to an IPv4 prefix (B.B.B.B) of Site B

E. Node 1 encapsulates incoming traffic with the MPLS VPN label (24010) and the prefix SID MPLS label (16002) of the BGP next-hop (Node 2).

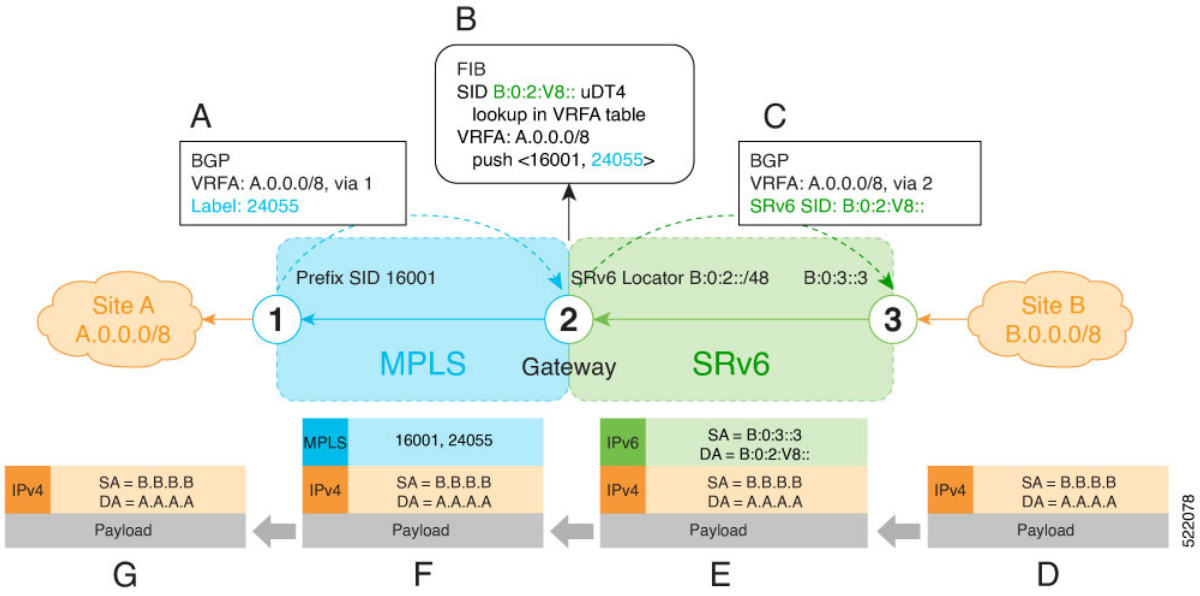
F. Node 2 performs the following actions:

- Pops the MPLS VPN label and looks up the destination prefix
- Encapsulates the payload in an outer IPv6 header with destination address (DA) equal to the H.Encaps.Red function (B:0:3:V9::)

G. Node 3 removes the outer IPv6 header, looks up the payload destination address (B.B.B.B), and forwards to Site B.

Scenario 2: MPLS-to-SRv6 Control-Plane Direction/SRv6-to-MPLS Data-Plane Direction

The figure below describes the associated control-plane behaviors in the MPLS-to-SRv6 direction for traffic in the SRv6-to-MPLS data-plane direction.



A. Node 1 advertises a BGP L3VPN update for prefix A.0.0.0/8 with RD corresponding to VRFA, including the MPLS VPN label (24055) assigned to this VRF, in the MPLS domain.

B. Node 2 (gateway) imports the BGP L3VPN update and programs its FIB:

- Prefix A.0.0.0/8 is programmed to impose an MPLS VPN label (24055) and the prefix SID MPLS label (16001) of the BGP next-hop (Node 1)
- "Endpoint with decapsulation and IPv4 table lookup" function (uDT4) of B:0:2:V8:: is allocated to VRFA

Note SRv6 uDT4 function value "V8" is not a valid hex number, however it is used for illustration purposes to remind you of its connection to a VRF.

Note The gateway follows per-VRF label and per-VRF SID allocation methods.

C. Node 2 re-originates a BGP L3VPN update for the same prefix, including the uDT4 function (B:0:2:V8::) allocated for the VRF, in the SRv6 domain.

D. Site B sends traffic to an IPv4 prefix (A.A.A.A) of Site A.

E. Node 3 Encapsulates the payload in an outer IPv6 header with destination address (DA) equal to the uDT4 function (B:0:2:V8::).

F. Node 2 performs the following actions:

- Removes the outer IPv6 header and looks up the destination prefix
- Pushes the MPLS VPN label (24055) and the prefix SID MPLS label (16001) of the BGP next-hop (Node 1)

G. Node 1 pops the MPLS VPN label, looks up the payload destination address (A.A.A.A), and forwards to Site A.

Configuration

This example shows how to enable SRv6 with locator and configure encapsulation parameters:

```
Node2(config)# segment-routing srv6
Node2(config-srv6)# encapsulation source-address a::2
Node2(config-srv6)# locators
Node2(config-srv6-locators)# locator LOC1
Node2(config-srv6-locator)# prefix b:2::/64
```

This example shows how to configure a VRF with 2 sets of route targets (RTs):

- MPLS L3VPN RT of 1111:1
- SRv6 L3VPN RT of 2222:1 (stitching RT)

```
Node2(config)# vrf ACME
Node2(config-vrf)# address-family ipv4 unicast
Node2(config-vrf-af)# import route-target
Node2(config-vrf-import-rt)# 1111:1
Node2(config-vrf-import-rt)# 2222:1 stitching
Node2(config-vrf-import-rt)# exit
Node2(config-vrf-af)# export route-target
Node2(config-vrf-export-rt)# 1111:1
Node2(config-vrf-export-rt)# 2222:1 stitching
```

This example shows how to configure BGP SRv6:

```
Node2(config)# router bgp 100
Node2(config-bgp)# segment-routing srv6
Node2(config-bgp-gbl-srv6)# locator LOC1
Node2(config-bgp-gbl-srv6)# exit

Node2(config-bgp)# neighbor 1.1.1.1
Node2(config-bgp-nbr)# address-family vpnv4 unicast
Node2(config-bgp-nbr-af)# import re-originate stitching-rt
Node2(config-bgp-nbr-af)# route-reflector-client
Node2(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Node2(config-bgp-nbr-af)# exit
Node2(config-bgp-nbr)# exit

Node2(config-bgp)# neighbor a::3
Node2(config-bgp-nbr)# address-family vpnv4 unicast
Node2(config-bgp-nbr-af)# import stitching-rt re-originate
Node2(config-bgp-nbr-af)# route-reflector-client
Node2(config-bgp-nbr-af)# encapsulation-type srv6
Node2(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
Node2(config-bgp-nbr-af)# exit
Node2(config-bgp-nbr)# exit
```

```
Node2(config-bgp)# vrf ACME
Node2(config-bgp-vrf)# address-family ipv4 unicast
Node2(config-bgp-vrf-af)# enable label-mode
Node2(config-bgp-vrf-af)# segment-routing srv6
Node2(config-bgp-vrf-af-srv6)# alloc mode per-vrf
Node2(config-bgp-vrf-af-srv6)# commit
```

Example

Leveraging the topology described in the above use-case, this example shows the SRv6/MPLS L3 Service Interworking Gateway configuration required at Node 2.

The following configuration shows how to enable SRv6 with locator and configure encapsulation parameters:

```
segment-routing
 srv6
  encapsulation
   source-address B:0:2::2
  !
  locators
   locator LOC1
    prefix B:0:2::/48
   !
  !
 !
!
```

The following configuration shows how to configure a VPNv4 VRF with the following route targets (RTs):

- 1111:1, RT used for MPLS L3VPN
- 2222:1, RT used for SRv6 L3VPN (stitching RT)

```
vrf ACME
 address-family ipv4 unicast
  import route-target
   1111:1
   2222:1 stitching
  !
  export route-target
   1111:1
   22222:1 stitching
  !
 !
!
```

The following configuration shows how to configure SRv6/SRv6 VPNs under BGP:

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 neighbor 1.1.1.1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   route-reflector-client
   advertise vpnv4 unicast re-originated
 !
 neighbor B:0:3::1
  address-family vpnv4 unicast
   import stitching-rt re-originate
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated stitching-rt
```

```
 !
 vrf ACME
  address-family ipv4 unicast
   enable label-mode
   segment-routing srv6
```

You can configure same route distinguisher (RD) on the Node 1, Node 2 and GW. This example shows how to configure same route distinguisher (RD) on the Node 1, Node 2 and GW. In this example, **rd 5000:2** is used on Node 1, Node 2 and GW.

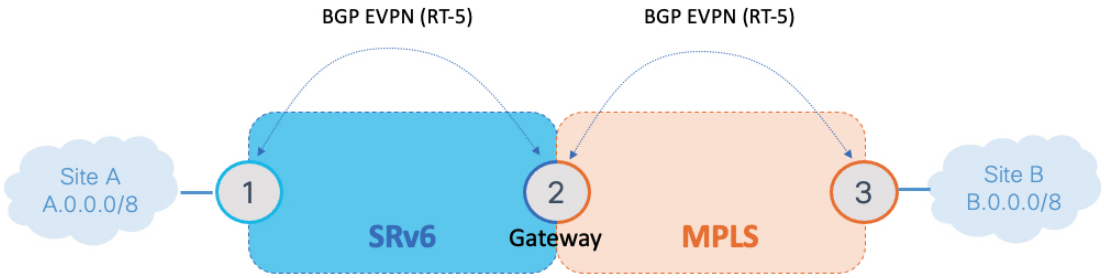
```
/* Configuration on Node 1*/
vrf ACME
rd 5000:2
 address-family ipv4 unicast
  import route-target
   1111:1
   2222:1 stitching
  !
  export route-target
   1111:1
   22222:1 stitching
  !
 !
!

/* Configuation on Node 2*/
vrf ACME
rd 5000:2
 address-family ipv4 unicast
  import route-target
   1111:1
   2222:1 stitching
  !
  export route-target
   1111:1
   22222:1 stitching
  !
 !
!

/* Configuration  on GW*/
vrf ACME
rd 5000:2
 address-family ipv4 unicast
  import route-target
   1111:1
   2222:1 stitching
  !
  export route-target
   1111:1
   22222:1 stitching
  !
 !
!
```

L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway

This feature adds support for L3 EVPN interworking between SRv6 and MPLS.

L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway enables you to extend L3 EVPN services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane.

This feature allows for SRv6 L3 EVPN domains to interwork with existing MPLS L3 EVPN domains. The feature also allows a way to migrate from MPLS L3 EVPN to SRv6 L3 EVPN.



The L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway provides both transport and service termination at the gateway node.

VRFs on the gateway node are configured with 2 sets of route targets (RTs):

- L3 EVPN/MPLS RTs
- L3 EVPN/SRv6 RTs (called *stitching RTs*)

The gateway performs the following actions:

- Imports service routes received from one domain (L3 EVPN/MPLS or L3 EVPN/SRv6)
- Re-originates exported service routes to the other domain and setting next-hop-self
- Stitches the service routes in the data plane (uDT4/H.Encaps.Red ↔ MPLS service label)

The gateway generates both L3 EVPN/SRv6 SIDs and L3 EVPN/MPLS labels for all prefixes under the VRF configured for re-origination:

- MPLS-to-SRv6 Control Plane Direction

  The gateway imports routes received from the MPLS side (via EVPN RT5) and re-originates them in L3VPN VRF with a per-VRF SRv6 SID.

- SRv6-to-MPLS Control Plane Direction

  The gateway imports routes received from the SRv6 side (via EVPN RT5) and re-originates them in L3VPN VRF with a per-VRF label.

In the data plane, the gateway forwards traffic from the MPLS domain to the SRv6 domain by popping the MPLS L3 EVPN label, looking up the destination prefix, and pushing the appropriate SRv6 encapsulation. In the opposite direction, the gateway removes the outer IPv6 header, looks up the destination prefix, and pushes the L3 EVPN and next-hop MPLS labels.

Usage Guidelines and Limitations

L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway is supported for IPv4 and IPv6.

Configuration Example

Leveraging the topology described above, this example shows the SRv6/MPLS L3 EVPN Service Interworking Gateway configuration required at Node 2.

The following configuration shows how to enable SRv6 with locator and configure encapsulation parameters.

```
segment-routing
 srv6
  encapsulation
   source-address b:0:2::2
  !
  locators
   locator LOC1
    prefix b:0:2::/48
   !
  !
 !
!
```

The following configuration shows how to configure a VPNv4/VPNv6 VRF with the following route targets (RTs):

- 1111:1, RT used for MPLS L3 EVPN
- 2222:1, RT used for SRv6 L3 EVPN (stitching RT)

```
vrf VPN1
 address-family ipv4 unicast
  import route-target
   1111:1
   2222:1 stitching
  !
  export route-target
   1111:1
   2222:1 stitching
  !
 !
 address-family ipv6 unicast
  import route-target
   1111:1
   2222:1 stitching
  !
  export route-target
   1111:1
   2222:1 stitching
  !
 !
!
```

The following configuration shows how to configure SRv6/SRv6 VPNs under BGP:

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
 !
 neighbor 2222::2
```

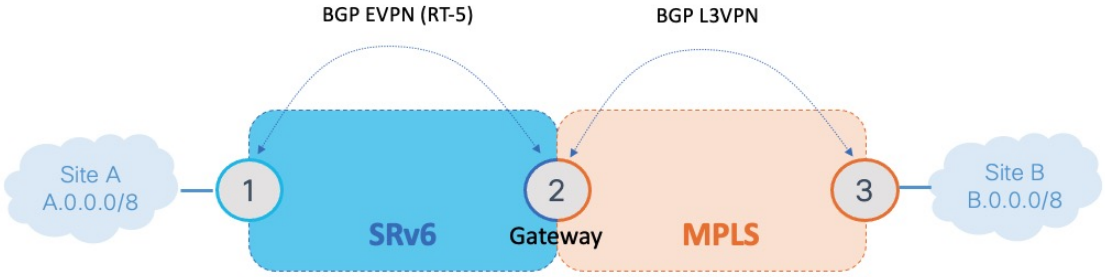
```
  remote-as 100
  description SRv6 side peering
  address-family l2vpn evpn
   import reoriginate stitching-rt (Imports NLRIs that match normal route target identifier

             and exports re-originated NLRIs assigned with the stitching route target
identifier)
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated (Specifies advertisement of re-originated VPNv4
             unicast routes)
   advertise vpnv6 unicast re-originated (Specifies advertisement of re-originated VPNv6
             unicast routes)
  !
 !
 neighbor 3.3.3.3
  remote-as 100
  description MPLS side peering stitching side
  address-family l2vpn evpn
   import stitching-rt reoriginate (Imports NLRIs that match stitching route target identifier

            and exports re-originated NLRIs assigned with the normal route target identifier)

   advertise vpnv4 unicast re-originated stitching-rt (Advertise local VPNv4 unicast routes

             assigned with stitching route target identifier)
   advertise vpnv6 unicast re-originated stitching-rt (Advertise local VPNv6 unicast routes

             assigned with stitching route target identifier)
  !
 !
 vrf VPN1
  rd 100:2
  address-family ipv4 unicast
   mpls alloc enable
  !
  address-family ipv6 unicast
   mpls alloc enable
  !
 !
!
```

L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway

This feature adds support for EVPN L3VPN interworking between SRv6 and MPLS.

L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway enables you to extend L3 services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane.

This feature allows for SRv6 L3 EVPN domains to interwork with existing MPLS L3VPN domains. The feature also allows a way to migrate from MPLS L3VPN to SRv6 L3 EVPN.



The L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway provides both transport and service termination at the gateway node.

VRFs on the gateway node are configured with 2 sets of route targets (RTs):

- L3VPN/MPLS RTs
- L3 EVPN/SRv6 RTs (called *stitching RTs*)

The gateway performs the following actions:

- Imports service routes received from one domain (L3VPN/MPLS or L3 EVPN/SRv6)
- Re-originates exported service routes to the other domain and setting next-hop-self
- Stitches the service routes in the data plane (uDT4/H.Encaps.Red ↔ MPLS service label)

The gateway generates both L3 EVPN/SRv6 SIDs and L3VPN/MPLS labels for all prefixes under the VRF configured for re-origination:

- MPLS to SRv6 Control Plane Direction

  The gateway imports routes received from the MPLS side (via EVPN RT5) and re-originates them in L3 EVPN VRF with a per-VRF SRv6 SID.

- SRv6 to MPLS Control Plane Direction

  The gateway imports routes received from the SRv6 side (via EVPN RT5) and re-originates them in L3VPN VRF with a per-VRF label.

In the data plane, the gateway forwards traffic from the MPLS domain to the SRv6 domain by popping the MPLS L3VPN label, looking up the destination prefix, and pushing the appropriate SRv6 encapsulation. In the opposite direction, the gateway removes the outer IPv6 header, looks up the destination prefix, and pushes the L3VPN and next-hop MPLS labels.

Usage Guidelines and Limitations

L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway is supported for IPv4 and IPv6.

Configuration Example

The following configuration shows how to enable SRv6 with locator and configure encapsulation parameters:

```
segment-routing
 srv6
  encapsulation
   source-address b:0:2::2
  !
```

```
  locators
   locator LOC1
    prefix b:0:2::/48
   !
  !
 !
!
```

The following configuration shows how to configure a VPNv4/VPNv6 VRF with the following route targets (RTs):

- **1111:1**, RT used for MPLS L3 EVPN
- **2222:1**, RT used for SRv6 L3 EVPN (stitching RT)

```
vrf VPN1
 address-family ipv4 unicast
  import route-target
   1:1
   1:1 stitching
  !
  export route-target
   1:1
   1:1 stitching
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
   1:1 stitching
  !
  export route-target
   1:1
   1:1 stitching
  !
 !
!
```

The following configuration shows how to configure SRv6/SRv6 VPNs under BGP:

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
 !
 neighbor 2222::2
  remote-as 100
  description SRv6 side peering
  address-family l2vpn evpn
   import reoriginate stitching-rt (Imports NLRIs that match normal route target identifier

             and exports re-originated NLRIs assigned with the stitching route target
identifier)
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated (Specifies advertisement of re-originated VPNv4
             unicast routes)
```

```
   advertise vpnv6 unicast re-originated (Specifies advertisement of re-originated VPNv6
             unicast routes)
  !
 !
 neighbor 3.3.3.3
  remote-as 100
  description MPLS side peering stitching side
  address-family vpnv4 unicast
   import stitching-rt reoriginate (Imports NLRIs that match stitching route target identifier

           and exports re-originated NLRIs assigned with the normal route target identifier)

   route-reflector-client
   advertise vpnv4 unicast re-originated stitching-rt (Advertise local VPNv4 unicast routes

             assigned with stitching route target identifier)
  !
  address-family vpnv6 unicast
   import stitching-rt reoriginate (Imports NLRIs that match stitching route target identifier

           and exports re-originated NLRIs assigned with the normal route target identifier)

   route-reflector-client
   advertise vpnv4 unicast re-originated stitching-rt (Advertise local VPNv4 unicast routes

             assigned with stitching route target identifier)
  !
 !
 vrf VPN1
  rd 100:2
  address-family ipv4 unicast
   mpls alloc enable
  !
  address-family ipv6 unicast
   mpls alloc enable
  !
 !
!
```

SRv6/MPLS Dual-Connected PE

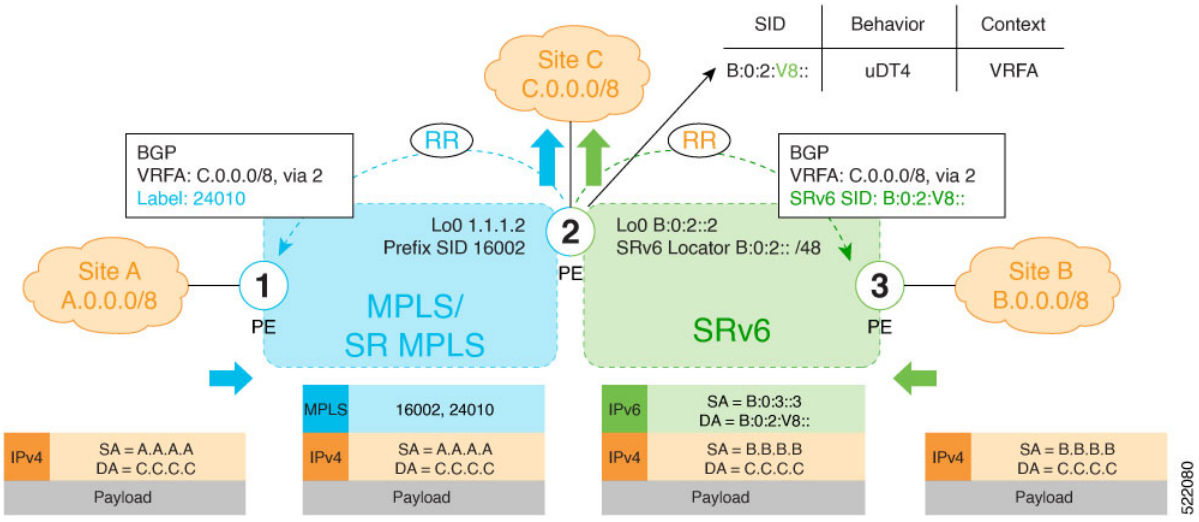
Table 12: Feature History Table

Feature Name	Release	Description
SRv6/MPLS Dual-Connected PE (SRv6 Micro SID)	Release 7.8.1	This feature allows a PE router to support IPv4 L3VPN services for a given VRF with both MPLS and SRv6. This is MPLS and SRv6 L3VPNv4 co-existence scenario and is sometimes referred to as dual-connected PE.

A PE router can support IPv4 or IPv6 L3VPN service for a given VRF with both MPLS and SRv6. This is MPLS and SRv6 L3VPNv4 co-existence scenario and is sometimes referred to as dual-connected PE.

In the figure below, node 2 is a dual-connected PE to Site C, providing:

- MPLS/IPv4 L3VPN between Site A and Site C
- SRv6/IPv4 L3VPN between Site B and Site C



Configure BGP to Support Dual-Mode

Enable MPLS Label Allocation

Use the **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast mpls alloc enable** command under the VRF address-family to enable per-prefix mode for MPLS labels. Additionally, use the **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast label mode** {**per-ce** | **per-vrf**} command to choose the type of label allocation.

```
Router(config)# router bgp 100
Router(config-bgp)# vrf blue
Router(config-bgp-vrf)# rd 1:10
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# mpls alloc enable
Router(config-bgp-vrf-af)# label mode per-ce
Router(config-bgp-vrf-af)# segment-routing srv6
Router(config-bgp-vrf-af-srv6)# alloc mode per-ce
Router(config-bgp-vrf-af-srv6)# exit
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# exit
Router(config-bgp)#
```

Configure Encaps on Neighbor to Send the SRv6 SID Toward the SRv6 Dataplane

By default, if a VRF prefix has both an MPLS label and an SRv6 SID, the MPLS label is sent when advertising the prefix to the PE. To advertise a VRF prefix with an SRv6 SID to an SRv6 session, use the **encapsulation-type srv6** command under the neighbor VPN address-family.

```
Router(config-bgp)# neighbor 192::6
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# encapsulation-type srv6
Router(config-bgp-nbr-af)# exit
```

Running Config

```
router bgp 100
 neighbor 192::6
  remote-as 1
  address-family ipv4 unicast
   encapsulation-type srv6
  !
 !
 vrf blue
  rd 1:10
  address-family ipv4 unicast
   mpls alloc enable
   label mode per-ce
   segment-routing srv6
    alloc mode per-ce
   !
  !
 !
!
```

SRv6 Provider Edge (PE) Lite Support

Table 13: Feature History Table

Feature Name	Release	Description
SRv6 Provider Edge (PE) Lite	Release 7.5.3	This feature provides VPN de-multiplexing-only behaviors (End.DT4/DT6/DT46) at an SRv6 PE node. This allows for a lightweight-PE implementation (no VPN encapsulation) that steers SRv6-encapsulated traffic across an SR-MPLS backbone after performing a VPN lookup.

SRv6 Provider Edge (PE) Lite leverages SRv6 programmability (SRv6 SID as a service ID) to steer traffic across SR MPLS (non-SRv6) backbone.

Service traffic is encapsulated with an explicit SRv6 End.DT46 SID in ingress PE for a VRF.

Note See Configuring Explicit End.DT46 SRv6 SIDs, on page 116 for information about explicit End.DT46 SRv6 SIDs.

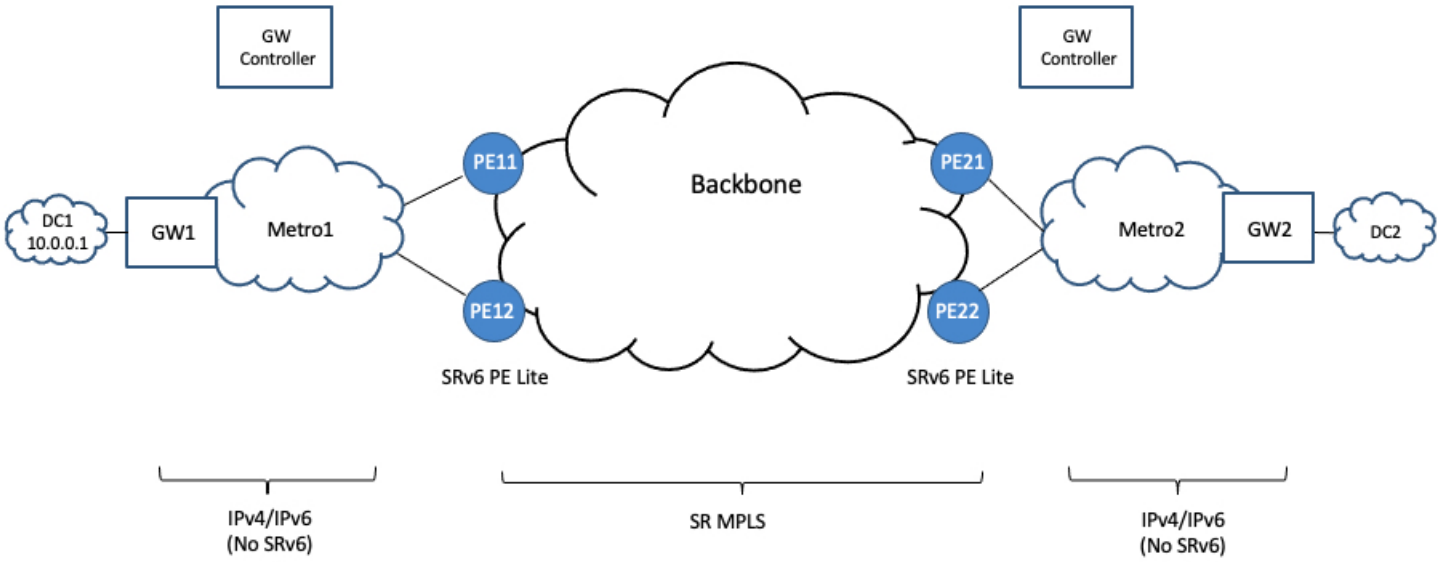
When traffic arrives at the ingress PE, it performs End.DT46 SRv6 end-point behavior.

The backbone leverages MPLS L3VPN and SR-TE MPLS (with route coloring and Automated Steering) to transport the traffic to the egress nodes in the backbone via different explicitly specified SLA paths using an SR-TE policy.

Use Case

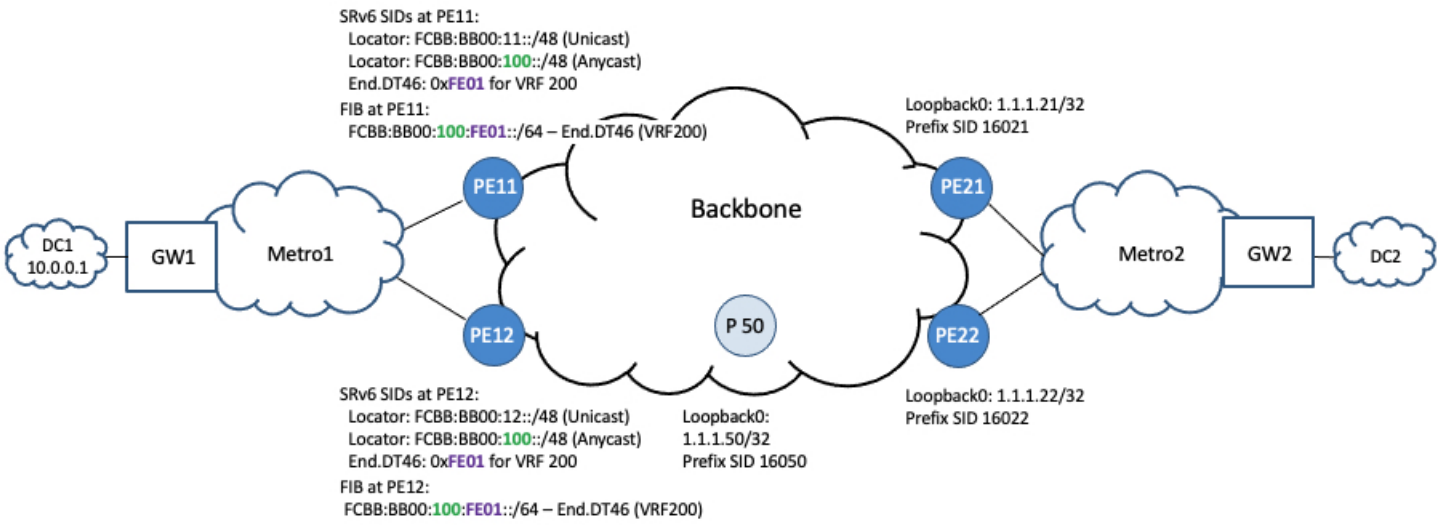
The figure below shows a use case where inter-data-center traffic is encapsulated in SRv6 (IP-in-IPv6) and is carried over IP-only metro domains and then over an SR-MPLS backbone.

Figure 12: Example Topology



Data center gateways (GW1 and GW2) perform IP-in-IPv6 encapsulation where the outer IPv6 destination address represents an SRv6 network program that leads traffic to the SRv6 PE lite nodes (PE11 and PE12). This outer IPv6 destination address is determined by the gateway controller to provide a desired transport SLA to an application over the backbone. The SRv6 PE lite nodes remove the SRv6 encapsulation and perform a lookup of the original encapsulated packet's IP destination address in the routing table of an MPLS VPN built over the backbone. The prefixes in the VPN table are associated with different transport SLAs (for example, best-effort or minimum delay). These prefixes can be steered over the native SR LSP or an SR-TE policy path, according to automated steering (AS) principles.

Figure 13: SRv6 Locator/Functions and SR-MPLS Prefix SIDs (Traffic Direction – DC1 to DC2)

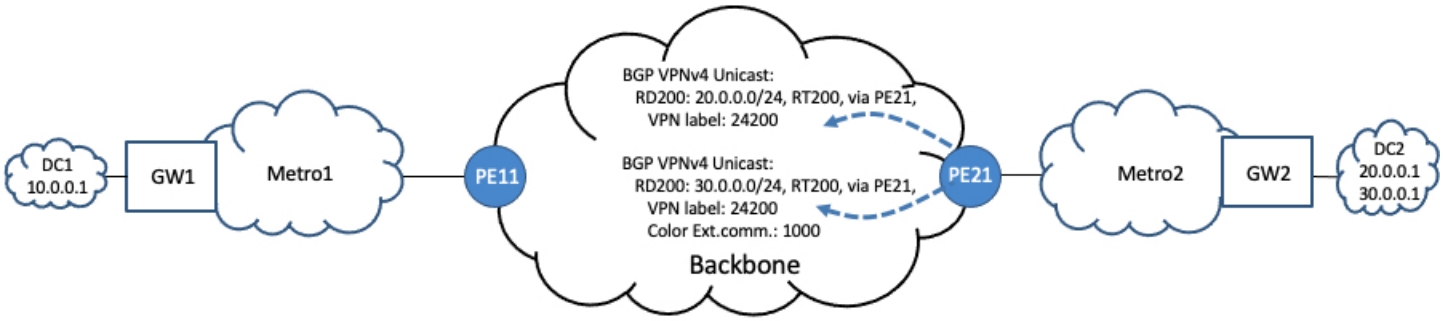


The SRv6 PE lite nodes are configured with SRv6 locators and explicit (manually assigned) service de-multiplexing end-point behaviors to perform decapsulation and VPN table lookup.

For high-availability, the SRv6 PE lite nodes are configured with an Anycast SRv6 locator (same locator in multiple nodes) and explicit end-point behavior with a common value among them. As a result, failure of a given SRv6 PE lite node can be handled by other nodes with the same Anycast locator and end-point behavior.

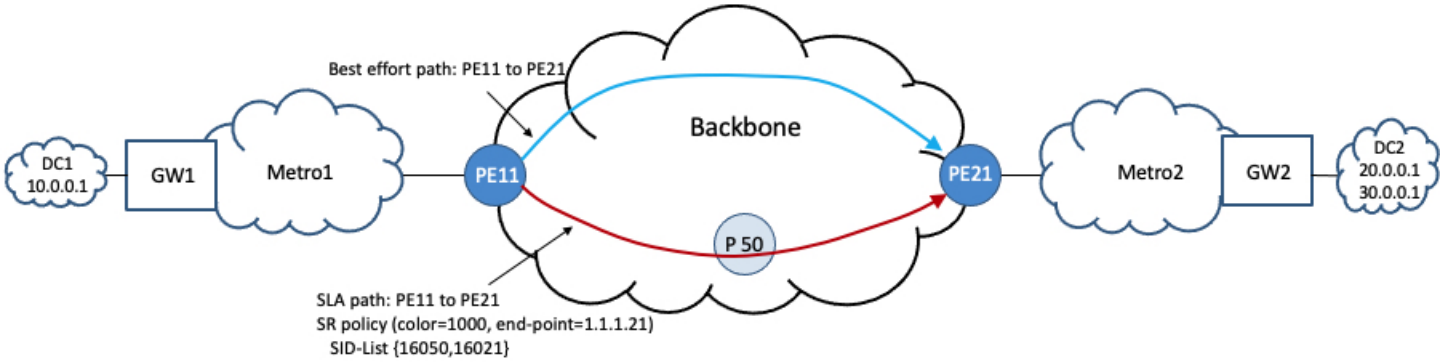
For example, SRv6 PE lite nodes (PE11 and PE12) are configured with the Anycast SRv6 locator (FCBB:BB00:100::/48) and a common End.DT46 function (0xFE01) associated with MPLS VPN VRF 200. The SRv6 PE lite nodes, which are part of the SR MPLS backbone, are configured with corresponding prefix SIDs.

Figure 14: BGP VPN Overlay Route Advertisement (Traffic Direction – DC1 to DC2)



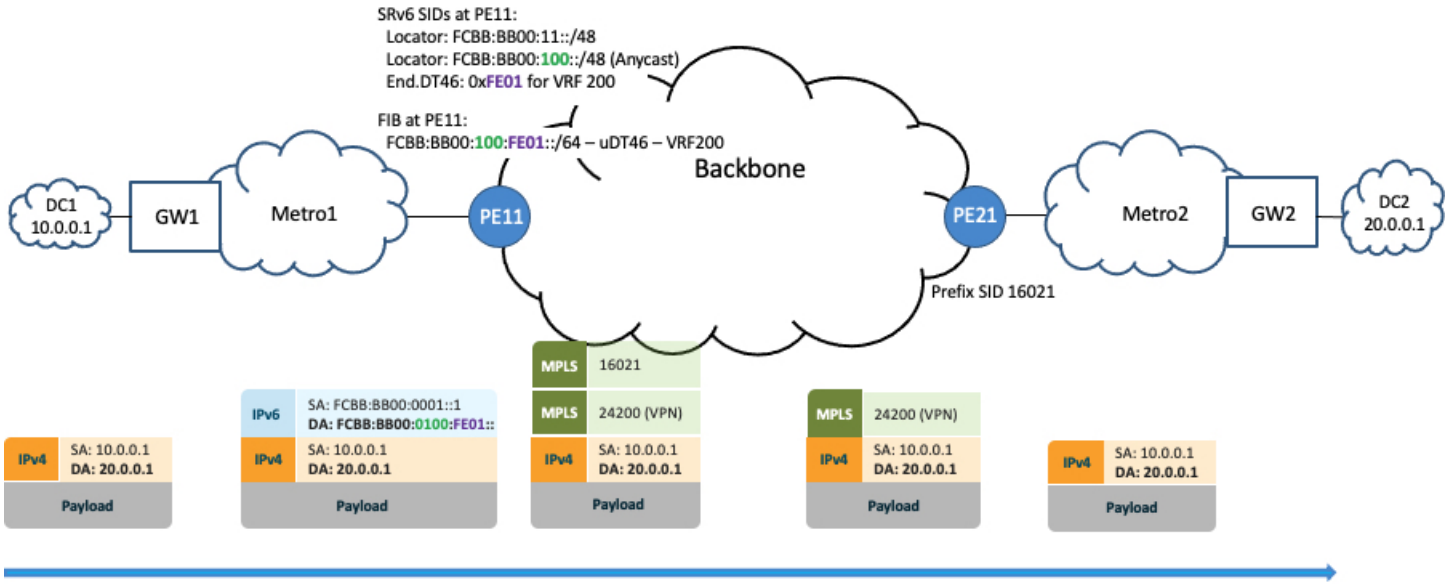
Prefixes from the data center are advertised in the backbone via multiprotocol BGP as part of a VPN. These prefixes can include a color extended community in order to indicate the desired transport SLA. For example, PE21 advertises BGP VPN overlay routes for DC2, 20.0.0.0/24 and 30.0.0.0/24. Prefix 20.0.0.0/24 requires best-effort treatment. Prefix 30.0.0.0/24 requires a transport SLA indicated by the presence of color extended community of value 1000.

Figure 15: Backbone Transport Paths (Traffic Direction – DC1 to DC2)



For traffic in the direction DC1 to DC2, the SRv6 PE lite node PE11 is an SR-TE headend of an SR policy associated with color 1000 and end-point of PE21. This SR policy will be used to steer traffic toward BGP service routes with color 1000 advertised by PE21. As an example, this SR policy is associated with a segment list that includes the prefix SID of a transit router in the backbone (PE50) and the prefix SID of the intended egress PE (PE21).

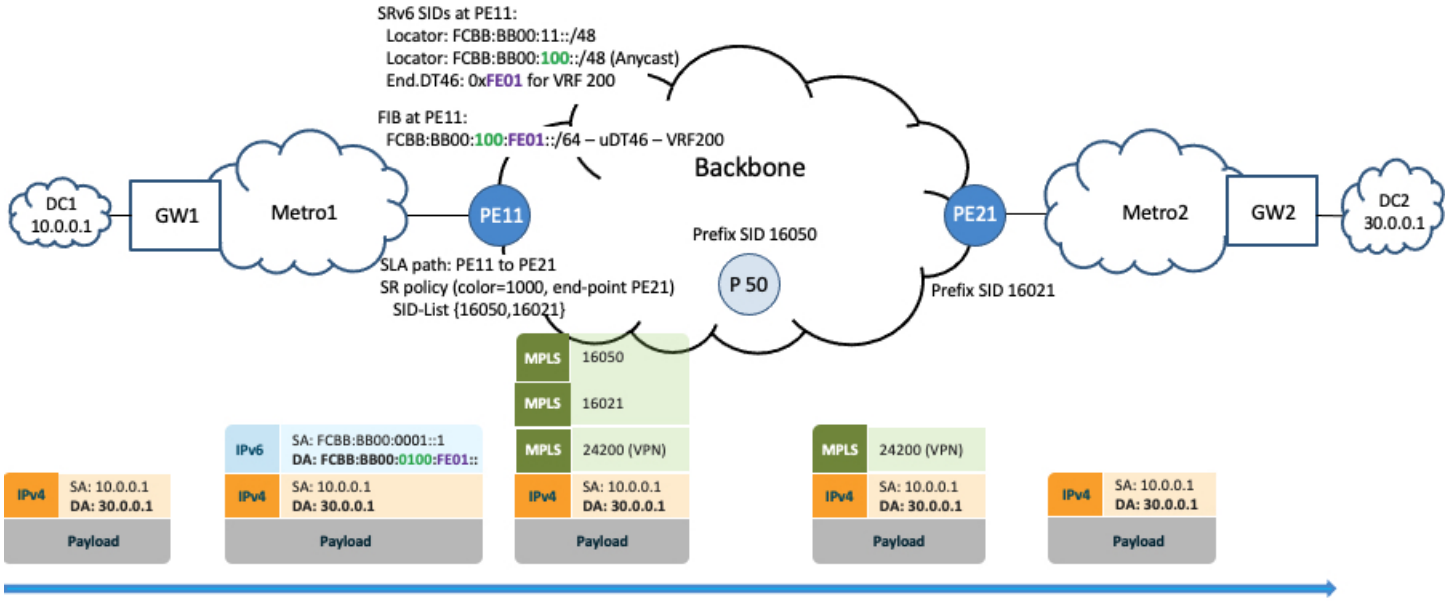
Figure 16: Best Effort Traffic (Traffic Direction – DC1 to DC2)



Traffic arriving at GW1 destined for 20.0.0.1 in DC2 is encapsulated into an outer IPv6 header with a destination address of FCBB:BB00:100:FE01::. This address is composed of the Anycast locator at PE11 and the explicit End.DT46 function value of VRF 200. Any transit nodes in the Metro1 domain simply perform a longest-prefix-match lookup for prefix FCBB:BB00:100::/48 and forward the packet along the shortest path to PE11.

PE11 removes the SRv6 encapsulation and looks up prefix 20.0.0.1 in VPN table of VRF 200. PE11 imposes the VPN label and the prefix SID of PE21 in order to steer traffic over the native LSP path.

Figure 17: SLA Traffic (Traffic Direction – DC1 to DC2)



Traffic arriving at GW1 destined for 30.0.0.1 in DC2 is encapsulated into an outer IPv6 header with a destination address of FCBB:BB00:100:FE01::. As in the previous case, this address is composed of the Anycast locator at PE11 and the explicit End.DT46 function value of VRF 200. Any transit nodes in the Metro1 domain simply perform a longest-prefix-match lookup for prefix FCBB:BB00:100::/48 and forward the packet along the shortest path to PE11.

PE11 removes the SRv6 encapsulation and looks up prefix 30.0.0.1 in VPN table of VRF 200. PE11 imposes the VPN label and transport labels corresponding to the segment list of the SR policy (1000, PE21) in order to steer traffic over the path associated with the SR policy.

Configuration for SRv6 PE Lite Node 11

Configure SRv6:

```
segment-routing
 srv6
  locators
   locator myLoc1
    micro-segment behavior unode psp-usd
    prefix fcbb:bb00:11::/48
   !
   locator myLocAnycast
    anycast
    micro-segment behavior unode psp-usd
    prefix fcbb:bb00:100::/48
   !
  !
 !
!
```

Configure IGP instance in core with SR MPLS enabled and prefix SID assigned to Loopback0:

```
router isis core
 address-family ipv4 unicast
  metric-style wide level 1
  router-id Loopback0
  segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16011
  !
 !
!
```

Configure interface Loopback0:

```
interface Loopback0
 ipv4 address 1.1.1.11 255.255.255.255
!
```

Configure the SR policy:

```
segment-routing
 traffic-eng
  segment-list sample-SIDLIST
   index 10 mpls label 16050
   index 20 mpls label 16021
  !
  policy pol-sla-to_21
   color 1000 end-point ipv4 1.1.1.21
   candidate-paths
    preference 100
     explicit segment-list sample-SIDLIST
```

```
     !
    !
   !
  !
 !
!
```

Configure the VRF (dual-stack IPv4/IPv6):

```
vrf VRF-200
 address-family ipv4 unicast
  import route-target
   1:200
  !
  export route-policy SET-COLOR-1000
  export route-target
   1:200
  !
 !
 address-family ipv6 unicast
  import route-target
   1:200
  !
  export route-policy SET-COLOR-1000
  export route-target
   1:200
  !
 !
!
!
extcommunity-set opaque COLOR-1000
  1000
end-set
!
route-policy SET-COLOR-1000
  set extcommunity color COLOR-1000
end-policy
!
```

Configure BGP:

```
router bgp 100
 segment-routing srv6
  locator myLoc1
 !
 address-family vpnv4 unicast
 !
 neighbor 1.1.1.21
  remote-as 100
  address-family vpnv4 unicast
  !
 !
 vrf VRF-200
  rd 200:1
  address-family ipv4 unicast
  !
 !
!
```

Configuring Explicit End.DT46 SRv6 SIDs

Table 14: Feature History Table

Feature Name	Release	Description
Support for End.DT46 SRv6 Endpoint Behavior	Release 7.5.3	This feature adds support for the “Endpoint with decapsulation and specific IP table lookup” SRv6 end-point behavior (End.DT46). The End.DT46 behavior is used for dual-stack L3VPNs. This behavior is equivalent to the single per-VRF VPN label (for IPv4 and IPv6) in MPLS.
Support for Explicit End.DT46 SRv6 SIDs	Release 7.5.3	This feature allows you to configure explicit SIDs associated with SRv6-based L3VPN/Internet BGP services. In previous releases, these SIDs were only allocated dynamically by BGP. Explicit End.DT46 SRv6 SIDs are persistent over reloads and restarts. The feature introduces these changes: **CLI:** • The **segment-routing srv6 static endpoint sid** *prefix* **behavior end-udt46** command mode is introduced.

Explicit End.DT46 SRv6 SIDs are persistent over reloads and restarts.

Multiple explicit uDT46 IDs allocated from the LIB or W-LIB range can be created under the same SRv6 locator. Each ID is uniquely associated to a VRF.

Note See GIB and LIB – IOS-XR Implementation, on page 18 for information about the LIB and W-LIB

Example: Explicit uDT46 (LIB)

- Locator: fcbb:bb**00**:**11**::/48
- Explicit functions: fe0a (VRF-A), fe0b (VRF-B), fe0c (VRF-C)
  - fcbb:bb00:**11**:fe0a::/64 — Explicit 16-bit DT46 function from LIB for VRF-A
  - fcbb:bb00:**11**:fe0b::/64 — Explicit 16-bit DT46 function from LIB for VRF-B
  - fcbb:bb00:**11**:fe0c::/64 — Explicit 16-bit DT46 function from LIB for VRF-C

Example: Explicit uDT46 (W-LIB)

- Locator: fcbb:bb**00**:**11**::/48
- Explicit functions: fff7:d (VRF-D), fff7:e (VRF-E), fff7:f (VRF-F)

- fcbb:bb00:**11**:fff7:d::/80 — Explicit 32-bit DT46 function from W-LIB for VRF-D
- fcbb:bb00:**11**:fff7:e::/80 — Explicit 32-bit DT46 function from W-LIB for VRF-E
- fcbb:bb00:**11**:fff7:f::/80 — Explicit 32-bit DT46 function from W-LIB for VRF-F

An explicit uDT46 ID allocated from the LIB or W-LIB range can be associated to the same VRF under multiple SRv6 locators.

This association is useful when a look-up under a given VPN table is desired for a node with multiple locators (for example, unicast and Anycast locators).

The locators can be from the same ID block or different ID blocks:

- When the locators are from the same block, the manual uDT46 IDs for a given VRF must have the same value across locators.
- When the locators are from different blocks, the manual uDT46 IDs for a given VRF could be either the same value or different values.

We recommend using the same function ID across locators since it allows for simpler identification to the associated VRF table.

Example 1: Explicit uDT46 SIDs (LIB) with a common function ID under 2 locators of the same block-id

- Locator 1: fcbb:bb**00**:**10**::/48 (unicast locator)
- Locator 2: fcbb:bb**00**:**100**::/48 (Anycast locator)
- Explicit function: fe0a
- VRF lookup: VRF-A

```
fcbb:bb00:10:fe0a::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-A
fcbb:bb00:100:fe0a::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-A
```

Example 2: Explicit uDT46 SIDs (LIB) with a common function ID under 2 locators of different block-id

- Locator 1: fcbb:bb**00**:**11**::/48 (unicast locator – algo0)
- Locator 2: fcbb:bb**01**:**11**::/48 (unicast locator – algo128)
- Explicit function: fe0b
- VRF lookup: VRF-B

```
fcbb:bb00:11:fe0b::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-B
fcbb:bb01:11:fe0b::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-B
```

Example 3: Explicit uDT46 SIDs (W-LIB) with a common function ID under 2 locators of the same block-id

- Locator 1: fcbb:bb**00**:**10**::/48 (unicast locator)
- Locator 2: fcbb:bb**00**:**100**::/48 (Anycast locator)
- Explicit function: fff7:d
- VRF lookup: VRF-D

```
fcbb:bb00:11:fff7:d::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-D
fcbb:bb00:100:fff7:d::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-D
```

Example 4: Explicit uDT46 SIDs (W-LIB) with a common function ID under 2 locators of different block-id

- Locator 1: fcbb:bb**00**:**11**::/48 (unicast locator – algo0)
- Locator 2: fcbb:bb**01**:**11**::/48 (unicast locator – algo128)
- Explicit function: fff7:e
- VRF lookup: VRF-E

```
fcbb:bb00:11:fff7:e::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-E
fcbb:bb01:11:fff7:e::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-E
```

Configuration

To configure explicit uDT46 IDs allocated from the LIB or W-LIB range, use the **segment-routing srv6 static endpoint sid** *prefix* **behavior end-udt46** command.

Use the **allocation-context vrf** *vrf-name* command to associate an explicit uDT46 ID allocated from the LIB or W-LIB to a VRF. Use the **forwarding** keyword if VRF-Lite (the deployment of VRFs without BGP/MPLS) is enabled.

```
RP/0/RP0/CPU0:ios(config)# segment-routing
RP/0/RP0/CPU0:ios(config-sr)# srv6
RP/0/RP0/CPU0:ios(config-srv6)# static
RP/0/RP0/CPU0:ios(config-srv6-static)# endpoint
RP/0/RP0/CPU0:ios(config-srv6-static-endpoint)# sid fcbb:bb00:10:fe0a:: behavior end-udt46
RP/0/RP0/CPU0:ios(config-srv6-static-sid)# allocation-context vrf VRF-A
RP/0/RP0/CPU0:ios(config-srv6-static-sid)# forwarding
RP/0/RP0/CPU0:ios(config-srv6-static-sid)# exit
RP/0/RP0/CPU0:ios(config-srv6-static-endpoint)# sid fcbb:bb00:11:fff7:d:: behavior end-udt46
RP/0/RP0/CPU0:ios(config-srv6-static-sid)# allocation-context vrf VRF-D
```

Running Config

```
segment-routing
 srv6
  static
   endpoint
    sid fcbb:bb00:10:fe0a:: behavior end-udt46
     allocation-context vrf VRF-A
     forwarding
     !
    !
    sid fcbb:bb00:11:fff7:d:: behavior end-udt46
     allocation-context vrf VRF-D
```

Configuring Explicit SRv6 uSID Allocation Start Range

You can modify the start of the range of IDs available in the explicit LIB and explicit W-LIB.

To modify the start value for the explicit LIB, use the following command:

- **segment-routing srv6 formats format usid-f3216 usid local-id-block explicit start** *lib-start-value*, where *lib-start-value* is from 0xE064 to 0xFEFF

Note When you increase the size of the explicit LIB range, you effectively decrease the number of available IDs in the dynamic LIB range. For example, if you configure the explicit LIB starting value to 0xE064, the dynamic LIB range is 0xE000 to 0xE063 (100 IDs).

To modify the start value for the explicit W-LIB, use the following command:

- **segment-routing srv6 formats format usid-f3216 usid wide-local-id-block explicit start** *wlib -start-value*, where *wlib-start-value* is from 0xFFF0 to 0xFFF7

Example

Use the **show segment-routing srv6 manager** command to display the default LIB and W-LIB start values:

```
RP/0/RP0/CPU0:ios# show segment-routing srv6 manager
Fri Sep  9 18:30:06.503 UTC
Parameters:
  SRv6 Enabled: No
  SRv6 Operational Mode: None
  Encapsulation:
    Source Address:
      Configured: ::
      Default: ::
    Hop-Limit: Default
    Traffic-class: Default
  SID Formats:
    f3216 <32B/16NFA> (2)
      uSID LIB Range:
        LIB Start   : 0xe000
        ELIB Start  : 0xfe00
      uSID WLIB Range:
        EWLIB Start : 0xfff7

. . .
```

The following example shows how to modify the start of the range for explicit LIB and W-LIB:

```
RP/0/RP0/CPU0:ios(config)# segment-routing
RP/0/RP0/CPU0:ios(config-sr)# srv6
RP/0/RP0/CPU0:ios(config-srv6)# formats
RP/0/RP0/CPU0:ios(config-srv6-fmts)# format usid-f3216
RP/0/RP0/CPU0:ios(config-srv6-fmt)# usid local-id-block explicit start 0xE064
RP/0/RP0/CPU0:ios(config-srv6-fmt)# usid wide-local-id-block explicit start 0xFFF0
```

Running Config

```
segment-routing
 srv6
  formats
   format usid-f3216
    usid local-id-block explicit start 0xe064
    usid wide-local-id-block explicit start 0xfff0
   !
  !
 !
!
```

Use the **show segment-routing srv6 manager** command to display the configured explicit LIB and W-LIB starting values:

```
RP/0/RP0/CPU0:ios# show segment-routing srv6 manager
Fri Sep  9 18:31:06.033 UTC
Parameters:
  SRv6 Enabled: Yes
  SRv6 Operational Mode: None
  Encapsulation:
    Source Address:
      Configured: ::
      Default: ::
    Hop-Limit: Default
    Traffic-class: Default
  SID Formats:
    f3216 <32B/16NFA> (2)
      uSID LIB Range:
        LIB Start   : 0xe000
        ELIB Start  : 0xe064 (configured)
      uSID WLIB Range:
        EWLIB Start : 0xfff0 (configured)

. . .
```

Configure Seamless Bidirectional Forwarding Detection

Table 15: Feature History Table

Feature Name	Release	Description
Support for Cisco 8000 routers as Seamless BFD reflector	Release 7.5.3	This feature introduces support for Cisco 8000 series routers to act as a Seamless Bidirectional Forwarding Detection (SBFD) reflector. Seamless BFD (SBFD) eliminates many negotiation aspects and thereby provides a simplified approach to using BFD. Benefits of SBFD include quick provisioning, improved control, and flexibility for network nodes that initiate path monitoring. The SBFD 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.

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.

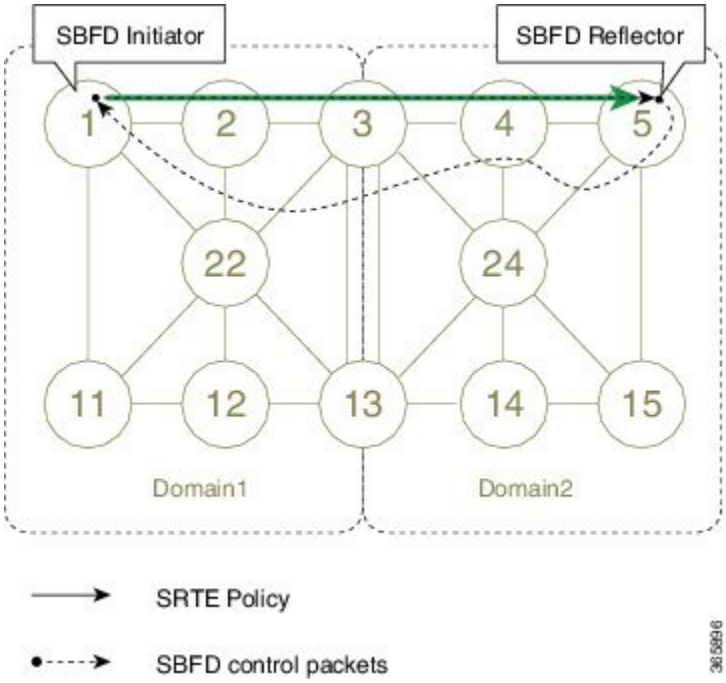
In BFD, each end of the connection maintains a BFD state and transmits packets periodically over a forwarding path. Seamless BFD (SBFD) is unidirectional, resulting in faster session activation than BFD. The BFD state and client context is maintained on the head-end (initiator) only. The tail-end (reflector) 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 18: 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 SR-TE 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.

Note Cisco 8000 series routers support reflector mode only.

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 Discriminator must be unique within an administrative domain. If multiple network nodes allocate the same SBFD Discriminator value, then SBFD Control packets falsely terminating on a wrong network node can result in a Reflector BFD session generating a response back because of a matching Your Discriminator value.

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.

Configuring 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.

Enable MPLS OAM

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)#
```

Configure Local Discriminators on the Reflector

Use the **sbfd local-discriminator** {*ipv4-address* | *32-bit-value* | **dynamic** | **interface** *interface*} command to configure a local discriminator for the reflector. The following example shows the different ways to configure a local discriminator.

```
Router_5(config)# sbfd
Router_5(config-sbfd)# local-discriminator 10.1.1.5
Router_5(config-sbfd)# local-discriminator 987654321
Router_5(config-sbfd)# local-discriminator dynamic
Router_5(config-sbfd)# local-discriminator interface Loopback0
```

Running Config

```
mpls oam
!
sbfd
 local-discriminator 10.1.1.5
 local-discriminator 987654321
 local-discriminator dynamic
 local-discriminator interface Loopback0
!
```

Verifying SBFD Reflector

```
Router_5# show bfd target-identifier local

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

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


Router_5# show bfd reflector info detail location 0/0/CPU0
```

```
Local Discr     : 2147483649
Remote Discr    : 65576
Source Address  : 1.1.1.1
Last DOWN received Time : (NA)
Last Rx packets timestamps before DOWN
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ]
Last Tx packets timestamps before DOWN
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ] [NA                 ] [NA                 ]
  [NA                 ] [NA                 ]
Last UP sent Time : (Jun  7 14:59:34.763)
Last recent Rx packets timestamps:
  [Jun 7 15:00:18.653 ] [Jun 7 15:00:18.751 ] [Jun 7 15:00:18.837 ] [Jun 7 15:00:18.927 ]
  [Jun 7 15:00:18.085 ] [Jun 7 15:00:18.185 ] [Jun 7 15:00:18.274 ] [Jun 7 15:00:18.372 ]
  [Jun 7 15:00:18.464 ] [Jun 7 15:00:18.562 ]
Last recent Tx packets timestamps:
  [Jun 7 15:00:18.653 ] [Jun 7 15:00:18.751 ] [Jun 7 15:00:18.837 ] [Jun 7 15:00:18.927 ]
  [Jun 7 15:00:18.085 ] [Jun 7 15:00:18.185 ] [Jun 7 15:00:18.274 ] [Jun 7 15:00:18.372 ]
  [Jun 7 15:00:18.464 ] [Jun 7 15:00:18.563 ]
```

SRv6 SID Information in BGP-LS Reporting

BGP Link-State (BGP-LS) is used to report the topology of the domain using nodes, links, and prefixes. This feature adds the capability to report SRv6 Segment Identifier (SID) Network Layer Reachability Information (NLRI).

The following NLRI has been added to the BGP-LS protocol to support SRv6:

- Node NLRI: SRv6 Capabilities, SRv6 MSD types
- Link NLRI: End.X, LAN End.X, and SRv6 MSD types
- Prefix NLRI: SRv6 Locator
- SRv6 SID NLRI (for SIDs associated with the node): Endpoint Function, BGP-EPE Peer Node/Set

This example shows how to distribute IS-IS SRv6 link-state data using BGP-LS:

```
Router(config)# router isis 200
Router(config-isis)# distribute link-state instance-id 200
```

Note It is still possible to ping or trace a SID:

- **ping** B:k:F::
- **traceroute** B:k:F::

It is possible to use a list of packed carriers to ping or trace a SID, to ping or trace route, use <destination SID> via srv6-carriers <list of packed carriers>

Full-Replace Migration to SRv6 Micro-SID

Table 16: Feature History Table

Feature Name	Release	Description
Full-Replace Migration to SRv6 Micro-SID	Release 7.8.1	This feature enables migration of existing SRv6 SID format1 to SRv6 Micro-SIDs (f3216) formats. Earlier, only one format was supported at a time, and you had to choose either format1 or Micro-SID format for the deployment of services. Migration from Full-length SIDs to SRv6 Micro-SIDs was not possible.

During the Full-Replace migration, both underlay and services are migrated from format1 to f3216. The underlay migration is done using the *Ship in the night* strategy, where updates into your environment are incremental, thereby phasing out your existing transport protocols when ready. This method minimizes the service disruption, and is recommended for seamless migration. The services migration is done using *swap* procedures, where the incoming transport label is swapped with an outgoing transport label.

The format1 to f3216 migration is seamless, requires minimal configurations, and no IETF signaling extensions. The migration enables preference of Micro-SID f3216 over format1, and minimizes traffic drop with faster convergence.

Cisco 8000 series routers support the following configurations for migration from format1 to Micro-SIDs:

- IS-IS underlay (TILFA, uLoop, FlexAlgo)

The following modes are supported in the context of migration:

- **Base**: SRv6 classic with format1 only.
- **Dual**: SRv6 classic with format1 and SRv6 Micro-SID with f3216 will both coexist.
- **f3216**: Micro-segment format. f3216 represents the format 3216, which is 32-bit block and 16-bit IDs.

The migration starts with SRv6 base format1, and ends with SRv6 uSID f3216. The migration states in detail are:

The migration process involves the following steps:

1. **Prepare for migration**: Upgrade the network nodes to an image that is Micro-SID f3216 capable, and allows the coexistence of format1 and f3216.
2. **Migrate the underlay to Micro-SID**: Enable IS-IS as an underlay protocol on PE nodes. The IS-IS configuration adds f3216 locators to format1 locators. Both format1 and f3216 endpoint SIDs are allocated, installed, and announced during this stage. f3216 is the preferred option over format1 for underlay paths.

   The IS-IS SR headends provide faster convergence to Micro-SID. Faster convergence to f3216 is done on the per-prefix per-path level, does not need any new CLI, and avoids packet drops. The format1 locators

are removed after underlay traffic convergence to f3216 on all nodes. The format1 locators are unconfigured from IS-IS, and deleted from SRv6.

At the end of this step, the migration status of the following P Nodes are:

- Locator reachability: f3216 only
- Underlay endpoint/headends: f3216 only

At the end of this step, the migration status of the following PE Nodes are:

- Locator reachability: format1 and f3216
- Underlay endpoint/headends: f3216 only
- Overlay endpoint/headends: format1

3. **Migrate the overlay to Micro-SID**: Enables overlay f3216 under BGP and EVPN on all PE nodes. The BGP and EVPN configuration replaces format1 by f3216 locators. During this stage, the f3216 Micro-SIDs are allocated, installed, and announced, while the format1 SIDs are deallocated, uninstalled, and withdrawn.

   The format1 locators are removed after overlay traffic convergence to f3216 on all nodes. The format1 locators are unconfigured from BGP and EVPN, and deleted from SRv6.

   For a transient period, BGP and EVPN might have some paths with format1 and some with f3216.

   At the end of this step, the migration status of the following are:

   - For P/PE Nodes:
     - Locator reachability: f3216 only
     - Underlay endpoint/headends: f3216 only
     - Overlay endpoint/headends: f3216 only

The migration starts with SRv6 base format1, and ends with SRv6 Micro-SID f3216. The migration states are:

1. **Initial state**: This is the early migration state of a deployment, for the supported features. This state comprises SRv6 base with format1.

   This example shows the initial state of migration with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc0
Router(config-srv6-locators)# prefix f1bb:bbbb:bb00:0001::/64
```

   This example shows the initial state of migration with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc0
```

   This example shows the initial state of migration with SRv6 and BGP/EVPN:

```
Router(config)# router bgp 100
```

```
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myLoc0
```

2. **In-migration state**: The migration procedures are initiated, and are in progress. This state comprises SRv6 in dual mode (base with format1, and Micro-SID with f3216).

   This example shows the in-migration state with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc0
Router(config-srv6-locators)# prefix f1bb:bbbb:bb00:0001::/64
Router(config-srv6-locators)# delayed-delete
Router(config-srv6-locators)# locator myuLoc0
Router(config-srv6-locators)# micro-segment behavior unode psp-usd
Router(config-srv6-locators)# prefix fcbb:bb00:0001::/48
```

   This example shows the in-migration state with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc0
Router(config-isis-srv6)# locator myuLoc0
```

   This example shows the in-migration state with SRv6 and BGP/EVPN:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myuLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myuLoc0
```

3. **End state**: This is the state of deployment at the end of the migration. At the end state, you can update the network and add new features. The Full-Replace migration end state can be of two modes:

   - **Full-Replace**: Both underlay and overlay are migrated to Micro-SID f3216. Full-Replace is the Cisco recommended migration type.
   - **uF1**: Underlay migrated to Micro-SID f3216, overlay remains format1. The uF1 migration is a transient state of the Full-Replace migration type.

   This example shows the end state with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myuLoc0
Router(config-srv6-locators)# micro-segment behavior unode psp-usd
Router(config-srv6-locators)# prefix fcbb:bb00:0001::/48
```

This example shows the end state with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myuLoc0
```

This example shows the end state with SRv6 and BGP/EVPN:

```
Router(config)# router  bgp 100
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myuLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myuLoc0
```

Run the following command to check the result of migration, as shown in the example:

```
RP/0/RSP0/CPU0:Router# sh route ipv6 fc00:cc30:600:e004:: detail
Wed Nov 10 18:57:56.645 UTC

Routing entry for fc00:cc30:600::/48
    Known via "isis 2", distance 115, metric 141, SRv6-locator, type level-2
    Installed Nov 2 18:56:55.718 for 00:01:01
    Routing Descriptor Blocks
      fe80::232:17ff:fec3:58c0, from 7511::1, via TenGigE0/0/0/16.1, Protected
        Route metric is 141
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:1  Path ref count:0
        NHID:0x20006(Ref:193)
        Backup path id:65
      fe80::226:80ff:fe36:7c01, from 7511::1, via TenGigE1/0/9/1.1, Backup (TI-LFA)
        Repair Node(s): 3888::1
        Route metric is 251
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:65  Path ref count:1
        NHID:0x20007(Ref:163)
        SRv6 Headend:H.Insert.Red [f3216], SID-list {fc00:cc30:700::}
 Route version is 0x0 (8)
 No local label
 IP Precedence: Not Set
 QoS Group ID: Not Set
 Flow-tag: Not Set
 Fwd-Class: Not Set
 Route Priority:RIB_PRIORITY_NON_RECURSIVE_LOW (8) SVD Type RIB_SVD_TYPE_LOCAL
 Download Priority 2, Download Version 261731
 No advertising protos.
```

SRv6 Traffic Accounting

Table 17: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Traffic Accounting	Release 7.10.1	You can now enable the router to record the number of packets and bytes transmitted on a specific egress interface for IPv6 traffic using the SRv6 locator counter. You can use this data to create deterministic data tools to anticipate and plan for future capacity planning solutions. This feature introduces or modifies the following changes: **CLI**: • **accounting prefixes ipv6 mode per-prefix per-nexthop srv6-locators** **YANG Data Models:** • `Cisco-IOS-XR-accounting-cfg` • `Cisco-IOS-XR-fib-common-oper.yang` (see GitHub, YANG Data Models Navigator)

SRv6 traffic accounting is an integral part of today's network for planning and forecasting traffic. Traffic accounting is the volume of aggregated traffic flows that enter, traverse, and leave the network in a given time. Traffic accounting is a solution to monitor the traffic that helps to measure traffic flows and record how much customer traffic is passing through the SR network.

To design a network topology and meet the defined Service-Level Agreement (SLA), capacity planning becomes essential for forecasting traffic load and failures. A complete view of the traffic in your network enables you to anticipate common failures, and provision for network expansion.

You can now monitor traffic on an ingress node of a domain that is SRv6 encapsulated towards an egress node of the domain. The traffic is recorded at the source using the per-locator, per-egress-interface (LOC.INT.E) counter, which is the locator per interface at egress to account the traffic. For a given locator (L) and interface (I), the router counts the number of packets and bytes for the traffic transmitted on the interface (I) with a destination address (DA) matching the locator L.

When this feature is enabled on routers, all traffic passing through the routers are accounted. These counters are periodically streamed through telemetry and you can retrieve the counters at any point.

To enable traffic accounting on PE and P routers, use the **accounting prefixes ipv6 mode per-prefix** command. You can retrieve the number of packets transmitted and received on the specific interface of a PE or P routers by using the following telemetry:

```
Cisco-IOS-XR-fib-common-oper:oef-accounting/vrfs/vrf[vrf-name='default']/afis/afi[afi-type=ipv6]/pfx/srv6locs/srv6loc
```

Benefits

Monitoring the traffic provides numerous benefits, and here are a few:

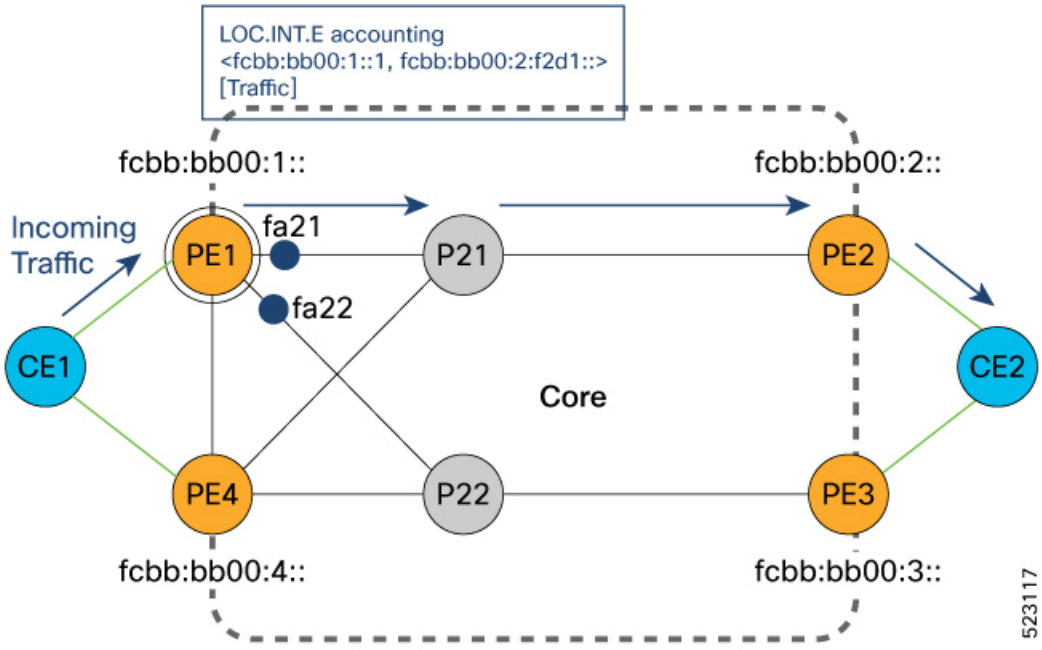
- To optimize network utilization and achieve a balance between underutilized and overutilized paths.
- To plan and optimize network capacity and avoid congestion.
- To plan the service provisioning and choose the right path and create an optimized backup path (for using SRLG's affinity, and so on).

Understanding SRv6 Locator Counters

Let’s understand this feature with the following topology:

Consider the topology where traffic is passing from CE1 to CE2 through PE1. The traffic sent and received from CE1 is considered as the external traffic. The traffic from PE4 destined to PE2 is considered as the internal traffic.

Figure 19: Sample Topology for SRv6 Traffic Accounting



PE1 learns CE2 reachability through PE2. Consider PE1 has ECMP paths via P21 and P22 to reach PE2.

- When traffic reaches PE1, PE1 imposes traffic with the PE2 locator fcbb:bb00:2::.
- SRv6 traffic accounting LOC.INT.E is per prefix per egress interface accounting.

When traffic exits the PE1 interface (fa21) through P21, PE1 keeps the count of this traffic that is sent. Also, when traffic exits the PE1 interface (fa22) through P22, PE1 keeps the count of this traffic that is sent. The traffic is accounted irrespective of the path PE1 takes to send traffic.

Here is the SRv6 label of the outgoing traffic for PE2:

```
<fcbb:bb00:1::1, fcbb:bb00:2:f2d1::> [CUSTTraffic]
```

- When the next set of packets are received and passed through PE1, the counters are incremented on fa21or fa22 interface based on the path the traffic sent through PE2.

The traffic from PE4 to PE1 is considered as internal traffic.

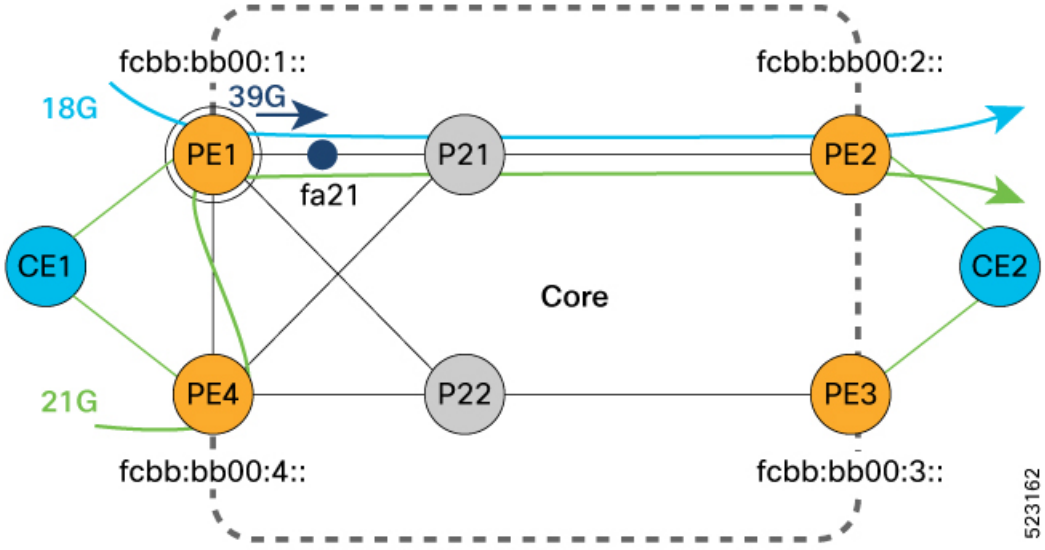
- When traffic is sent from PE4 to PE2 through PE1, PE4 imposes the traffic with the PE2 locator ID fcbb:bb00:2::. The traffic count is recorded at PE4 for this locator ID.
- When traffic reaches PE1, it looks for the PE2 locator ID and keeps the traffic count at PE1 when the traffic exit the fa21 interface.

Let's see how the SRv6 traffic is calculated using the demand matrix.

The Demand Matrix (DM) also known as a traffic matrix is a representation of the amount of data transmitted between every pair of routers. Each cell in the DM represents a traffic volume from one router to another. DM gives a complete view of the traffic in your network.

In the topology, the amount of external traffic destined for PE2 is a combination of external and internal traffic.

Figure 20: SampleTopology for Demand Matrix



- The traffic transmitted from PE1 is marked in **blue**.
- The traffic transmitted from PE4 is marked as in **green**.

The external traffic that PE2 receives is equal to the total traffic sent out from PE1 minus the received internal traffic.

```
External traffic to PE2
```

```
= (Total traffic sent out from PE1) - (Internal traffic received by PE1)
= (sum of all Loc.int.E counters on PE1) - ( sum of the Loc.int.E counters of all neighbors
 of PE1)
```

Let's try to calculate with this example.

- PE1 transmits a total of 39 gigabits per second towards PE2.
- PE1 receives 21 gigabits per second of internal traffic from PE4.
- PE1 receives 0 gigabits per second from P21 and P22.

You can calculate the external traffic to PE2 as follows:

```
External traffic to PE2
= (sum of all Loc.int.E counters on PE1) - ( sum of the Loc.int.E counters of all neighbors
 of PE1)
= 39 gigabits per second - (21 + 0 + 0) gigabits per second
= 18 gigabits per second external traffic
```

So, PE2 recieves 18 gigabits per second external traffic from PE1.

The calculation for external traffic for routers follows a similar approach. Let's see few examples in the following demand matrix.

Table 18: Demand Matrix showing traffic transmitted from PE1 and PE4 to PE2

From/To	PE1	PE2
PE1	NA	39 - (21 + 0 +0) = 18 gigabits per second
PE4	21- (18 + 0 + 0) = 3 gigabits per second	39 - (18 + 0 + 0) = 21 gigabits per second

Usage Guidelines and Limitations

Supported Traffic Types

- IPv6 packets.
- SRv6 packets with the local SID as the top SID.
  - If the top SID is a local uN, traffic is counted against the remote locator prefix of the next SID.
  - Traffic is not counted if the top SID is a local uA.
- SRv6 VPNv4
- SRv6 VPNv6
- SRv6 INETv4
- SRv6 INETv6

Limitations

- Supports a minimum telemetry pull interval of 30 seconds.

- Ethernet header is considered for bytes accounting.
- SRv6 traffic accounting does not count locally generated control plane packets such as ping to the remote locator.
- Packets aren't counted if the local uA is the top SID.
- SRv6 traffic accounting is not supported with SRv6 TE policy.
- No additional MIBs are supported to retrieve SRv6 traffic statistics. We recommend to use telemetry through the newly added sensor-path in `Cisco-IOS-XR-fib-common-oper` to retrieve these statistics.

Configure SRv6 Traffic Accounting

Before you begin ensure that you enable SRv6 and its services.

Configuration Example

To enable SRv6 traffic accounting:

```
Router#configure
Router(config)#accounting
Router(config-acct)#prefixes ipv6 mode per-prefix per-nexthop srv6-locators
Router(config-acct)#commit
```

Running Configuration

```
Router#show run
accounting
 prefixes
  ipv6
   mode per-prefix per-nexthop srv6-locators
  !
 !
!
```

Verification

Verify the Stats ID allocated for remote locator. The following example shows the SRv6 locator ID and the stats ID allocated for the prefixes with the locator ID.

```
Router#show route ipv6  fccc:cc00:1:: detail

Routing entry for fccc:cc00:1::/48
  Known via "isis 100", distance 115, metric 101, SRv6-locator, type level-1  <==========
locator flag
  Installed Jun  1 11:59:10.941 for 00:00:04
  Routing Descriptor Blocks
    fe80::1, from 1::1, via Bundle-Ether1201, Protected, ECMP-Backup (Local-LFA)
      Route metric is 101
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:2       Path ref count:1
      NHID: 0x2001b (Ref: 79)
      Stats-NHID: 0x2001c (Ref: 6)
```

```
     Backup path id:1
   fe80::1, from 1::1, via TenGigE0/1/0/5/2, Protected, ECMP-Backup (Local-LFA)
     Route metric is 101
     Label: None
     Tunnel ID: None
     Binding Label: None
     Extended communities count: 0
     Path id:1       Path ref count:1
     NHID: 0x2001a (Ref: 79)
     Stats-NHID: 0x2001d (Ref: 6)    <=========  Stats-NHID is allocated for prefixes with
 locator flag
     Backup path id:2
  Route version is 0x68 (104)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_NON_RECURSIVE_LOW (8) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 2, Download Version 39779
  No advertising protos.
```

Configuring Telemetry Data

Configure the sensory path to retrieve the accounting data using telemetry:

```
Router#configure
Router(config)#grpc
Router(config-grpc)#port 57400
Router(config-grpc)#no-tls
Router(config-grpc)#commit
Router(config-grpc)#exit
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group s1
Router(config-model-driven-snsr-grp)#sensor-path
Cisco-IOS-XR-fib-common-oper:cef-accounting/vrfs/vrf[vrf-name='default']/af$
Router(config-model-driven-snsr-grp)#exit
Router(config-model-driven)#subscription sub1
Router(config-model-driven-subs)#sensor-group-id s1 sample-interval 30000
Router(config-model-driven-subs)#commit
Router(config-model-driven-subs)#root
Router(config)#exit
Router#
```

Running Configuration for Configuring Telemetry Data

The following shows the show running configuration:

```
Router#show run
grpc
 port 57400
 no-tls
!
telemetry model-driven
 sensor-group s1
  sensor-path
Cisco-IOS-XR-fib-common-oper:cef-accounting/vrfs/vrf[vrf-name='default']/afis/afi[afi-type=ipv6]/pfx/srv6locs/srv6loc

 !
 subscription sub1
  sensor-group-id s1 sample-interval 30000
```

```
 !
!
```

Verification for Configuring Telemetry Data

Verify the counters using the telemetry data. The following example shows the accounting data with the number of packets and the bytes transmitted through the interface.

```
{
"Cisco-IOS-XR-fib-common-oper:cef-accounting": {
  "vrfs": {
   "vrf": [
    {
     "vrf-name": "default",
     "afis": {
      "afi": [
       {
        "afi-type": "ipv6",
        "pfx": {
         "srv6locs": {
          "srv6loc": [
           {
            "ipv6-address": " fccc:cc00:1::",
            "prefix-length": 48,
            "ipv6-prefix": " fccc:cc00:1::",
            "ipv6-prefix-length": 48,
            "accounting-information": [
             {
              "number-of-tx-packets": "1500000",          <=========  Accounting data
              "number-of-tx-bytes": "378000000",          <=========  Accounting data
              "path-index": 0,
              "outgoing-interface": "Bundle-Ether1201",
              "nexthop-addr": "fe80::2/128"
             },
             {
              "number-of-tx-packets": "1000000",          <=========  Accounting data
              "number-of-tx-bytes": "252000000",          <=========  Accounting data
              "path-index": 1,
              "outgoing-interface": "TenGigE0/0/0/22",
              "nexthop-addr": "fe80::2/128"
             }
            ],
            "total-number-of-packets-switched": "2500000",
            "total-number-of-bytes-switched": "630000000"
           }
          ]
         }
        }
       }
      ]
     }
    }
   ]
  }
}
}
```

Run **sh cef ipv6 accounting** command to display the packets per bytes:

```
Router#sh cef ipv6 accounting
fccc:cc00:33::/48
```

```
Accounting: 0/0 packets/bytes output (per-prefix-per-path mode)
 via fe80::2/128, Bundle-Ether1201
  path-idx 0
  next hop fe80::2/128
  Accounting: 0/0 packets/bytes output
fccc:cc05:2::/48
Accounting: 0/0 packets/bytes output (per-prefix-per-path mode)
 via fe80::2/128, Bundle-Ether1201
  path-idx 0
  next hop fe80::2/128
  Accounting: 0/0 packets/bytes output
fccc:cc3e:2::/48
Accounting: 0/0 packets/bytes output (per-prefix-per-path mode)
 via fe80::2/128, Bundle-Ether1201
  path-idx 0
  next hop fe80::2/128
  Accounting: 0/0 packets/bytes output
fccc:cc3e:3::/48
Accounting: 0/0 packets/bytes output (per-prefix-per-path mode)
 via fe80::2/128, Bundle-Ether1201
  path-idx 0
  next hop fe80::2/128
  Accounting: 200000/58400000 packets/bytes output  <<< for prefix fccc:cc3e:3:: we can see
 2lac packets count
```

Path Maximum Transmission Unit (MTU) Discovery for SRv6 Encapsulated Packets

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
Path MTU discovery for SRv6 Packets on Ingress Provider Edge (PE) Routers, Egress (PE) Routers, and P Role Transit Nodes	Release 24.1.1	You can measure and monitor the packet loss information when one SRv6-enabled router sends an oversized packet to another. This functionality enables a router to send an ICMP error message to the source in such cases, prompting the sender to resend a packet whose size is within the MTU value, thus ensuring the packet moves ahead. The feature is critical for SRv6-enabled routers as these routers do not support packet fragmentation. Previously, a router dropped oversized packets without notifying the source, resulting in packet loss. The **hw-module** configuration is not required, this feature is enabled by default.

Earlier, routers did not account for the SRv6 encapsulated packets while checking the MTU of a link along a given data path in the egress core interface. When the path MTU of a link along a given data path was not large enough to accommodate the size of the encapsulated packets from a source, the router silently dropped the packets without notifying the source.

With this configuration, the Ingress PE router, Egress PE routers, and P or Transit nodes with IPv6 roles supports Path MTU discovery for SRv6 encapsulated packets. The router does not drop the packets along a given data path without notifying the source. The router sends an ICMP type 3 or type 2 error message for IPv4 or IPv6 links respectively. The configuration enables the source to learn to use a smaller MTU for packets sent to a destination.

For example, the maximum allowed MTU for an IPv4 link is 1500 bytes. Consider a source that sends an IPv4 packet of size 1480 bytes with an SRv6 encapsulation of 40 bytes. The overall IPv4 packet size is increased to 1520 bytes, which is greater than the maximum MTU allowed on the IPv4 link. In this case, the router sends an ICMP Type 3 error message to the source to request the packet originator to adjust the size of the packet.

We calculate the maximum allowed MTU on IPv4 and IPv6 links using the following formula:

Maximum MTU = Egress Interface MTU + SRv6 Encapsulation Size (maximum 64 bytes) + size of L2 Header

IPV6 IO makes changes to provide SRv6 Path MTU support for **ICMP Too Big** message handing for network inbound packets on P node. ICMP errors are processed as per IETF RFC 4443.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- Ingress
  - The SRv6 uSID (F3216) format supports the feature.
  - The SRv6 Full-length SID format does not support Path MTU discovery.
  - You must configure this feature on the Ingress Provider Edge (PE) router starting from Cisco IOS XR Release 7.11.1.

Note Egress and Provider Core Router or Transit node with IPv6 are supported starting from Cisco IOS XR Release 24.1.1.

  - SRv6 encapsulation supports the following scenarios:
    - IPv4/IPv6 over SRv6
    - SRv6-TE
    - H insert
    - TI-LFA for Single Carrier and Multi Carrier
  - L2 services over SRv6 (L2VPN) do not support the feature.

- Ingress
  - The SRv6 uSID (F3216) format supports the feature.

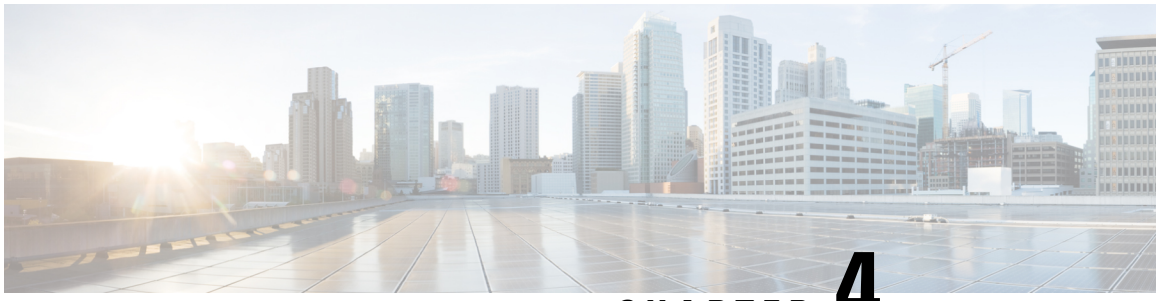
- The SRv6 Full-length SID format does not support Path MTU discovery.
- You must configure this feature on the Ingress Provider Edge (PE) router starting from Cisco IOS XR Release 7.11.1.

Note Egress and P or Transit node with IPv6 are supported starting from Cisco IOS XR Release 24.1.1.

- SRv6 encapsulation supports the following scenarios:
  - IPv4/IPv6 over SRv6
  - SRv6-TE
  - H insert
  - TI-LFA for Single Carrier and Multi Carrier
- L2 services over SRv6 (L2VPN) do not support the feature.

- Egress
  - You can configure this feature on the Egress Provider Edge (PE) router starting from Cisco IOS XR Release 24.1.1.
  - Only supported for the following functions:
    - Decapsulation and specific IPv4 table lookup (DT4), Decapsulation and specific IPv6 table lookup (DT6), and Decapsulation and specific IP table lookup (DT46)
    - Decapsulation and IPv4 cross-connect (DX4) and Decapsulation and IPv6 cross-connect (DX6)
  - Decapsulated packet is punted to PI (after removing the SR6 headers).
  - Does not support Decapsulation and L2 table lookup (DT2) and Decapsulation and L2 cross-connect (DX2) functions (because L2 payload is not a use-case).

- Provider Core Router or Transit Node
  - Path MTU discovery supports P node, that are in IPv6-only, starting from Cisco IOS XR Release 24.1.1.
  - Path MTU discovery now supports SR6 enabled endpoint is a Provider Core Router or a Transit node. This is possible when the destination address of the packet is a SID.



CHAPTER 4

Configure Segment Routing over IPv6 (SRv6) with Full-Length SIDs

Table 20: Feature History Table

Feature Name	Release	Description
SRv6 with Full-Length SIDs	Release 7.5.2	This feature extends Segment Routing support with IPv6 data plane. In a Segment Routing over IPv6 (SRv6) network, an IPv6 address serves as the Segment Identifier (SID). The source router encodes the path to destination as an ordered list of segments (list of IPv6 addresses) in the IPv6 packet using a new header for SRv6 called a Segment Routing Header (SRH). In an SRv6 enabled network, the active segment is indicated by the destination address of the packet, and the next segment is indicated by a pointer in the SRH.

Segment Routing for IPv6 (SRv6) is the implementation of Segment Routing over the IPv6 dataplane.

- Segment Routing over IPv6 Overview, on page 140
- Configuring SRv6 under IS-IS, on page 148
- Configuring SRv6 IS-IS TI-LFA, on page 149
- Configuring SRv6 IS-IS Microloop Avoidance, on page 152
- Configuring SRv6 IS-IS Flexible Algorithm, on page 153
- SRv6 Services: IPv4 L3VPN, on page 155
- SRv6 Services: IPv6 L3VPN, on page 159
- SRv6 Services: L3VPN VPNv4 Active-Standby Redundancy using Port-Active Mode, on page 167
- SRv6 Services: L3VPN VPNv4 Active-Active Redundancy , on page 171
- SRv6 Services: BGP Global IPv6, on page 172
- SRv6 Services: BGP Global IPv6, on page 177
- SRv6 Services on EVPN E-Line, on page 183
- SRv6 SID Information in BGP-LS Reporting, on page 187

Segment Routing over IPv6 Overview

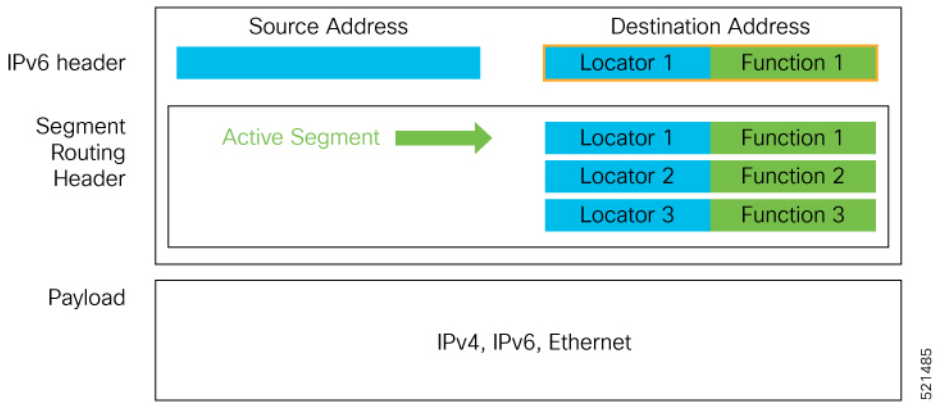
Segment Routing (SR) can be applied on both MPLS and IPv6 data planes. Segment Routing over IPv6 (SRv6) extends Segment Routing support with IPv6 data plane.

In an SR-MPLS enabled network, an MPLS label represents an instruction. The source nodes programs the path to a destination in the packet header as a stack of labels.

SRv6 introduces the Network Programming framework that enables a network operator or an application to specify a packet processing program by encoding a sequence of instructions in the IPv6 packet header. Each instruction is implemented on one or several nodes in the network and identified by an SRv6 Segment Identifier (SID) in the packet. The SRv6 Network Programming framework is defined in IETF RFC 8986 SRv6 Network Programming.

In SRv6, an IPv6 address represents an instruction. SRv6 uses a new type of IPv6 Routing Extension Header, called the Segment Routing Header (SRH), in order to encode an ordered list of instructions. The active segment is indicated by the destination address of the packet, and the next segment is indicated by a pointer in the SRH.

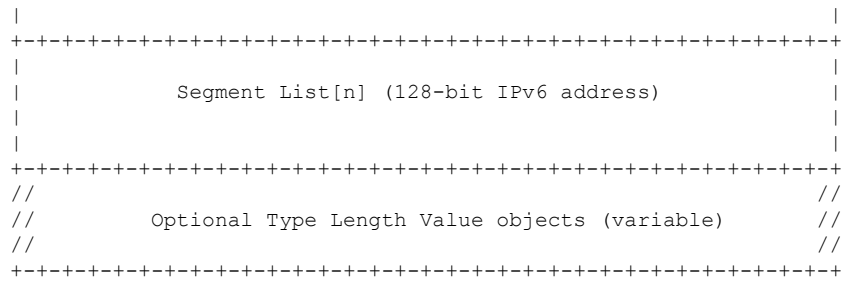
Figure 21: Network Program in the Packet Header



The SRv6 SRH is documented in IETF RFC IPv6 Segment Routing Header (SRH).

The SRH is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Last Entry   |     Flags     |              Tag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128-bit IPv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
                               ...
|                                                               |
```



The following list explains the fields in SRH:

- Next header—Identifies the type of header immediately following the SRH.
- Hdr Ext Len (header extension length)—The length of the SRH in 8-octet units, not including the first 8 octets.
- Segments left—Specifies the number of route segments remaining. That means, the number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
- Last Entry—Contains the index (zero based) of the last element of the segment list.
- Flags— Contains 8 bits of flags.
- Tag—Tag a packet as part of a class or group of packets like packets sharing the same set of properties.
- Segment list—128-bit IPv6 addresses representing the *n*th segment in the segment list. The segment list encoding starts from the last segment of the SR policy (path). That means the first element of the segment list (Segment list [0]) contains the last segment of the SR policy, the second element contains the penultimate segment of the SR policy and so on.

In SRv6, a SID represents a 128-bit value, consisting of the following three parts:

- Locator: This is the first part of the SID with most significant bits and represents an address of a specific SRv6 node.
- Function: This is the portion of the SID that is local to the owner node and designates a specific SRv6 function (network instruction) that is executed locally on a particular node, specified by the locator bits.
- Args: This field is optional and represents optional arguments to the function.

The locator part can be further divided into two parts:

- SID Block: This field is the SRv6 network designator and is a fixed or known address space for an SRv6 domain. This is the most significant bit (MSB) portion of a locator subnet.
- Node Id: This field is the node designator in an SRv6 network and is the least significant bit (LSB) portion of a locator subnet.

SRv6 Node Roles

Each node along the SRv6 packet path has a different functionality:

- Source node—A node that can generate an IPv6 packet with an SRH (an SRv6 packet), or an ingress node that can impose an SRH on an IPv6 packet.

- Transit node—A node along the path of the SRv6 packet (IPv6 packet and SRH). The transit node does not inspect the SRH. The destination address of the IPv6 packet does not correspond to the transit node.
- Endpoint node—A node in the SRv6 domain where the SRv6 segment is terminated. The destination address of the IPv6 packet with an SRH corresponds to the end point node. The segment endpoint node executes the function bound to the SID

SRv6 Head-End Behaviors

The SR Headend with Encapsulation behaviors are documented in the IETF RFC 8986 SRv6 Network Programming.

The SR Headend with Insertion head-end behaviors are documented in the following IETF draft:

https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-net-pgm-insertion/

SRv6 Endpoint Behaviors

The SRv6 endpoint behaviors are documented in the IETF RFC 8986 SRv6 Network Programming.

The following is a subset of defined SRv6 endpoint behaviors that can be associated with a SID.

- End—Endpoint function. The SRv6 instantiation of a Prefix SID [RFC8402].
- End.X—Endpoint with Layer-3 cross-connect. The SRv6 instantiation of an Adj SID [RFC8402].
- End.DX6—Endpoint with decapsulation and IPv6 cross-connect (IPv6-L3VPN - equivalent to per-CE VPN label).
- End.DX4—Endpoint with decapsulation and IPv4 cross-connect (IPv4-L3VPN - equivalent to per-CE VPN label).
- End.DT6—Endpoint with decapsulation and IPv6 table lookup (IPv6-L3VPN - equivalent to per-VRF VPN label).
- End.DT4—Endpoint with decapsulation and IPv4 table lookup (IPv4-L3VPN - equivalent to per-VRF VPN label).
- End.DX2—Endpoint with decapsulation and L2 cross-connect (L2VPN use-case).
- End.B6.Encaps—Endpoint bound to an SRv6 policy with encapsulation. SRv6 instantiation of a Binding SID.
- End.B6.Encaps.RED—End.B6.Encaps with reduced SRH. SRv6 instantiation of a Binding SID.

SRv6 Endpoint Behavior Variants

Depending on how the SRH is handled, different behavior variants are defined for the End and End.X behaviors. The End and End.X behaviors can support these variants, either individually or in combinations.

- **Penultimate Segment Pop (PSP) of the SRH variant**—An SR Segment Endpoint Nodes receive the IPv6 packet with the Destination Address field of the IPv6 Header equal to its SID address.

  A penultimate SR Segment Endpoint Node is one that, as part of the SID processing, copies the last SID from the SRH into the IPv6 Destination Address and decrements the Segments Left value from one to zero.

The PSP operation takes place only at a penultimate SR Segment Endpoint Node and does not happen at non-penultimate endpoint nodes. When a SID of PSP-flavor is processed at a non-penultimate SR Segment Endpoint Node, the PSP behavior is not performed since Segments Left would not be zero.

The SR Segment Endpoint Nodes advertise the SIDs instantiated on them via control plane protocols. A PSP-flavored SID is used by the Source SR Node when it needs to instruct the penultimate SR Segment Endpoint Node listed in the SRH to remove the SRH from the IPv6 header.

- **Ultimate Segment Pop (USP) of the SRH variant**—The SRH processing of the End and End.X behaviors are modified as follows:

  If Segments Left is 0, then:

  1. Update the Next Header field in the preceding header to the Next Header value of the SRH
  2. Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)
  3. Remove the SRH from the IPv6 extension header chain
  4. Proceed to process the next header in the packet

  One of the applications of the USP flavor is when a packet with an SRH is destined to an application on hosts with smartNICs implementing SRv6. The USP flavor is used to remove the consumed SRH from the extension header chain before sending the packet to the host.

- **Ultimate Segment Decapsulation (USD) variant**—The Upper-layer header processing of the End and End.X behaviors are modified as follows:

  - **End** behavior: If the Upper-layer Header type is 41 (IPv6), then:

    1. Remove the outer IPv6 Header with all its extension headers
    2. Submit the packet to the egress IPv6 FIB lookup and transmission to the new destination
    3. Else, if the Upper-layer Header type is 4 (IPv4)
    4. Remove the outer IPv6 Header with all its extension headers
    5. Submit the packet to the egress IPv4 FIB lookup and transmission to the new destination
    6. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

  - **End.X** behavior: If the Upper-layer Header type is 41 (IPv6) or 4 (IPv4), then:

    1. Remove the outer IPv6 Header with all its extension headers
    2. Forward the exposed IP packet to the L3 adjacency J
    3. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

  One of the applications of the USD flavor is the case of TI-LFA in P routers with encapsulation with H.Encaps. The USD flavor allows the last Segment Endpoint Node in the repair path list to decapsulate the IPv6 header added at the TI-LFA Point of Local Repair and forward the inner packet.

Usage Guidelines and Limitations

General Guidelines and Limitations

- SRv6 Underlay support includes:
  - IGP redistribution/leaking between levels
  - Prefix Summarization on ABR routers
  - IS-IS TI-LFA
  - Microloop Avoidance
  - Flex-algo

Platform-Specific Guidelines and Limitations

- SRv6 is supported on the following Cisco 8000 series Q200-based line cards and fixed-port routers:
  - Cisco 8800 with 88-LC0-36FH-M, 88-LC0-36FH, 88-LC0-34H14FH line cards
  - Cisco 8201-32FH
  - Cisco 8102-64H, 8101-32-FH
- SRv6 is not supported on Q100-based line cards and fixed-port routers.
- Egress marking on the outer header during SRv6 encapsulation operations (TI-LFA) is not supported.
- OAM: Ping and traceroute are supported.
- This release supports the following SRv6 behaviors and variants:
  - **Endpoint behaviors:**
    - END with PSP/USD
    - END.X with PSP/USD
    - END.DT4
    - END.DT6
  - **Head-end behaviors:**
    - H.Encap.Red

Encapsulation Capabilities and Parameters

- Cisco 8000 series routers are able to add an SRH with one segment. Therefore, a total of two segments are supported (1 SID in the outer DA, and 1 SID in the SRH).
- **Encapsulation Capabilities and Parameters**

  The following describes the Cisco 8000 series router capabilities for setting or propagating certain fields in the outer IPv6 header for SRv6 encapsulated packets:

- **Source address**: Cisco 8000 series routers support a single source address (SA) for SRv6 encapsulated packets. The SA is derived from the SRv6 global configuration; if not configured, it is derived from the IPv6 Loopback address.
- **Hop limit**: Cisco 8000 series routers propagate the hop-limit of the inner packet into the outer IPv6 header during encapsulation and decapsulation operations. Propagation of the hop-limit value of locally generated OAM packets into the outer IPv6 header can be enabled via configuration.
  - Overlay encapsulation

    Default: propagate=No

    The **hop-limit propagate** command enables propagation from inner header to outer header.
  - Underlay encapsulation (TI-LFA) behavior is always in propagate mode, regardless of the CLI.

  Manual configuration of the hop-limit value in the outer IPv6 header is not supported.
- **Traffic-class**: Cisco 8000 series routers propagate the traffic-class of the inner packet into the outer IPv6 header during encapsulation and decapsulation operations. Propagation of the traffic-class value of locally generated OAM packets into the outer IPv6 header can be enabled via configuration.
  - Overlay encapsulation

    Default: propagate=No

    The **traffic-class propagate** command enables propagation from inner header to outer header.
  - Underlay encapsulation (TI-LFA) behavior is always in propagate mode, regardless of the CLI.

  Manual configuration of the traffic-class value in the outer IPv6 header is not supported.
- **Flow Label**:
  - Cisco 8000 series routers use the flow-label from the incoming IPv6 header. In case of USD operations, flow-label is used from the inner IPv6 header.
  - During H.Encap.Red operations, if the inner packet has a flow label (non-zero value), the Cisco 8000 series routers propagate it to the outer IPv6 header. If the flow label is not present (zero), it is computed.

- **PE role:**
  - Overlay H-Encaps: 3 sids (1 carrier with 3 sids per carrier)

Configuring SRv6

To enable SRv6 globally, you should first configure a locator with its prefix. The IS-IS protocol announces the locator prefix in IPv6 network and SRv6 applications (like ISIS, BGP) use it to allocate SIDs.

The following usage guidelines and restrictions apply while configuring SRv6.

- All routers in the SRv6 domain should have the same SID block (network designator) in their locator.
- The locator length should be 64-bits long.
  - The SID block portion (MSBs) cannot exceed 40 bits. If this value is less than 40 bits, user should use a pattern of zeros as a filler.

- The Node Id portion (LSBs) cannot exceed 24 bits.

- You can configure up to 8 locators to support SRv6 Flexible Algorithm. All locators prefix must share the same SID block (first 40-bits).

Enabling SRv6 with Locator

This example shows how to globally enable SRv6 and configure locator.

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc1
Router(config-srv6-locator)# prefix 2001:db8:0:a2::/64
```

Optional: Enabling Syslog Logging for Locator Status Changes

This example shows how to enable the logging of locator status.

```
Router(config)# segment-routing srv6
Router(config-srv6)# logging locator status
```

Verifying SRv6 Manager

This example shows how to verify the overall SRv6 state from SRv6 Manager point of view. The output displays parameters in use, summary information, and platform specific capabilities.

```
Router# SF-D#sh segment-routing srv6 manager
Parameters:
  SRv6 Enabled: No
  SRv6 Operational Mode: None
  Encapsulation:
    Source Address:
      Configured: ::
      Default: 77::77
    Hop-Limit: Default
    Traffic-class: Default
    SID Formats:
      f3216 <32B/16NFA> (2)
    uSID LIB Range:
    LIB Start : 0xe000
    ELIB Start : 0xfe00
    uSID WLIB Range:
    EWLIB Start : 0xfff7
Summary:
  Number of Locators: 0 (0 operational)
  Number of SIDs: 0 (0 stale)
  Max SID resources: 24000
  Number of free SID resources: 24000
  OOR:
    Thresholds (resources): Green 1200, Warning 720
    Status: Resource Available
      History: (0 cleared, 0 warnings, 0 full)
Platform Capabilities:
  SRv6: Yes
  TILFA: Yes
  Microloop-Avoidance: Yes
  Endpoint behaviors:
    End.DT6
    End.DT4
    End.DT46
    End (PSP/USD)
    End.X (PSP/USD)
```

```
    uN (PSP/USD)
    uA (PSP/USD)
    uDT6
    uDT4
    uDT46
Headend behaviors:
  T
  H.Encaps.Red
Security rules:
  SEC-1
  SEC-2
  SEC-3
Counters:
  None
Signaled parameters:
  Max-SL : 3
  Max-End-Pop-SRH : 3
  Max-H-Insert : 0 sids
  Max-H-Encap : 2 sids
  Max-End-D : 5
Configurable parameters (under srv6):
  Ranges:
   LIB : Yes
   WLIB : Yes
  Encapsulation:
    Source Address: Yes
    Hop-Limit : value=No, propagate=Yes
    Traffic-class : value=No, propagate=Yes
    Default parameters (under srv6):
  Encapsulation:
    Hop-Limit : value=128, propagate=No
    Traffic-class : value=0, propagate=No
    Max Locators: 16
    Max SIDs: 24000
    SID Holdtime: 3 mins
Router# :SF-D#
```

Verifying SRv6 Locator

This example shows how to verify the locator configuration and its operational status.

```
Router# show segment-routing srv6 locator myLoc1 detail
Name                 ID      Prefix                   Status
-------------------- ------- ------------------------ -------
myLoc1*              5       2001:db8:0:a2::/64       Up
  (*): is-default
  Interface:
    Name: srv6-myLoc1
    IFH : 0x00000170
    IPv6 address: 2001:db8:0:a2::/64
  Chkpt Obj ID: 0x2fc8
  Created: Apr 25 06:21:57.077 (00:03:37 ago)
```

Verifying SRv6 local SIDs

This example shows how to verify the allocation of SRv6 local SIDs off locator(s).

```
Router# show segment-routing srv6 locator myLoc1 sid

SID                         Function     Context                           Owner
     State  RW
```

```
-------------------------  -----------  ------------------------------   ------------------
  -----  --
2001:db8:0:a2:1::          End (PSP)    'default':1                        sidmgr
     InUse  Y
2001:db8:0:a2:40::         End.DT4      'VRF1'                             bgp-100
     InUse  Y
2001:db8:0:a2:41::         End.X (PSP)  [Hu0/1/0/1, Link-Local]            isis-srv6
     InUse  Y
```

The following example shows how to display detail information about an allocated SRv6 local SID:

```
Router# show segment-routing srv6 locator myLoc1 sid 2001:db8:0:a2:40:: detail

SID                        Function     Context                            Owner
     State  RW
-------------------------  -----------  ------------------------------   ------------------
  -----  --
2001:db8:0:a2:40::         End.DT4      'VRF1'                             bgp-100
     InUse  Y
  SID context: { table-id=0xe0000011 ('VRF1':IPv4/Unicast) }
  Locator: myLoc1'
  Allocation type: Dynamic
  Created: Feb  1 14:04:02.901 (3d00h ago)
```

show Commands

You can use the following **show** commands to verify the SRv6 global and locator configuration:

Command	Description
show segment-routing srv6 manager	Displays the summary information from SRv6 manager, including platform capabilities.
show segment-routing srv6 locator *locator-name* [**detail**]	Displays the SRv6 locator information on the router.
show segment-routing srv6 locator *locator-name* **sid** [[*sid-ipv6-address* [**detail**]	Displays the information regarding SRv6 local SID(s) allocated from a given locator.
show segment-routing srv6 sid [*sid-ipv6-address* | **all** | **stale**] [**detail**]	Displays SID information across locators. By default, only “active” (i.e. non-stale) SIDs are displayed.
show route ipv6 local-srv6	Displays all SRv6 local-SID prefixes in IPv6 RIB.

Configuring SRv6 under IS-IS

Intermediate System-to-Intermediate System (IS-IS) protocol already supports segment routing with MPLS dataplane (SR-MPLS). This feature enables extensions in IS-IS to support Segment Routing with IPv6 data plane (SRv6). The extensions include advertising the SRv6 capabilities of nodes and node and adjacency segments as SRv6 SIDs.

SRv6 IS-IS performs the following functionalities:

1. Interacts with SID Manager to learn local locator prefixes and announces the locator prefixes in the IGP domain.
2. Learns remote locator prefixes from other ISIS neighbor routers and installs the learned remote locator IPv6 prefix in RIB or FIB.
3. Allocate or learn prefix SID and adjacency SIDs, create local SID entries, and advertise them in the IGP domain.

Usage Guidelines and Restrictions

The following usage guidelines and restrictions apply for SRv6 IS-IS:

- An IS-IS address-family can support either SR-MPLS or SRv6, but both at the same time is not supported.

Configuring SRv6 IS-IS

To configure SRv6 IS-IS, you should enable SRv6 under the IS-IS IPv6 address-family. The following example shows how to configure SRv6 IS-IS.

```
Router(config)# router isis core
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc1
Router(config-isis-srv6-loc)# exit
```

Configuring SRv6 IS-IS TI-LFA

This feature introduces support for implementing TI-LFA using IS-IS SRv6.

Topology-Independent Loop-Free Alternate (TI-LFA) provides link 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. TI-LFA leverages the post-convergence path which is planned to carry the traffic and ensures link and node protection with in 50 milliseconds. TI-LFA with IS-IS SR-MPLS is already supported.

TI-LFA provides link, node, and Shared Risk Link Groups (SRLG) protection in any topology.

For more information, see Configure Topology-Independent Loop-Free Alternate (TI-LFA), on page 611.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- TI-LFA provides link protection by default. Additional tiebreaker configuration is required to enable node or SRLG protection.
- Usage guidelines for node and SRLG protection:
  - TI-LFA node protection functionality 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.
- 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.

- Cisco 8000 Series Routers support the insertion of up to two SIDs. If the computed backup path requires more than two SIDs, the backup path will not be installed in the RIB and the path will not be protected.

Configuring SRv6 IS-IS TI-LFA

The following example shows how to configure SRv6 IS-IS TI-LFA.

Note Complete the Configuring SRv6, on page 145 before performing these steps.

```
Router(config)# router isis core
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator locator1
Router(config-isis-srv6-loc)# exit
Router(config-isis)# interface loopback 0
Router(config-isis-if)# passive
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# exit
Router(config-isis)# interface bundle-ether 1201
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# fast-reroute per-prefix
Router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Router(config-isis-if-af)# exit
Router(config-isis)# interface bundle-ether 1301
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# fast-reroute per-prefix
Router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Router(config-isis-if-af)# fast-reroute per-prefix tiebreaker node-protecting index 100
Router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint index 200
Router(config-isis-if-af)# exit
```

Verification

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show isis ipv6 fast-reroute** *ipv6-prefix* **detail** command.

```
Router# show isis ipv6 fast-reroute cafe:0:0:66::/64 detail
Thu Nov 22 16:12:51.983 EST

L1 cafe:0:0:66::/64 [11/115] low priority
     via fe80::2, TenGigE0/0/0/6, SRv6-HUB6, Weight: 0
      Backup path: TI-LFA (link), via fe80::1, Bundle-Ether1201 SRv6-LF1, Weight: 0, Metric:
 51
```

```
        P node: SRv6-TP8.00 [8::8], SRv6 SID: cafe:0:0:88:1:: End (PSP)
        Backup-src: SRv6-HUB6.00
      P: No, TM: 51, LC: No, NP: No, D: No, SRLG: Yes
    src SRv6-HUB6.00-00, 6::6
```

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show route ipv6** *ipv6-prefix* **detail** command.

```
Router# show route ipv6 cafe:0:0:66::/64 detail
Thu Nov 22 16:14:07.385 EST

Routing entry for cafe:0:0:66::/64
  Known via "isis srv6", distance 115, metric 11, type level-1
  Installed Nov 22 09:24:05.160 for 06:50:02
  Routing Descriptor Blocks
    fe80::2, from 6::6, via TenGigE0/0/0/6, Protected
      Route metric is 11
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:0
      NHID:0x2000a(Ref:11)
      NHID eid:0xffffffffffffffff
      Backup path id:65
    fe80::1, from 6::6, via Bundle-Ether1201, Backup (TI-LFA)
      Repair Node(s): 8::8
      Route metric is 51
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:65              Path ref count:1
      NHID:0x2000d(Ref:11)
      NHID eid:0xffffffffffffffff
      SRv6 Headend: H.Encaps.Red
      SRv6 SID-list { cafe:0:0:88:1:: }
      MPLS eid:0x1380800000001
```

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show cef ipv6** *ipv6-prefix* **detail location** *location* command.

```
Router# show cef  ipv6 cafe:0:0:66::/64 detail location 0/0/cpu0
Thu Nov 22 17:01:58.536 EST
cafe:0:0:66::/64, version 1356, SRv6 Transit, internal 0x1000001 0x2 (ptr 0x8a4a45cc) [1],
 0x0 (0x8a46ae20), 0x0 (0x8c8f31b0)
 Updated Nov 22 09:24:05.166
 local adjacency fe80::2
 Prefix Len 64, traffic index 0, precedence n/a, priority 2
  gateway array (0x8a2dfaf0) reference count 4, flags 0x500000, source rib (7), 0 backups
                [5 type 3 flags 0x8401 (0x8a395d58) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8a46ae20, sh-ldi=0x8a395d58]
  gateway array update type-time 1 Nov 22 09:24:05.163
 LDI Update time Nov 22 09:24:05.163
 LW-LDI-TS Nov 22 09:24:05.166
   via fe80::2/128, TenGigE0/0/0/6, 8 dependencies, weight 0, class 0, protected [flags
0x400]
    path-idx 0 bkup-idx 1 NHID 0x2000a [0x8a2c2fd0 0x0]
    next hop fe80::2/128
   via fe80::1/128, Bundle-Ether1201, 8 dependencies, weight 0, class 0, backup (TI-LFA)
[flags 0xb00]
```

```
    path-idx 1 NHID 0x2000d [0x8c2670b0 0x0]
    next hop fe80::1/128, Repair Node(s): 8::8
    local adjacency
     SRv6 H.Encaps.Red SID-list {cafe:0:0:88:1::}


    Load distribution: 0 (refcount 5)

    Hash  OK  Interface                 Address
    0     Y   TenGigE0/0/0/6            fe80::2
```

This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show cef ipv6 fast-reroute-db** command.

```
Router# show cef ipv6 fast-reroute-db
Sun Dec  9 20:23:08.111 EST

 PROTECT-FRR: per-prefix [1, 0x0, 0x0, 0x98c83270]
 protect-interface: Te0/0/0/6 (0x208)
 protect-next-hop:  fe80::2/128
 ipv6 nhinfo [0x977397d0]
 Update Time Dec  9 17:29:42.427

     BACKUP-FRR: per-prefix [5, 0x0, 0x2, 0x98c83350]
     backup-interface: BE1201 (0x800002c)
     backup-next-hop:  fe80::1/128
     ipv6 nhinfo [0x977396a0 protect-frr: 0x98c83270]
 Update Time Dec  9 17:29:42.428

 PROTECT-FRR: per-prefix [1, 0x0, 0x0, 0x98c830b0]
 protect-interface: BE1201 (0x800002c)
 protect-next-hop:  fe80::1/128
 ipv6 nhinfo [0x977396a0]
 Update Time Dec  9 17:29:42.429

     BACKUP-FRR: per-prefix [5, 0x0, 0x1, 0x98c83190]
     backup-interface: Te0/0/0/6 (0x208)
     backup-next-hop:  fe80::2/128
     ipv6 nhinfo [0x977397d0 protect-frr: 0x98c830b0]
 Update Time Dec  9 17:29:42.429
```

Configuring SRv6 IS-IS Microloop Avoidance

This feature introduces support for implementing microloop avoidance using IS-IS SRv6.

Restrictions and Usage Guidelines

The following restrictions and usage guidelines apply:

- The Routing Information Base (RIB) update delay value specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The *delay-time* range is from 1 to 60000 milliseconds; the default value is 5000.

Configuring SRv6 IS-IS Microloop Avoidance

The following example shows how to configure SRv6 IS-IS Microloop Avoidance and set the Routing Information Base (RIB) update delay value.

Note Complete the Configuring SRv6, on page 145 before performing these steps.

```
Router(config)# router isis test-igp
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# microloop avoidance segment-routing
Router(config-isis-af)# commit
```

Configuring SRv6 IS-IS Flexible Algorithm

This feature introduces support for implementing Flexible Algorithm using IS-IS SRv6.

SRv6 Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

Restrictions and Usage Guidelines

The following restrictions and usage guidelines apply:

- You can configure up to 8 locators to support SRv6 Flexible Algorithm:
  - All locators prefix must share the same SID block (first 40-bits).
  - The Locator Algorithm value range is 128 to 255.

Configuring SRv6 IS-IS Flexible Algorithm

The following example shows how to configure SRv6 IS-IS Flexible Algorithm.

Note Complete the Configuring SRv6, on page 145 before performing these steps.

Note For Cisco NCS5500 Series Routers, you must first use the **hw-module profile segment-routing srv6** command to enable SRv6 functionality. Then, reload the line card after enabling this command.

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator Loc1-BE // best-effort
Router(config-srv6-locator)# prefix 2001:db8:0:a2::/64
Router(config-srv6-locator)# exit
```

```
Router(config-srv6-locators)# locator Loc1-LL // low latency
Router(config-srv6-locator)# prefix 2001:db8:1:a2::/64
Router(config-srv6-locator)# algorithm 128
Router(config-srv6-locator)# exit
Router(config-srv6)# exit
```

Configuring SRv6 IS-IS

The following example shows how to configure SRv6 IS-IS.

```
Router(config)# router isis test-igp
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator Loc1-BE
Router(config-isis-srv6-loc)# exit
Router(config-isis-srv6)# locator Loc1-LL
Router(config-isis-srv6-loc)# exit
```

Enable Flexible Algorithm for Low Latency

The following example shows how to enable Flexible Algorithm for low-latency:

- IS-IS: Configure Flexible Algorithm definition with **delay** objective
- Performance-measurement: Configure static delay per interface

```
Router(config)# router isis test-igp
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# metric-type delay
Router(config-isis-flex-algo)# exit
Router(config-isis)# interface GigabitEthernet0/0/0/0
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# root

Router(config)# performance-measurement
Router(config-perf-meas)# interface GigabitEthernet0/0/0/0
Router(config-pm-intf)# delay-measurement
Router(config-pm-intf-dm)# advertise-delay 100
Router(config-pm-intf-dm)# commit
```

Verification

```
Router# show segment-routing srv6 locator
Mon Aug 12 20:54:15.414 EDT
Name                  ID       Algo  Prefix                    Status
--------------------  -------  ----  ------------------------  -------
Loc1-BE               17       0     2001:db8:0:a2::/64        Up
Loc1-LL               18       128   2001:db8:1:a2::/64        Up


Router# show isis flex-algo 128
Mon Aug 12 21:00:54.282 EDT

IS-IS 200 Flex-Algo Database

Flex-Algo 128:

Level-2:
```

```
        Definition Priority: 128
        Definition Source: SRv6-LF1.00, (Local)
        Definition Equal to Local: Yes
        Disabled: No


Level-1:
        Definition Priority: 128
        Definition Source: SRv6-LF1.00, (Local)
        Definition Equal to Local: Yes
        Disabled: No


    Local Priority: 128
    FRR Disabled: No
    Microloop Avoidance Disabled: No
```

SRv6 Services: IPv4 L3VPN

The SRv6-based IPv4 L3VPN feature enables deployment of IPv4 L3VPN over a SRv6 data plane. Traditionally, it was done over an MPLS-based system. SRv6-based L3VPN uses SRv6 Segment IDs (SIDs) for service segments instead of labels. SRv6-based L3VPN functionality interconnects multiple sites to resemble a private network service over public infrastructure. To use this feature, you must configure SRv6-base.

For this feature, BGP allocates an SRv6 SID from the locator space, configured under SRv6-base and VPNv4 address family. For more information on this, refer to Segment Routing over IPv6 Overview, on page 140. The BGP SID can be allocated in the following ways:

- Per-VRF mode that provides End.DT4 support. End.DT4 represents the Endpoint with decapsulation and IPv4 table lookup.

BGP encodes the SRv6 SID in the prefix-SID attribute of the IPv4 L3VPN Network Layer Reachability Information (NLRI) and advertises it to IPv6 peering over an SRv6 network. The Ingress PE (provider edge) router encapsulates the VRF IPv4 traffic with the SRv6 VPN SID and sends it over the SRv6 network.

Restrictions and Usage Guidelines

- MPLS based L3VPN inter-operability is not supported on a router that is configured for SRv6-based L3VPN.
- Only IPv4 L3VPN is supported, and IPv6 L3VPN is not supported.
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.

Configuring SRv6 based IPv4 L3VPN

To enable SRv6-based L3VPN, you need to configure SRv6 under BGP and SID allocation mode. The following example shows how to configure SRv6-based L3VPN:

```
/*Configure SRv6 locator name under BGP Global*/
RP/0/0/CPU0:Router(config)# router bgp 100
RP/0/0/CPU0:Router(config-bgp)# bgp router-id 10.6.6.6
RP/0/0/CPU0:Router(config-bgp)# segment-routing srv6
```

```
RP/0/0/CPU0:Router(config-bgp-srv6)# locator my-locator
RP/0/0/CPU0:Router(config-bgp-srv6)# exit


/*Configure SRv6 locator under VRF All under VPNv4 AFI*/
RP/0/0/CPU0:Router(config)# router bgp 100
RP/0/0/CPU0:Router(config-bgp)# bgp router-id 10.6.6.6
RP/0/0/CPU0:Router(config-bgp)# address-family vpnv4 unicast
RP/0/0/CPU0:Router(config-bgp-af)# verf all
RP/0/0/CPU0:Router(config-bgp-af-vrfall)# segment-routing srv6
RP/0/0/CPU0:Router(config-bgp-af-vrfall-srv6)# locator my-locator
RP/0/0/CPU0:Router(config-bgp-af-vrfall-srv6)# exit


/*Configure a VRF with per-vrf label allocation mode*/
RP/0/0/CPU0:Router(config-bgp-af)# vrf vrf1
RP/0/0/CPU0:Router(config-bgp-vrf)# rd 106:1
RP/0/0/CPU0:Router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:Router(config-bgp-vrf-af)# segment-routing srv6
RP/0/0/CPU0:Router(config-bgp-vrf-af-srv6)# alloc mode per-vrf
RP/0/0/CPU0:Router(config-bgp-vrf-af-srv6)# exit
RP/0/0/CPU0:Router(config-bgp-vrf-af)# exit
RP/0/0/CPU0:Router(config-bgp-vrf)# neighbor 10.1.2.2
RP/0/0/CPU0:Router(config-bgp-vrf-nbr)# remote-as 100
RP/0/0/CPU0:Router(config-bgp-vrf-nbr)# address-family ipv4 unicast

/*Configure a VRF with per-ce label allocation mode*/
RP/0/0/CPU0:Router(config-bgp-af)# vrf vrf2
RP/0/0/CPU0:Router(config-bgp-vrf)# rd 106:2
RP/0/0/CPU0:Router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:Router(config-bgp-vrf-af)# segment-routing srv6
RP/0/0/CPU0:Router(config-bgp-vrf-af-srv6)# alloc mode per-ce
RP/0/0/CPU0:Router(config-bgp-vrf-af-srv6)# exit
RP/0/0/CPU0:Router(config-bgp-vrf-af)# exit
RP/0/0/CPU0:Router(config-bgp-vrf)# neighbor 10.1.2.2
RP/0/0/CPU0:Router(config-bgp-vrf-nbr)# remote-as 100
RP/0/0/CPU0:Router(config-bgp-vrf-nbr)# address-family ipv4 unicast
```

Verification

The following example shows how to verify the SRv6 based L3VPN configuration using the **show segment-routing srv6 sid** command.

In this example, End.X represents Endpoint function with Layer-3 cross-connect, End.DT4 represents Endpoint with decapsulation and IPv4 table lookup represents Endpoint with decapsulation and IPv4 cross-connect.

```
 RP/0/0/CPU0:Router# show segment-routing srv6 sid
*** Locator: 'my_locator' ***
SID                         Function     Context                          Owner
     State  RW
--------------------------  -----------  -------------------------------   ------------------
  -----  --
cafe:0:0:66:1::             End (PSP)    'my_locator':1                    sidmgr
     InUse  Y
cafe:0:0:66:40::            End.X (PSP)  [Te0/0/0/2, Link-Local]           isis-srv6
     InUse  Y
cafe:0:0:66:41::            End.X (PSP)  [BE6801, Link-Local]              isis-srv6
     InUse  Y
cafe:0:0:66:42::            End.X (PSP)  [BE5601, Link-Local]              isis-srv6
     InUse  Y
cafe:0:0:66:43::            End.X (PSP)  [BE5602, Link-Local]              isis-srv6
     InUse  Y
cafe:0:0:66:44::            End.DT4      'VRF1'                            bgp-100
```

```
     InUse  Y
cafe:0:0:66:45::            End.DT4      'VRF2'                              bgp-100
     InUse  Y
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show segment-routing srv6** *SID-prefix* **detail** command.

```
RP/0/RP0/CPU0:Router#sh segment-routing srv6 sid cafe:0:0:66:43:: detail
Sun Dec  9 16:52:54.015 EST
*** Locator: 'my_locator' ***
SID                         Function     Context                           Owner
     State  RW
--------------------------  -----------  ------------------------------    ------------------
  -----  --
cafe:0:0:66:44::            End.DT4      'VRF1'                              bgp-100
     InUse  Y
  SID context: { table-id=0xe0000001 ('VRF1':IPv4/Unicast) }
  Locator: 'my_locator'
  Allocation type: Dynamic
  Created: Dec  8 16:34:32.506 (1d00h ago)
RP/0/RP0/CPU0:SRv6-HUB6#sh segment-routing srv6 sid cafe:0:0:66:47:: detail
Sun Dec  9 16:54:26.073 EST
*** Locator: 'my_locator' ***
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show bgp vpnv4 un rd** *route-distinguisher prefix* command on Egress PE.

```
RP/0/RP0/CPU0:SRv6-Hub6#sh bgp vpnv4 un rd 106:1 10.15.0.0/30
Wed Nov 21 16:08:44.765 EST
BGP routing table entry for 10.15.0.0/30, Route Distinguisher: 106:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker            2282449     2282449
    SRv6-VPN SID: cafe:0:0:66:44::/128
Last Modified: Nov 21 15:50:34.235 for 00:18:10
Paths: (2 available, best #1)
  Advertised to peers (in unique update groups):
    2::2
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    2::2
  200
    10.1.2.2 from 10.1.2.2  (10.7.0.1)
      Origin IGP, localpref 200, valid, internal, best, group-best, import-candidate
      Received Path ID 0, Local Path ID 1, version 2276228
      Extended community: RT:201:1
  Path #2: Received by speaker 0
  Not advertised to any peer
  200
    10.2.2.2 from 10.2.2.2 (10.20.1.2)
      Origin IGP, localpref 100, valid, internal
      Received Path ID 0, Local Path ID 0, version 0
      Extended community: RT:201:1
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show bgp vpnv4 un rd** *route-distinguisher prefix* command on Ingress PE.

```
RP/0/RP0/CPU0:SRv6-LF1#sh bgp vpnv4 un rd 106:1 10.15.0.0/30
Wed Nov 21 16:11:45.538 EST
BGP routing table entry for 10.15.0.0/30, Route Distinguisher: 106:1
Versions:
  Process           bRIB/RIB  SendTblVer
```

```
  Speaker            2286222     2286222
Last Modified: Nov 21 15:47:26.288 for 00:24:19
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  200, (received & used)
    6::6 (metric 24) from 2::2 (6.6.6.6)
      Received Label 3
      Origin IGP, localpref 200, valid, internal, best, group-best, import-candidate,
not-in-vrf
      Received Path ID 1, Local Path ID 1, version 2286222
      Extended community: RT:201:1
      Originator: 6.6.6.6, Cluster list: 2.2.2.2
      SRv6-VPN-SID: T1-cafe:0:0:66:44:: [total 1]
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show route vrf** *vrf-name prefix* **detail** command.

```
RP/0/RP0/CPU0:Router#sh route vrf VRF1 10.15.0.0/30detail
Wed Nov 21 16:35:17.775 EST
Routing entry for 10.15.0.0/30
  Known via "bgp 100", distance 200, metric 0
  Tag 200, type internal
  Installed Nov 21 16:35:14.107 for 00:00:03
  Routing Descriptor Blocks
    6::6, from 2::2
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Source RD attributes: 0x0000:106:1
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [base]
      SRv6 SID-list { cafe:0:0:66:44:: }
      MPLS eid:0x1380600000001
  Route version is 0xd (13)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_REMOTE
  Download Priority 3, Download Version 3038384
  No advertising protos.
```

The following example shows how to verify the SRv6 based L3VPN configuration for per-ce allocation mode using the **show bgp vrf** *vrf* **nexthop-set** command.

```
RP/0/RP0/CPU0:Router#show bgp vrf VRF2 nexthop-set
Wed Nov 21 15:52:17.464 EST
 Resilient per-CE nexthop set, ID 3
 Number of nexthops 1, Label 0, Flags 0x2200
 SRv6-VPN SID: cafe:0:0:66:46::/128
 Nexthops:
10.1.2.2
 Reference count 1,
 Resilient per-CE nexthop set, ID 4
 Number of nexthops 2, Label 0, Flags 0x2100
 SRv6-VPN SID: cafe:0:0:66:47::/128
 Nexthops:
```

```
 10.1.2.2
 10.2.2.2
Reference count 2,
```

The following example shows how to verify the SRv6 based L3VPN configuration using the **show cef vrf** *vrf-name prefix* **detail location** *line-card* command.

```
RP/0/RP0/CPU0:Router#sh cef vrf VRF1 10.15.0.0/30 detail location 0/0/cpu0
Wed Nov 21 16:37:06.894 EST
151.1.0.0/30, version 3038384, SRv6 Transit, internal 0x5000001 0x0 (ptr 0x9ae6474c) [1],
0x0 (0x0), 0x0 (0x8c11b238)
 Updated Nov 21 16:35:14.109
 Prefix Len 30, traffic index 0, precedence n/a, priority 3
  gateway array (0x8cd85190) reference count 1014, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x40441 (0x8a529798) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Nov 21 14:47:26.816
 LDI Update time Nov 21 14:52:53.073
  Level 1 - Load distribution: 0
  [0] via cafe:0:0:66::/128, recursive
   via cafe:0:0:66::/128, 7 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x8acb53cc 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:0:66::/128 via cafe:0:0:66::/64
     SRv6 H.Encaps.Red SID-list {cafe:0:0:66:44::}
    Load distribution: 0 (refcount 1)
    Hash  OK  Interface                 Address
    0     Y   Bundle-Ether1201          fe80::2
```

SRv6 Services: IPv6 L3VPN

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: IPv6 L3VPN	Release 7.8.1	With this feature, the egress PE can signal an SRv6 Service SID with the BGP overlay service route. The ingress PE encapsulates the IPv4/IPv6 payload in an outer IPv6 header where the destination address is the SRv6 Service SID provided by the egress PE. BGP messages between PEs carry SRv6 Service SIDs as a means to interconnect PEs and form VPNs.

Building on the messages and procedures defined in IETF draft "BGP/MPLS IP Virtual Private Networks (VPNs)", this feature provides IPv6 L3VPNs (VPNv6) over an SRv6 network.

In SRv6-based L3VPNs, the egress PE signals an SRv6 Service SID with the BGP overlay service route. The ingress PE encapsulates the IPv4/IPv6 payload in an outer IPv6 header where the destination address is the SRv6 Service SID provided by the egress PE. BGP messages between PEs carry SRv6 Service SIDs as a means to interconnect PEs and form VPNs.

SRv6 Service SID refers to a segment identifier associated with one of the SRv6 service-specific behaviors on the advertising VPNv6 PE router, such as END.DT6 (Endpoint with decapsulation and IPv6 table lookup) behaviors.

Based on the messages and procedures defined in IETF draft "SRv6 BGP based Overlay services", BGP encodes the SRv6 Service SID in the prefix-SID attribute of the IPv6 L3VPN Network Layer Reachability Information (NLRI) and advertises it to its IPv6 BGP peers.

BGP allocates an SRv6 Service SID from the locator space, configured under SRv6 and VPNv6 address family. For more information on this, see Segment Routing over IPv6 Overview. The SRv6 Service SID can be allocated in the following ways:

- Per-VRF mode that provides End.DT6 support. End.DT6 represents the Endpoint with decapsulation and IPv6 table lookup.

Usage Guidelines and Restrictions

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF, or per-prefix.
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.
- Dual-Stack L3VPN Services (IPv4, IPv6) are supported.
- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.

Configuring SRv6-based IPv6 L3VPN

To enable SRv6-based L3VPN, you need to configure SRv6 under BGP and configure the SID allocation mode.

The following examples show how to configure SRv6-based L3VPN.

Configure SRv6 Locator Under BGP Global

This example shows how to configure the SRv6 locator name under BGP Global:

```
RP/0/0/CPU0:Node1(config)# router bgp 100
RP/0/0/CPU0:Node1(config-bgp)# segment-routing srv6
RP/0/0/CPU0:Node1(config-bgp-gbl-srv6)# locator Node1-locator
RP/0/0/CPU0:Node1(config-bgp-gbl-srv6)# exit
RP/0/0/CPU0:Node1(config-bgp)# address-family vpnv6 unicast
RP/0/0/CPU0:Node1(config-bgp-af)# exit
RP/0/0/CPU0:Node1(config-bgp)# neighbor 3001::1:1:1:4
RP/0/0/CPU0:Node1(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:Node1(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/0/CPU0:Node1(config-bgp-nbr-af)# exit
RP/0/0/CPU0:Node1(config-bgp-nbr)# exit
RP/0/0/CPU0:Node1(config-bgp)# vrf vrf_cust6
RP/0/0/CPU0:Node1(config-bgp-vrf)# rd 100:6
RP/0/0/CPU0:Node1(config-bgp-vrf)# address-family ipv6 unicast
RP/0/0/CPU0:Node1(config-bgp-vrf-af)# commit
```

Running Configuration

```
router bgp 100
 segment-routing srv6
  locator Node1-locator
```

```
 !
 address-family vpnv6 unicast
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
  !
 !
!
end
```

Configure SRv6 Locator For All VRF Under VPNv6 AFI

This example shows how to configure the SRv6 locator for all VRFs under VPNv6 address family, with per-VRF label allocation mode:

```
RP/0/0/CPU0:Node1(config)# router bgp 100
RP/0/0/CPU0:Node1(config-bgp)# address-family vpnv6 unicast
RP/0/0/CPU0:Node1(config-bgp-af)# vrf all
RP/0/0/CPU0:Node1(config-bgp-af-vrfall)# segment-routing srv6
RP/0/0/CPU0:Node1(config-bgp-af-vrfall-srv6)# locator Node1-locator
RP/0/0/CPU0:Node1(config-bgp-af-vrfall-srv6)# alloc mode per-vrf
RP/0/0/CPU0:Node1(config-bgp-af-vrfall-srv6)# exit
RP/0/0/CPU0:Node1(config-bgp-af-vrfall)# exit
RP/0/0/CPU0:Node1(config-bgp-af)# exit
RP/0/0/CPU0:Node1(config-bgp)# neighbor 3001::1:1:1:4
RP/0/0/CPU0:Node1(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:Node1(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/0/CPU0:Node1(config-bgp-nbr-af)# exit
RP/0/0/CPU0:Node1(config-bgp-nbr)# exit
RP/0/0/CPU0:Node1(config-bgp)# vrf vrf_cust6
RP/0/0/CPU0:Node1(config-bgp-vrf)# rd 100:6
RP/0/0/CPU0:Node1(config-bgp-vrf)# address-family ipv6 unicast
RP/0/0/CPU0:Node1(config-bgp-vrf-af)# commit
```

Running Configuration

```
router bgp 100
 address-family vpnv6 unicast
  vrf all
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf
   !
  !
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
  !
 !
```

```
!
end
```

Configure an Individual VRF with Per-VRF Label Allocation Mode

This example shows how to configure the SRv6 locator for an individual VRF, with per-VRF label allocation mode:

```
RP/0/0/CPU0:Node1(config)# router bgp 100
RP/0/0/CPU0:Node1(config-bgp)# address-family vpnv6 unicast
RP/0/0/CPU0:Node1(config-bgp-af)# exit
RP/0/0/CPU0:Node1(config-bgp)# neighbor 3001::1:1:1:4
RP/0/0/CPU0:Node1(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:Node1(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/0/CPU0:Node1(config-bgp-nbr-af)# exit
RP/0/0/CPU0:Node1(config-bgp-nbr)# exit
RP/0/0/CPU0:Node1(config-bgp)# vrf vrf_cust6
RP/0/0/CPU0:Node1(config-bgp-vrf)# rd 100:6
RP/0/0/CPU0:Node1(config-bgp-vrf)# address-family ipv6 unicast
RP/0/0/CPU0:Node1(config-bgp-vrf-af)# segment-routing srv6
RP/0/0/CPU0:Node1(config-bgp-vrf-af-srv6)# locator Node1-locator
RP/0/0/CPU0:Node1(config-bgp-vrf-af-srv6)# alloc mode per-vrf
RP/0/0/CPU0:Node1(config-bgp-vrf-af-srv6)# commit
```

Running Configuration

```
router bgp 100
 address-family vpnv6 unicast
 !
 neighbor 3001::1:1:1:4
  remote-as 100
  address-family vpnv6 unicast
  !
 !
 vrf vrf_cust6
  rd 100:6
  address-family ipv6 unicast
   segment-routing srv6
    locator Node1-locator
    alloc mode per-vrf
   !
  !
 !
!
end
```

Verification

The following examples shows how to verify the SRv6 based L3VPN configurations for an Individual VRF with per VRF label allocation mode.

In this example, End.X represents Endpoint function with Layer-3 cross-connect, and End.DT6 represents Endpoint with decapsulation and IPv6 table lookup.

```
RP/0/RSP0/CPU0:Node1# show segment-routing srv6 sid
Fri Jan 15 18:58:04.911 UTC

*** Locator: 'Node1-locator' ***

SID                         Behavior          Context                           Owner
          State  RW
--------------------------  ----------------  ------------------------------
```

```
-----------------  -----  --
cafe:0:0:1:1::              End (PSP)         'default':1                       sidmgr
          InUse  Y
cafe:0:0:1:40::             End.X (PSP)       [Hu0/0/0/0, Link-Local]           isis-1
          InUse  Y
cafe:0:0:1:41::             End.X (PSP)       [Hu0/0/0/1, Link-Local]           isis-1
          InUse  Y
cafe:0:0:1:47::             End.X (PSP)       [Hu0/0/0/0, Link-Local]:P         isis-1
          InUse  Y
cafe:0:0:1:48::             End.X (PSP)       [Hu0/0/0/1, Link-Local]:P         isis-1
          InUse  Y
cafe:0:0:1:49::             End.DT6           'default'                         bgp-100
          InUse  Y
cafe:0:0:1:4a::             End.DT6           'vrf_cust6'                       bgp-100
          InUse  Y
```

The following examples show how to verify the SRv6 based L3VPN configuration using the **show bgp vpnv6 unicast** commands on the Ingress PE.

```
RP/0/RSP0/CPU0:Node1# show bgp vpnv6 unicast summary
Fri Jan 15 18:37:04.791 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker              21         21         21         21          21           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
3001::1:1:1:4     0   100    1352    1352       21    0    0 01:46:26          1
3001::1:1:1:5     0   100    1351    1351       21    0    0 01:44:47          1

RP/0/RSP0/CPU0:Node1# show bgp vpnv6 unicast rd 100:6
Fri Jan 15 18:38:02.919 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::12:1:1:1/128 ::                        0         32768 ?
*>i3001::12:1:1:4/128 3001::1:1:1:4             0    100      0 ?
*>i3001::12:1:1:5/128 3001::1:1:1:5             0    100      0 ?

Processed 3 prefixes, 3 paths
```

```
RP/0/RSP0/CPU0:Node1# show bgp vpnv6 unicast rd 100:6 3001::12:1:1:4/128
Fri Jan 15 18:38:26.492 UTC
BGP routing table entry for 3001::12:1:1:4/128, Route Distinguisher: 100:6
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 17          17
Last Modified: Jan 15 16:50:44.032 for 01:47:43
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (received & used)
    3001::1:1:1:4 (metric 30) from 3001::1:1:1:4 (10.1.1.4)
      Received Label 0x4900
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 17
      Extended community: RT:100:6
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:0:4::, Behavior:18, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv6 Unicast, Source VRF: vrf_cust6, Source Route Distinguisher: 100:6
```

The following examples show how to verify the BGP prefix information for VRF instances:

```
RP/0/RSP0/CPU0:Node1# show bgp vrf vrf_cust6 ipv6 unicast
Fri Jan 15 18:38:49.705 UTC
BGP VRF vrf_cust6, state: Active
BGP Route Distinguisher: 100:6
VRF ID: 0x60000008
BGP router identifier 10.1.1.1, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800017   RD version: 21
BGP main routing table version 21
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::12:1:1:1/128 ::                        0         32768 ?
*>i3001::12:1:1:4/128 3001::1:1:1:4             0    100      0 ?
*>i3001::12:1:1:5/128 3001::1:1:1:5             0    100      0 ?

Processed 3 prefixes, 3 paths

RP/0/RSP0/CPU0:Node1# show bgp vrf vrf_cust6 ipv6 unicast 3001::12:1:1:4/128
Fri Jan 15 18:39:05.115 UTC
BGP routing table entry for 3001::12:1:1:4/128, Route Distinguisher: 100:6
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 17          17
Last Modified: Jan 15 16:50:44.032 for 01:48:21
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local, (received & used)
```

```
    3001::1:1:1:4 (metric 30) from 3001::1:1:1:4 (10.1.1.4)
      Received Label 0x4900
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 17
      Extended community: RT:100:6
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:0:4::, Behavior:18, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv6 Unicast, Source VRF: vrf_cust6, Source Route Distinguisher: 100:6
```

The following examples show how to verify the current routes in the Routing Information Base (RIB):

```
RP/0/RSP0/CPU0:Node1# show route vrf vrf_cust6 ipv6 unicast
Fri Jan 15 18:39:20.619 UTC

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is not set

L    3001::12:1:1:1/128 is directly connected,
      21:14:10, Loopback105
B    3001::12:1:1:4/128
      [200/0] via 3001::1:1:1:4 (nexthop in vrf default), 01:48:36
B    3001::12:1:1:5/128
      [200/0] via 3001::1:1:1:5 (nexthop in vrf default), 01:46:56

RP/0/RSP0/CPU0:Node1# show route vrf vrf_cust6 ipv6 unicast 3001::12:1:1:4/128
Fri Jan 15 18:39:39.689 UTC

Routing entry for 3001::12:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 15 16:50:44.381 for 01:48:55
  Routing Descriptor Blocks
    3001::1:1:1:4, from 3001::1:1:1:4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
  No advertising protos.

RP/0/RSP0/CPU0:Node1# show route vrf vrf_cust6 ipv6 unicast 3001::12:1:1:4/128 detail
Fri Jan 15 18:39:51.573 UTC

Routing entry for 3001::12:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 15 16:50:44.381 for 01:49:07
  Routing Descriptor Blocks
    3001::1:1:1:4, from 3001::1:1:1:4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Source RD attributes: 0x0000:100:6
```

```
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [base], SID-list {cafe:0:0:4:49::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_REMOTE
  Download Priority 3, Download Version 3
  No advertising protos.
```

The following examples show how to verify the current IPv6 Cisco Express Forwarding (CEF) table:

```
RP/0/RSP0/CPU0:Node1# show cef vrf vrf_cust6 ipv6
Fri Jan 15 18:40:15.833 UTC

::/0
  drop       default handler
3001::12:1:1:1/128
  receive    Loopback105
3001::12:1:1:4/128
  recursive    cafe:0:0:4::/128
3001::12:1:1:5/128
  recursive    cafe:0:0:5::/128
fe80::/10
  receive
ff02::/16
  receive
ff02::2/128
  receive
ff02::1:ff00:0/104
  receive
ff05::/16
  receive
ff12::/16
  receive

RP/0/RSP0/CPU0:Node1# show cef vrf vrf_cust6 ipv6 3001::12:1:1:4/128
Fri Jan 15 18:40:28.853 UTC
3001::12:1:1:4/128, version 3, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x78f2e0e0) [1],
 0x0 (0x0), 0x0 (0x8886b768)
 Updated Jan 15 16:50:44.385
 Prefix Len 128, traffic index 0, precedence n/a, priority 3
   via cafe:0:0:4::/128, 9 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78a0f504 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:0:4::/128 via cafe:0:0:4::/64
    SRv6 H.Encaps.Red SID-list {cafe:0:0:4:49::}

RP/0/RSP0/CPU0:Node1# show cef vrf vrf_cust6 ipv6 3001::12:1:1:4/128 detail
Fri Jan 15 18:40:55.327 UTC
3001::12:1:1:4/128, version 3, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x78f2e0e0) [1],
 0x0 (0x0), 0x0 (0x8886b768)
 Updated Jan 15 16:50:44.385
 Prefix Len 128, traffic index 0, precedence n/a, priority 3
  gateway array (0x7883b320) reference count 1, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x788e6ad8) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Jan 15 16:50:44.385
 LDI Update time Jan 15 16:50:44.385

  Level 1 - Load distribution: 0
  [0] via cafe:0:0:4::/128, recursive
```

```
via cafe:0:0:4::/128, 9 dependencies, recursive [flags 0x6000]
 path-idx 0 NHID 0x0 [0x78a0f504 0x0]
 next hop VRF - 'default', table - 0xe0800000
 next hop cafe:0:0:4::/128 via cafe:0:0:4::/64
 SRv6 H.Encaps.Red SID-list {cafe:0:0:4:49::}

 Load distribution: 0 1 (refcount 1)

 Hash  OK  Interface                 Address
 0     Y   HundredGigE0/0/0/0        remote
 1     Y   HundredGigE0/0/0/1        remote
```

SRv6 Services: L3VPN VPNv4 Active-Standby Redundancy using Port-Active Mode

The Segment Routing IPv6 (SRv6) Services: L3VPN VPNv4 Active-Standby Redundancy using Port-Active Mode feature provides all-active per-port load balancing for multihoming. The forwarding of traffic is determined based on a specific interface rather than per-flow across multiple Provider Edge routers. This feature enables efficient load-balancing and provides faster convergence. In an active-standby scenario, the active PE router is detected using designated forwarder (DF) election by modulo calculation and the interface of the standby PE router brought down. For Modulo calculation, byte 10 of the Ethernet Segment Identifier (ESI) is used.

Restrictions

- This feature can only be configured for bundle interfaces.
- When an EVPN Ethernet Segment (ES) is configured with port-active load-balancing mode, you cannot configure ACs of that bundle on bridge-domains with a configured EVPN instance (EVI). EVPN Layer 2 bridging service is not compatible with port-active load-balancing.

SRv6 Services for L3VPN Active-Standby Redundancy using Port-Active Mode: Operation

Under port-active operational mode, EVPN Ethernet Segment (ES) routes are exchanged across BGP for the routers servicing the multihomed ES. Each PE router then builds an ordered list of the IP addresses of all PEs connected to the ES, including itself, and assigns itself an ordinal for its position in the list. The ordinals are used with the modulo calculation to determine which PE will be the Designated Forwarder (DF) for a given ES. All non-DF PEs will take the respective bundles out of service.

In the case of link or port failure, the active DF PE withdraws its ES route. This re-triggers DF election for all PEs that service the ES and a new PE is elected as DF.

Configure SRv6 Services L3VPN Active-Standby Redundancy using Port-Active Mode

This section describes how you can configure SRv6 services L3VPN active-standby redundancy using port-active mode under an Ethernet Segment (ES).

Configuration Example

```
/* Configure Ethernet Link Bundles */
Router# configure
Router(config)# interface Bundle-Ether10
Router(config-if)# ipv4 address 10.0.0.2 255.255.255.0
Router(config-if)# ipv6 address 2001:DB8::1
Router(config-if)# lacp period short
Router(config-if)# mac-address 1.2.3
Router(config-if)# bundle wait-while 0
Router(config-if)# exit
Router(config)# interface GigabitEthernet 0/2/0/5
Router(config-if)# bundle id 14 mode active
Router(config-if)# commit

/* Configure load balancing. */
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 11.11.11.11.11.11.11.11.14
Router(config-evpn-ac-es)# load-balancing-mode port-active
Router(config-evpn-ac-es)# commit
!
/* Configure address family session in BGP. */
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 192.168.0.2
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp)# neighbor 192.168.0.3
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr)# commit
```

Running Configuration

```
interface Bundle-Ether14
 ipv4 address 14.0.0.2 255.255.255.0
 ipv6 address 14::2/64
 lacp period short
 mac-address 1.2.3
 bundle wait-while 0
!
interface GigabitEthernet0/2/0/5
 bundle id 14 mode active
!
evpn
 interface Bundle-Ether14
  ethernet-segment
   identifier type 0 11.11.11.11.11.11.11.11.14
   load-balancing-mode port-active
```

```
  !
 !
!
router bgp 100
 bgp router-id 192.168.0.2
 address-family l2vpn evpn
 !
 neighbor 192.168.0.3
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
!
```

Verification

Verify the SRv6 services L3VPN active-standby redundancy using port-active mode configuration.

```
/* Verify ethernet-segment details on active DF router */
Router# show evpn ethernet-segment interface Bundle-Ether14 detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0011.1111.1111.1111.1114 BE14                              192.168.0.2
                                                           192.168.0.3
    ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether14
     Interface MAC  : 0001.0002.0003
     IfHandle       : 0x000041d0
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.1111.1114
  ES Import RT      : 1111.1111.1111 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Peering Details   :
     192.168.0.2 [MOD:P:00]
     192.168.0.3 [MOD:P:00]

  Service Carving Results:
     Forwarders     : 0
     Permanent      : 0
     Elected        : 0
     Not Elected    : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : None
  Remote SHG labels : 0


/* Verify bundle Ethernet configuration on active DF router */
Router# show bundle bundle-ether 14
Bundle-Ether14
  Status:                                  Up
```

```
  Local links <active/standby/configured>:   1 / 0 / 1
  Local bandwidth <effective/available>:     1000000 (1000000) kbps
  MAC address (source):                      0001.0002.0003 (Configured)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          Off
  Load balancing:
    Link order signaling:                    Not configured
    Hash type:                               Default
    Locality threshold:                      None
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured
  IPv6 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Gi0/2/0/5             Local            Active       0x8000, 0x0003     1000000
      Link is Active
```

```
/*  Verify ethernet-segment details on standby DF router */
Router# show evpn ethernet-segment interface bundle-ether 10 detail

Router# show evpn ethernet-segment interface Bundle-Ether24 detail
Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0011.1111.1111.1111.1114 BE24                              192.168.0.2
                                                           192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether24
     Interface MAC  : 0001.0002.0003
     IfHandle       : 0x000041b0
     State          : Standby
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.1111.1114
  ES Import RT      : 1111.1111.1111 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Peering Details   :
     192.168.0.2 [MOD:P:00]
     192.168.0.3 [MOD:P:00]

  Service Carving Results:
     Forwarders     : 0
     Permanent      : 0
     Elected        : 0
     Not Elected    : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
```

```
  Local SHG label   : None
  Remote SHG labels : 0

/* Verify bundle configuration on standby DF router */
Router# show bundle bundle-ether 24

Bundle-Ether24
  Status:                                  LACP OOS (out of service)
  Local links <active/standby/configured>:   0 / 1 / 1
  Local bandwidth <effective/available>:     0 (0) kbps
  MAC address (source):                    0001.0002.0003 (Configured)
  Inter-chassis link:                      No
  Minimum active links / bandwidth:        1 / 1 kbps
  Maximum active links:                    64
  Wait while timer:                        Off
  Load balancing:
    Link order signaling:                  Not configured
    Hash type:                             Default
    Locality threshold:                    None
  LACP:                                    Operational
    Flap suppression timer:                Off
    Cisco extensions:                      Disabled
    Non-revertive:                         Disabled
  mLACP:                                   Not configured
  IPv4 BFD:                                Not configured
  IPv6 BFD:                                Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Gi0/0/0/4             Local            Standby      0x8000, 0x0002     1000000
      Link is in standby due to bundle out of service state
```

SRv6 Services: L3VPN VPNv4 Active-Active Redundancy

This feature provides active-active connectivity to a CE device in a L3VPN deployment. The CE device can be Layer-2 or Layer-3 device connecting to the redundant PEs over a single LACP LAG port.

Depending on the bundle hashing, an ARP or IPv6 Network Discovery (ND) packet can be sent to any of the redundant routers. As a result, not all entries will exist on a given PE. In order to provide complete awareness, Layer-3 local route learning is augmented with remote route-synchronization programming.

Route synchronization between service PEs is required in order to provide minimum interruption to unicast and multicast services after failure on a redundant service PE. The following EVPN route-types are used for Layer-3 route synchronization:

- EVPN route-type 2 for synchronizing ARP tables
- EVPN route-type 7/8 for synchronizing IGMP JOINS/LEAVES

In a Layer-3 CE scenario, the router that connects to the redundant PEs may establish an IGP adjacency on the bundle port. In this case, the adjacency will be formed to one of the redundant PEs, and IGP customer routes will only be present on that PE. To synchronize Layer-3 customer subnet routes (IP Prefixes), the EVPN route-type 5 is used to carry the ESI and ETAG as well as the gateway address (prefix next-hop address).

Note Gratuitous ARP (GARP) or IPv6 Network Advertisement (NA) replay is not needed for CEs connected to the redundant PEs over a single LAG port.

The below configuration enables Layer-3 route synchronization for routes learned on the Ethernet-segment sub-interfaces.

```
evpn
 route-sync vrf default
!
vrf RED
 evi route-sync 10
!
vrf BLUE
 evi route-sync 20
!
```

Note EVPN does not support untagged interfaces.

SRv6 Services: BGP Global IPv6

This feature extends support of SRv6-based BGP services to include Internet (IPv6) services by implementing End.DT6 SRv6 functions at the PE node, as defined in IETF draft "SRv6 BGP based Overlay services".

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv4 unicast address family
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.

BGP Global IPv6 Over SRv6 with Per-VRF SID Allocation Mode (End.DT6)

To configure BGP global IPv6 over SRv6, use the following commands:

- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 alloc mode per-vrf**: Specify the SID behavior (allocation mode).

  The **per-vrf** keyword specifies that the same label is be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 alloc mode** {**per-vrf** | **route-policy** *policy_name*}: Specify the SID behavior (allocation mode).
  - **per-vrf**: Specifies that the same label is be used for all the routes advertised from a unique VRF.
  - **route-policy** *policy_name*: Uses a route policy to determine the SID allocation mode and locator (if provided) for given prefix.
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 locator** *WORD*: Specify the locator
- **router bgp** *as-number* {**af-group** *WORD*| **neighbor-group** *WORD* | **neighbor** *ipv6-addr*} **address-family ipv6 unicast encapsulation-type srv6**: Specify the encapuslation type for SRv6.

- Use **af-group** *WORD* to apply the SRv6 encapsulation type to the address family group for BGP neighbors.
- Use **neighbor-group** *WORD*to apply the SRv6 encapsulation type to the neighbor group for Border Gateway Protocol (BGP) neighbors.
- Use **neighbor** *ipv6-addr* to apply the SRv6 encapsulation type to the specific BGP neighbor.

Use Case 1: BGP Global IPv6 over SRv6 with Per-AFI SID Allocation

The following example shows how to configure BGP global IPv6 over SRv6 with per-VRF SID allocation.

```
Node1(config)# router bgp 100
Node1(config-bgp)# bgp router-id 10.1.1.1
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:5
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 100
 bgp router-id 10.1.1.1
 segment-routing srv6
  locator Node1
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf
  !
 !
 neighbor 3001::1:1:1:4
  address-family ipv6 unicast
   encapsulation-type srv6
  !
 !
 neighbor 3001::1:1:1:5
  address-family ipv6 unicast
   encapsulation-type srv6
```

Use Case 2: BGP Global IPv6 over SRv6 with Per-Prefix SID Allocation

This use case provides the ability to assign a specific SRv6 locator for a given prefix or a set of prefixes. The egress PE advertises the prefix with the specified locator. This allows for per-prefix steering into desired transport behaviors, such as Flex Algo.

To assign an SRv6 locator for a specific prefix, configure a route policy to specify the SID allocation mode based on match criteria. Examples of match criteria are destination-based match or community-based match.

- Supported SID allocation modes are per-VRF and per-CE.
- For per-VRF allocation mode, you can also specify the SRv6 locator.
  - If an SRv6 locator is specified in the route policy, BGP will use that to allocate per-VRF SID. If the specified locator is invalid, the SID will not be allocated.
  - If an SRv6 locator is not specified in the route policy, the default locator is used to allocate the SID. If the default locator is not configured in BGP, then the SID will not be allocated.
- Per-CE allocation mode always uses the default locator to allocate the SID.

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

The following example shows a route policy specifying the SID allocation mode with destination-based match:

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (3001::1:1:1:1/128) then
Node1(config-rpl-if)# set srv6-alloc-mode per-vrf locator locator1
Node1(config-rpl-if)# elseif destination in (3001::2:2:2:2/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf locator locator2
Node1(config-rpl-elseif)# elseif destination in (3001::3:3:3:3/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf
Node1(config-rpl-elseif)# elseif destination in (3001::4:4:4:4/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-ce
Node1(config-rpl-elseif)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
```

The following example shows how to configure BGP global IPv6 over SRv6 with a route policy to determine the SID allocation mode for given prefix.

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family ipv4 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# alloc mode route-policy set_per_prefix_locator_rpl
```

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (3001::1:1:1:1/128) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (3001::2:2:2:2/128) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3001::3:3:3:3/128) then
    set srv6-alloc-mode per-vrf
  elseif destination in (3001::4:4:4:4/128) then
    set srv6-alloc-mode per-ce
  else
    drop
  endif
end-policy
!
router bgp 100
 address-family ipv6 unicast
```

```
  segment-routing srv6
   alloc mode route-policy set_per_prefix_locator_rpl
  !
 !
```

Verify that the local and received SIDs have been correctly allocated under BGP IPv6 address family:

```
Node1# show bgp ipv6 unicast local-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Local Sid                                Alloc mode   Locator
*> 3001::1:1:1:1/128  fc00:0:0:1:41::                          per-vrf      locator1
*> 3001::2:2:2:2/128  fc00:0:8:1:41::                          per-vrf      locator2
*> 3001::3:3:3:3/128  fc00:0:9:1:42::                          per-vrf      locator4
*> 3001::4:4:4:4/128  fc00:0:9:1:43::                          per-ce       locator4
*> 3001::5:5:5:5/128  NO SRv6 Sid                              -            -
* i3008::8:8:8:8/128  NO SRv6 Sid                              -            -


Node1# show bgp ipv6 unicast received-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop                            Received Sid
*> 3001::1:1:1:1/128  66.2.2.2                            NO SRv6 Sid
*> 3001::2:2:2:2/128  66.2.2.2                            NO SRv6 Sid
*> 3001::3:3:3:3/128  66.2.2.2                            NO SRv6 Sid
*> 3001::4:4:4:4/128  66.2.2.2                            NO SRv6 Sid
*> 3001::5:5:5:5/128  66.2.2.2                            NO SRv6 Sid
* i3008::8:8:8:8/128  77.1.1.2                            fc00:0:0:2:41::
```

Verification

The following examples show how to verify the BGP global IPv6 configuration using the **show bgp ipv6 unicast** commands.

```
RP/0/RSP0/CPU0:Node1# show bgp ipv6 unicast summary
Fri Jan 15 21:07:04.681 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               4          4          4          4           4           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
3001::1:1:1:4     0   100    1502    1502        4    0    0 04:16:26          1
3001::1:1:1:5     0   100    1501    1501        4    0    0 04:14:47          1

RP/0/RSP0/CPU0:Node1# show bgp ipv6 unicast
```

```
Fri Jan 15 21:07:26.818 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 3001::13:1:1:1/128 ::                       0         32768 i
*>i3001::13:1:1:4/128 3001::1:1:1:4            0    100      0 i
*>i3001::13:1:1:5/128 3001::1:1:1:5            0    100      0 i

Processed 3 prefixes, 3 paths

RP/0/RSP0/CPU0:Node1# show bgp ipv6 unicast 3001::13:1:1:4/128
Fri Jan 15 21:07:50.309 UTC
BGP routing table entry for 3001::13:1:1:4/128
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 4           4
Last Modified: Jan 15 17:13:50.032 for 03:54:01
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    3001::1:1:1:4 (metric 30) from 3001::1:1:1:4 (10.1.1.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 4
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:0:4:4b::, Behavior:18, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
```

The following examples show how to verify the current routes in the Routing Information Base (RIB):

```
RP/0/RSP0/CPU0:Node1# show route ipv6 3001::13:1:1:4/128
Fri Jan 15 21:08:05.499 UTC

Routing entry for 3001::13:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 15 17:13:50.431 for 03:54:15
  Routing Descriptor Blocks
    3001::1:1:1:4, from 3001::1:1:1:4
      Route metric is 0
  No advertising protos.

RP/0/RSP0/CPU0:Node1# show route ipv6 3001::13:1:1:4/128 detail
Fri Jan 15 21:08:22.628 UTC

Routing entry for 3001::13:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 15 17:13:50.431 for 03:54:32
  Routing Descriptor Blocks
    3001::1:1:1:4, from 3001::1:1:1:4
      Route metric is 0
```

```
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [base], SID-list {cafe:0:0:4:4b::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 4, Download Version 93
  No advertising protos.
```

The following examples show how to verify the current IPv6 Cisco Express Forwarding (CEF) table:

```
RP/0/RSP0/CPU0:Node1# show cef ipv6 3001::13:1:1:4/128
Fri Jan 15 21:08:41.483 UTC
3001::13:1:1:4/128, version 93, SRv6 Headend, internal 0x5000001 0x40 (ptr 0x78a100d4) [1],
 0x0 (0x0), 0x0 (0x8886b840)
 Updated Jan 15 17:13:50.433
 Prefix Len 128, traffic index 0, precedence n/a, priority 4
   via cafe:0:0:4::/128, 9 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78a0f504 0x0]
    next hop cafe:0:0:4::/128 via cafe:0:0:4::/64
    SRv6 H.Encaps.Red SID-list {cafe:0:0:4:4b::}

RP/0/RSP0/CPU0:Node1# show cef ipv6 3001::13:1:1:4/128 detail
Fri Jan 15 21:08:59.789 UTC
3001::13:1:1:4/128, version 93, SRv6 Headend, internal 0x5000001 0x40 (ptr 0x78a100d4) [1],
 0x0 (0x0), 0x0 (0x8886b840)
 Updated Jan 15 17:13:50.433
 Prefix Len 128, traffic index 0, precedence n/a, priority 4
  gateway array (0x7883b5d8) reference count 1, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x788e6c40) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Jan 15 17:13:50.433
 LDI Update time Jan 15 17:13:50.433

  Level 1 - Load distribution: 0
  [0] via cafe:0:0:4::/128, recursive

   via cafe:0:0:4::/128, 9 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78a0f504 0x0]
    next hop cafe:0:0:4::/128 via cafe:0:0:4::/64
    SRv6 H.Encaps.Red SID-list {cafe:0:0:4:4b::}

    Load distribution: 0 1 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/0        remote
    1     Y   HundredGigE0/0/0/1        remote
```

SRv6 Services: BGP Global IPv6

This feature extends support of SRv6-based BGP services to include Internet (IPv6) services by implementing End.DT6 SRv6 functions at the PE node, as defined in IETF draft "SRv6 BGP based Overlay services".

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv4 unicast address family
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.

BGP Global IPv6 Over SRv6 with Per-VRF SID Allocation Mode (End.DT6)

To configure BGP global IPv6 over SRv6, use the following commands:

- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 alloc mode per-vrf**: Specify the SID behavior (allocation mode).

  The **per-vrf** keyword specifies that the same label is be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 alloc mode** {**per-vrf** | **route-policy** *policy_name*}: Specify the SID behavior (allocation mode).
  - **per-vrf**: Specifies that the same label is be used for all the routes advertised from a unique VRF.
  - **route-policy** *policy_name*: Uses a route policy to determine the SID allocation mode and locator (if provided) for given prefix.
- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6 locator** *WORD*: Specify the locator
- **router bgp** *as-number* {**af-group** *WORD*| **neighbor-group** *WORD* | **neighbor** *ipv6-addr*} **address-family ipv6 unicast encapsulation-type srv6**: Specify the encapuslation type for SRv6.
  - Use **af-group** *WORD* to apply the SRv6 encapsulation type to the address family group for BGP neighbors.
  - Use **neighbor-group** *WORD*to apply the SRv6 encapsulation type to the neighbor group for Border Gateway Protocol (BGP) neighbors.
  - Use **neighbor** *ipv6-addr* to apply the SRv6 encapsulation type to the specific BGP neighbor.

Use Case 1: BGP Global IPv6 over SRv6 with Per-AFI SID Allocation

The following example shows how to configure BGP global IPv6 over SRv6 with per-VRF SID allocation.

```
Node1(config)# router bgp 100
Node1(config-bgp)# bgp router-id 10.1.1.1
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:4
Node1(config-bgp-nbr)# address-family ipv6 unicast
```

```
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# exit
Node1(config-bgp-nbr)# exit
Node1(config-bgp)# neighbor 3001::1:1:1:5
Node1(config-bgp-nbr)# address-family ipv6 unicast
Node1(config-bgp-nbr-af)# encapsulation-type srv6
Node1(config-bgp-nbr-af)# commit
```

Running Configuration

```
router bgp 100
 bgp router-id 10.1.1.1
 segment-routing srv6
  locator Node1
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator Node1
   alloc mode per-vrf
  !
 !
 neighbor 3001::1:1:1:4
  address-family ipv6 unicast
   encapsulation-type srv6
  !
 !
 neighbor 3001::1:1:1:5
  address-family ipv6 unicast
   encapsulation-type srv6
```

Use Case 2: BGP Global IPv6 over SRv6 with Per-Prefix SID Allocation

This use case provides the ability to assign a specific SRv6 locator for a given prefix or a set of prefixes. The egress PE advertises the prefix with the specified locator. This allows for per-prefix steering into desired transport behaviors, such as Flex Algo.

To assign an SRv6 locator for a specific prefix, configure a route policy to specify the SID allocation mode based on match criteria. Examples of match criteria are destination-based match or community-based match.

- Supported SID allocation modes are per-VRF and per-CE.
- For per-VRF allocation mode, you can also specify the SRv6 locator.
  - If an SRv6 locator is specified in the route policy, BGP will use that to allocate per-VRF SID. If the specified locator is invalid, the SID will not be allocated.
  - If an SRv6 locator is not specified in the route policy, the default locator is used to allocate the SID. If the default locator is not configured in BGP, then the SID will not be allocated.
- Per-CE allocation mode always uses the default locator to allocate the SID.

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

The following example shows a route policy specifying the SID allocation mode with destination-based match:

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (3001::1:1:1:1/128) then
Node1(config-rpl-if)# set srv6-alloc-mode per-vrf locator locator1
Node1(config-rpl-if)# elseif destination in (3001::2:2:2:2/128) then
```

```
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf locator locator2
Node1(config-rpl-elseif)# elseif destination in (3001::3:3:3:3/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf
Node1(config-rpl-elseif)# elseif destination in (3001::4:4:4:4/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-ce
Node1(config-rpl-elseif)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
```

The following example shows how to configure BGP global IPv6 over SRv6 with a route policy to determine the SID allocation mode for given prefix.

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family ipv4 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# alloc mode route-policy set_per_prefix_locator_rpl
```

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (3001::1:1:1:1/128) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (3001::2:2:2:2/128) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3001::3:3:3:3/128) then
    set srv6-alloc-mode per-vrf
  elseif destination in (3001::4:4:4:4/128) then
    set srv6-alloc-mode per-ce
  else
    drop
  endif
end-policy
!
router bgp 100
 address-family ipv6 unicast
  segment-routing srv6
   alloc mode route-policy set_per_prefix_locator_rpl
  !
 !
```

Verify that the local and received SIDs have been correctly allocated under BGP IPv6 address family:

```
Node1# show bgp ipv6 unicast local-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Local Sid                                Alloc mode   Locator
*> 3001::1:1:1:1/128  fc00:0:0:1:41::                          per-vrf      locator1
*> 3001::2:2:2:2/128  fc00:0:8:1:41::                          per-vrf      locator2
*> 3001::3:3:3:3/128  fc00:0:9:1:42::                          per-vrf      locator4
*> 3001::4:4:4:4/128  fc00:0:9:1:43::                          per-ce       locator4
*> 3001::5:5:5:5/128  NO SRv6 Sid                              -            -
* i3008::8:8:8:8/128  NO SRv6 Sid                              -            -


Node1# show bgp ipv6 unicast received-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network            Next Hop                              Received Sid
*> 3001::1:1:1:1/128  66.2.2.2                              NO SRv6 Sid
*> 3001::2:2:2:2/128  66.2.2.2                              NO SRv6 Sid
*> 3001::3:3:3:3/128  66.2.2.2                              NO SRv6 Sid
*> 3001::4:4:4:4/128  66.2.2.2                              NO SRv6 Sid
*> 3001::5:5:5:5/128  66.2.2.2                              NO SRv6 Sid
* i3008::8:8:8:8/128  77.1.1.2                              fc00:0:0:2:41::
```

Verification

The following examples show how to verify the BGP global IPv6 configuration using the **show bgp ipv6 unicast** commands.

```
RP/0/RSP0/CPU0:Node1# show bgp ipv6 unicast summary
Fri Jan 15 21:07:04.681 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.


Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker               4          4          4          4           4           0

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
3001::1:1:1:4     0   100    1502    1502        4    0    0 04:16:26          1
3001::1:1:1:5     0   100    1501    1501        4    0    0 04:14:47          1

RP/0/RSP0/CPU0:Node1# show bgp ipv6 unicast
Fri Jan 15 21:07:26.818 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 3001::13:1:1:1/128 ::                       0         32768 i
*>i3001::13:1:1:4/128 3001::1:1:1:4            0    100      0 i
*>i3001::13:1:1:5/128 3001::1:1:1:5            0    100      0 i

Processed 3 prefixes, 3 paths

RP/0/RSP0/CPU0:Node1# show bgp ipv6 unicast 3001::13:1:1:4/128
Fri Jan 15 21:07:50.309 UTC
BGP routing table entry for 3001::13:1:1:4/128
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                  4           4
```

```
Last Modified: Jan 15 17:13:50.032 for 03:54:01
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    3001::1:1:1:4 (metric 30) from 3001::1:1:1:4 (10.1.1.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 4
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:0:4:4b::, Behavior:18, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
```

The following examples show how to verify the current routes in the Routing Information Base (RIB):

```
RP/0/RSP0/CPU0:Node1# show route ipv6 3001::13:1:1:4/128
Fri Jan 15 21:08:05.499 UTC

Routing entry for 3001::13:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 15 17:13:50.431 for 03:54:15
  Routing Descriptor Blocks
    3001::1:1:1:4, from 3001::1:1:1:4
      Route metric is 0
  No advertising protos.

RP/0/RSP0/CPU0:Node1# show route ipv6 3001::13:1:1:4/128 detail
Fri Jan 15 21:08:22.628 UTC

Routing entry for 3001::13:1:1:4/128
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Jan 15 17:13:50.431 for 03:54:32
  Routing Descriptor Blocks
    3001::1:1:1:4, from 3001::1:1:1:4
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [base], SID-list {cafe:0:0:4:4b::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 4, Download Version 93
  No advertising protos.
```

The following examples show how to verify the current IPv6 Cisco Express Forwarding (CEF) table:

```
RP/0/RSP0/CPU0:Node1# show cef ipv6 3001::13:1:1:4/128
Fri Jan 15 21:08:41.483 UTC
3001::13:1:1:4/128, version 93, SRv6 Headend, internal 0x5000001 0x40 (ptr 0x78a100d4) [1],
 0x0 (0x0), 0x0 (0x8886b840)
 Updated Jan 15 17:13:50.433
 Prefix Len 128, traffic index 0, precedence n/a, priority 4
   via cafe:0:0:4::/128, 9 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78a0f504 0x0]
```

```
    next hop cafe:0:0:4::/128 via cafe:0:0:4::/64
    SRv6 H.Encaps.Red SID-list {cafe:0:0:4:4b::}

RP/0/RSP0/CPU0:Node1# show cef ipv6 3001::13:1:1:4/128 detail
Fri Jan 15 21:08:59.789 UTC
3001::13:1:1:4/128, version 93, SRv6 Headend, internal 0x5000001 0x40 (ptr 0x78a100d4) [1],
 0x0 (0x0), 0x0 (0x8886b840)
 Updated Jan 15 17:13:50.433
 Prefix Len 128, traffic index 0, precedence n/a, priority 4
  gateway array (0x7883b5d8) reference count 1, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x788e6c40) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Jan 15 17:13:50.433
 LDI Update time Jan 15 17:13:50.433

  Level 1 - Load distribution: 0
  [0] via cafe:0:0:4::/128, recursive

   via cafe:0:0:4::/128, 9 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78a0f504 0x0]
    next hop cafe:0:0:4::/128 via cafe:0:0:4::/64
    SRv6 H.Encaps.Red SID-list {cafe:0:0:4:4b::}

    Load distribution: 0 1 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/0        remote
    1     Y   HundredGigE0/0/0/1        remote
```

SRv6 Services on EVPN E-Line

Table 22: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services on EVPN E-Line	Release 24.3.1	Introduced in this release on: Fixed Systems (8200, 8700); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: P100]) Segment Routing over IPv6 (SRv6) services on Ethernet Virtual Private Network (EVPN) E-Line offers a modern approach to simplify and enhance network operations. SRv6 utilizes IPv6 addresses to encode segment routing information, seamlessly integrating with EVPN E-Line services to provide efficient, scalable, and flexible network solutions.

SRv6 Services over EVPN E-LINE

You can now configure Segment Routing over IPv6 (SRv6) services on an EVPN E-Line network. SRv6 leverages the Segment Routing Header in IPv6 packets to define paths and services. SRv6 services on EVPN E-Line provide point-to-point (P2P) Ethernet services over an SRv6 network. This setup leverages the benefits of both EVPN E-Line and SRv6 technologies to deliver efficient, scalable, and flexible network services.

For more information on EVPN E-Line, see the *EVPN Configuration Guide for Cisco 8000 Series Routers*.

Components and Functions of SRv6 Locators

SRv6 locators define the IPv6 address space used for segment routing. They represent the first part of a Segment Identifier (SID) and are used to identify a specific SRv6 node within a network. The locator is can be divided into two parts:

- **SID Block**: This is the SRv6 network designator, representing a fixed or known address space for an SRv6 domain.
- **Node ID**: This designates a specific node within the SRv6 network.

Priority Levels for SRv6 Locator Configuration in EVPN E-Line

For EVPN E-Line services, the SRv6 locators can be configured at different levels. When configuring the locators at different levels simultaneously, the priority is as follows:

- **Individual EVI Service Locator**: This locator applies to an individual EVI service and has the highest priority.
- **Specific EVI Locator**: This locator applies to all EVPN E-Line services for a specific EVI.
- **Global locator**: This is the default locator for all EVPN E-Line services, with the lowest priority.

Benefits of SRv6 Services on EVPN E-line

- **Load Balancing**: Distributes traffic across multiple PEs, enhancing network efficiency and performance.
- **Redundancy**: Provides failover capabilities, ensuring continuous service availability.
- **Scalability**: Easily scales to accommodate large networks with numerous nodes.
- **Simplified Operations**: Reduces complexity by eliminating the need for traditional protocols like LDP and RSVP.
- **Enhanced Automation**: Supports comprehensive automation strategies for network management and optimization.

Restrictions for SRv6 Services on EVPN E-Line

The SRv6 services are supported on:

- EVPN E-Line single-homing mode on the following routers:
  - Fixed Systems (8201-32FH)
  - Centralized Systems (8600)

- EVPN E-Line single-homing and multi-homing modes on the following routers:
  - Fixed Systems (8212-48FH-M, 8711-32FH-M)
  - Modular Systems (8800 [LC ASIC: P100])
- SRv6 Services on EVPN E-Line are not supported on routers with Q200-based line cards.
- Both the format1 and Micro-SID with f3216. These formats are used to encode multiple segments within an IPv6 header. For more information on the formats, see Configure Segment Routing over IPv6 (SRv6) with Micro-SIDs, on page 7.

Configure SRv6 Services on EVPN E-Line

To configure SRv6 Services on EVPN E-Line:

- Configure address family session in BGP.
- Configure Segment Routing and SRv6 locator on EVPN.
- Configure EVPN E-Line and associate the SRv6 locator with the EVPN E-Line instance.

The following example shows configuration of global locator.

Procedure

Step 1 Configure address family session in BGP.

```
Router(config)#router bgp 100
Router(config-bgp)#bgp router-id 172.16.0.1
Router(config-bgp)#address-family l2vpn evpn
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor 2001:db8:1::1
Router(config-bgp-nbr)#remote-as 100
Router(config-bgp-nbr)#update-source loopback0
Router(config-bgp-nbr)#address-family l2vpn evpn
Router(config-bgp-nbr-af)#root
```

Step 2 Configure Segment Routing and SRv6 locator on EVPN.

```
Router(config)#evpn
Router(config-evpn)#interface tengigE 0/0/0/0
Router(config-evpn-ac)#exit
Router(config-evpn)#segment-routing srv6
Router(config-evpn-srv6)#locator POD0
Router(config-evpn-srv6-locator)#root
```

Step 3 Configure EVPN E-Line and associate the SRv6 locator with the EVPN E-Line instance.

```
Router(config)#l2vpn
Router(config-l2vpn)#xconnect group 2
Router(config-l2vpn-xc)#p2p 2
Router(config-l2vpn-xc-p2p)#interface tengigE 0/0/0/0.2
Router(config-l2vpn-xc-p2p)#neighbor evpn evi 2 service 2 segment-routing srv6
```

Step 4 Use the **show running** command to view the running configuration.

```
Router# show running configuration
router bgp 100
  bgp router-id 172.16.0.1
  address-family l2vpn evpn
  !
  neighbor 2001:db8:1::1
   remote-as 100
   update-source Loopback0
  address-family l2vpn evpn

  evpn
 interface TenGigE0/0/0/0
 !
 segment-routing srv6
  locator POD0
 !
!
l2vpn
 xconnect group 2
  p2p 2
   interface TenGigE0/0/0/0.2
   neighbor evpn evi 11001 service 2002 segment-routing srv6
```

Step 5 Verify the SRv6 configuration using the following show commands. The outputs show the status of the global locator, which is the default locator.

```
Router#show segment-routing srv6 locator POD0 sid
SID                         Behavior     Context                          Owner               State
  RW
--------------------------  -----------  -------------------------------  ------------------  -----
  --
2001:db8:1::1               End.DX2      'default':1                      sidmgr              InUse
  Y

Router#show evpn segment-routing srv6 detail
Configured default locator: POD0
EVIs with unknown locator config: 0
VPWS with unknown locator config: 0

Locator name       Prefix                    OOR      Service count    SID count
------------       ------                    ---      -------------    ---------
POD0              2001:db8:1::/48            False    1                1
   Default locator
```

Step 6 Use the **show l2vpn** command to view a detailed information about a specific L2VPN P2P service for a VPWS instance.

Example:

```
Router#show l2vpn xconnect group xg2001 xc-name vpws-20010001 detail
Thu Aug 22 18:27:14.344 UTC

Group xg2001, XC vpws-20010001, state is up; Interworking none
  AC: Bundle-Ether201.20010001, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 2001
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 9194; XC ID 0xa00003e9; interworking none
    Statistics:
      packets: received 14089, sent 11603
      bytes: received 1803392, sent 1461978
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor ::ffff:10.0.0.5, PW ID: evi 2001, ac-id 1, state is up ( established )
    XC ID 0xc00003e9
    Encapsulation SRv6
```

```
Encap type Ethernet
Ignore MTU mismatch: Enabled
Transmit MTU zero: Enabled
Reachability: Up
Nexthop type: Internal ID ::ffff:10.0.0.5

  SRv6              Local                        Remote
  ----------------  ---------------------------- --------------------------
  uDX2              fccc:cc00:1:e001::            fccc:cc00:2:e001::
  AC ID             1                            1
  MTU               9208                         0
  Locator           locator0                     N/A
  Locator Resolved  Yes                          N/A
  SRv6 Headend      H.Encaps.L2.Red              N/A
Statistics:
  packets: received 11603, sent 14089
  bytes: received 1461978, sent 1803392
```

Step 7 Use the **show evpn internal-id** command to view the internal ID mappings for the EVPN instance.

Example:

```
Router#show evpn internal-id
Mon Aug 12 16:56:18.218 UTC


VPN-ID     Encap  Ethernet Segment Id         EtherTag     Internal ID
---------- ------ --------------------------- ----------   -----------

2001       SRv6   2::1                        1            ::ffff:10.0.0.5
   Summary pathlist (ID 0x0000000000000201):
 0x05000007 (P) 2::1                                      fccc:cc00:2:e001::
```

SRv6 SID Information in BGP-LS Reporting

BGP Link-State (LS) is used to report the topology of the domain using nodes, links, and prefixes. This feature adds the capability to report SRv6 Segment Identifier (SID) Network Layer Reachability Information (NLRI).

The following NLRI has been added to the BGP-LS protocol to support SRv6:

- Node NLRI: SRv6 Capabilities, SRv6 MSD types
- Link NLRI: End.X, LAN End.X, and SRv6 MSD types
- Prefix NLRI: SRv6 Locator
- SRv6 SID NLRI (for SIDs associated with the node): Endpoint Function, BGP-EPE Peer Node/Set and Opaque

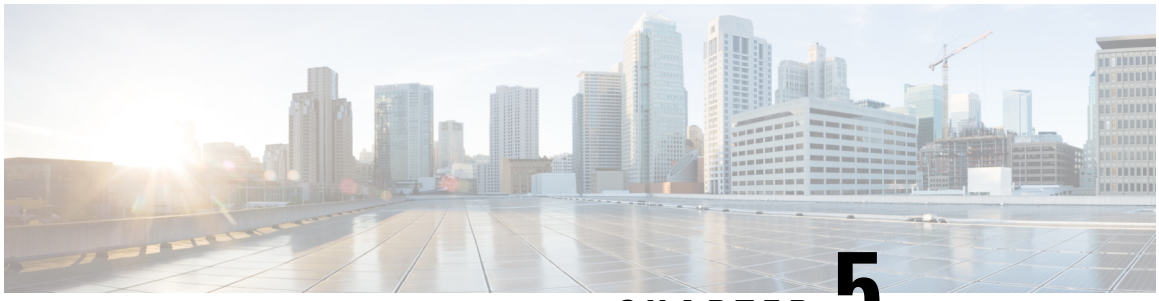
This example shows how to distribute IS-IS SRv6 link-state data using BGP-LS:

```
Router(config)# router isis 200
Router(config-isis)# distribute link-state instance-id 200
```

Note It is still possible to ping or trace a SID:

- **ping** B:k:F::
- **traceroute** B:k:F::

It is possible to use a list of packed carriers to ping or trace a SID, to ping or trace route, use <destination SID> via srv6-carriers <list of packed carriers>



CHAPTER 5

Configure SRv6 Traffic Engineering

This module provides information about Segment Routing over IPv6 (SRv6) Traffic Engineering, how to configure SRv6-TE, and how to steer traffic into an SRv6-TE policy.

Table 23: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Traffic Engineering	Release 7.10.1	

Feature Name	Release Information	Feature Description
		You can now control the traffic flows within the network by defining the explicit and dynamic paths for traffic flows using the Segment Identifier (SID) within the IPv6 packet header. Defining explicit and dynamic paths based on different attributes and constraints allow the router to optimize routing decisions and enhance resource utilization. SRv6-TE policies supports the following functionalities: • SRv6-TE with SRv6 micro-SIDs (uSIDs) • Explicit SRv6 policies • Automated steering for Layer 3-based BGP services (IPv4 L3VPN, IPv6 L3VPN, IPv4 BGP global, IPv6 BGP global) • SRv6-aware Path Computation Element (PCE) • PCEPv4 and PCEPv6 • Path computation optimization objectives (TE, IGP, latency) • Path computation constraints (affinity, disjointness) This feature introduces the following changes: **CLI:** • **policy srv6 locator** • **segment-routing traffic-eng srv6** • **srv6 locator** • **srv6 maximum-sid-depth** • **segment-lists segment-list** • **segment-lists srv6** **YANG Data Model:**

Feature Name	Release Information	Feature Description
		• Cisco-IOS-XR-segment-routing-ms-cfg (see GitHub, YANG Data Models Navigator)

- SRv6-TE Overview, on page 192
- Usage Guidelines and Limitations, on page 192
- SRv6-TE Policy, on page 194
- **SRv6 Flexible Algorithm**, on page 209
- Automated Steering, on page 213
- Protocols, on page 214
- SR-TE Application Programming Interface (API), on page 221
- Reporting of SR-TE Policies Using BGP- Link State , on page 230

SRv6-TE Overview

Segment Routing over IPv6 Traffic Engineering (SRv6-TE) allows you to steer traffic across a network based on specific policies and requirements and provides greater control over how traffic flows through the network.

SRv6-TE also allows you to create explicit paths through the network for specific traffic flows where a particular application or service requires a specific quality of service (QoS) level, such as low latency or high bandwidth.

SRv6-TE uses the concept of source routing, where the source calculates the path and encodes it in the packet header as a list of segments. This list of segments is added to an IPv6 routing header called the SRv6 Segment Routing Header (SRH) in the incoming packet. With SRv6-TE, the network does not need to maintain per-application and per-flow state on each node. Instead, only the head-end nodes on the edge of the network where the traffic enters the policy need to maintain a state.

The remaining nodes obey the forwarding instructions that are included in the packet. SRv6-TE can utilize network bandwidth more effectively than traditional MPLS RSVP-TE using ECMP within each segment. In addition, by using a single intelligent source it relieves remaining routers from the task of calculating the required path through the network.

Traffic engineering over SRv6 can be accomplished in the following ways:

- End-to-End Flexible Algorithm: This is used for traffic engineering intents achieved with Flexible Algorithm, including low latency, multi-plane disjointness, affinity inclusion/exclusion, and SRLG exclusion. See SRv6 Flexible Algorithm, on page 209.
- SRv6-TE Policy: This is used for traffic engineering intents beyond Flex Algo capabilities, such as path disjointness that rely on path computation by a PCE. In addition, this is used for user-configured explicit paths. SRv6-TE Policy, on page 194.

Usage Guidelines and Limitations

The following are the usage guidelines and limitations for SRv6 policy.

Supported Features

The following are the supported functionalities:

- SRv6 policies with SRv6 uSID segments
- SRv6 policies with PCE-delegated dynamic paths
- SRv6 policies with explicit paths
- SRv6 policies with single or multiple candidate paths (CP)
- SRv6 policies with a single SID list per CP
- SRv6 policies with multiple (weighted ECMP) SID lists per CP
- Path delegation and reporting with PCEPv4
- Path delegation and reporting with PCEPv6
- PCEPv4 and PCEPv6 sessions with different PCEs
- PCEs with PCEP state-sync sessions must be either PCEPv4 or PCEPv6.
- PCEP path delegation with the following optimization objective (metric) types:
  - IGP metric
  - TE metric
  - Delay (latency)
  - Hop-count
- PCEP path delegation with the following constraint types:
  - Affinity (include-any, include-all, exclude-any)
  - Path disjointness (link, node, SRLG, SRLG+node)
    - For path-computation on PCE, configuring both affinity and disjoint-path is not supported.
  - Segment protection-type:
    - Protected-only
    - Protected-preferred
    - Unprotected-only
    - Unprotected-preferred
- SRv6 policy with TI-LFA (protection of the first segment in the segment list at the head-end)
- Steering over SRv6 policies with Automated Steering for the following services:
  - L3 BGP-based services (IPv4 L3VPN, IPv6 L3VPN, IPv4 BGP global, IPv6 BGP global)

Unsupported Features

The following functionalities are not supported:

- SRv6 policy counters
- PCE-initiated SRv6 policies via PCEP
- SRv6 policies with head-end computed dynamic paths
- PCE path delegation with segment-type Flex Algo constraint
- PCEPv4 and PCEPv6 sessions with same PCE
- Steering over SRv6 policies based on incoming BSID (remote automated steering)
- PCC with user-configured PCE groups
- SR-PM delay-measurement over SRv6 policies
- SR-PM liveness detection over SRv6 policies
- L3 services with BGP PIC over SRv6 policies
- SRv6 policy ping
- L2 BGP-based service (EVPN VPWS)

By default, 1K polices are supported. If MPLS tunnels are not configured on the router, use the following commands to get higher SRv6-TE scale:

- If the support for LDPoTE or SRoSR-TE MPLS features are not required, use the following command to configure higher RSVP-TE/SR-TE scale.

  hw-module profile cef te-tunnel highscale-no-ldp-over-te

- If the support for LDPoTE is required but not SRoSR-TE MPLS features, use the following command to configure higher tunnel scale.

  hw-module profile cef te-tunnel highscale-ldp-over-te-no-sr-over-srte

SRv6-TE Policy

SRv6-TE uses a “policy” to steer traffic through the network. An SRv6-TE policy path is expressed as a list of micro-segments that specifies the path, called a micro-segment ID (uSID) list. Each segment list 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 following the shortest path calculated by the IGP. If a packet is steered into an SRv6-TE policy, the uSID list is pushed on the packet by the head-end. The rest of the network executes the instructions embedded in the uSID list.

An SRv6-TE policy is identified as an ordered list (head-end, color, end-point):

- Head-end – Where the SRv6-TE policy is instantiated
- Color – A numerical value that distinguishes between two or more policies to the same node pairs (Head-end – End point)
- End-point – The destination of the SRv6-TE policy

Every SRv6-TE policy has a color value. Every policy between the same node pairs requires a unique color value.

An SRv6-TE policy uses one or more candidate paths. A candidate path can be made up of a single SID-list or a set of weighted SID-lists (for weighted equal cost multi-path [WECMP]).

A SID list can be either the result of a dynamic path computation by a PCE or a user-configured explicit path. See SRv6-TE Policy for more information.

SRv6-TE Policy Path Types

The following SRv6-TE policy path types are supported:

Explicit Paths

An **explicit** path is a specified SID-list or set of SID-lists.

When configuring an explicit path using IP addresses of links along the path, the SRv6-TE process prefers the protected Adj-SID of the link, if one is available. In addition, when manual Adj-SIDs are configured, the SRv6-TE process prefers a manual-protected Adj-SID over a dynamic-protected Adj-SID.

You can configure the path to prefer the protected or unprotected Adj-SID, or to use only protected or unprotected Adj-SID. See Segment Protection-Type Constraint, on page 208.

You can enable SRv6-TE explicit segment list SID validation to allow the head-end node to validate the SIDs in an explicit SRv6-TE segment list against the SR-TE topology database. See SRv6-TE Explicit Segment List SID Validation, on page 199.

A segment list can use uSIDs or uSID carrier, or a combination of both.

Configure SRv6-TE Policy with Explicit Path

To configure an SRv6-TE policy with an explicit path, complete the following configurations:

1. Create the segment lists with SRv6 segments.
2. Create the SRv6-TE policy.

Create a segment list with SRv6 uSIDs:

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# segment-lists
Router(config-sr-te-segment-lists)# srv6
Router(config-sr-te-sl-global-srv6)# sid-format usid-f3216
Router(config-sr-te-sl-global-srv6)# exit
Router(config-sr-te-segment-lists)# segment-list p1_r8_1
Router(config-sr-te-sl)# srv6
Router(config-sr-te-sl-srv6)# index 10 sid FCBB:BB00:10:feff::
Router(config-sr-te-sl-srv6)# index 15 sid FCBB:BB00:100:fe00::
Router(config-sr-te-sl-srv6)# index 20 sid FCBB:BB00:2::
Router(config-sr-te-sl-srv6)# index 30 sid FCBB:BB00:3::
Router(config-sr-te-sl-srv6)# index 40 sid FCBB:BB00:4::
Router(config-sr-te-sl-srv6)# index 50 sid FCBB:BB00:5::
Router(config-sr-te-sl-srv6)# index 60 sid FCBB:BB00:6::
```

Create the SRv6-TE policy:

```
Router(config-sr-te)# policy POLICY1
Router(config-sr-te-policy)# srv6 locator loc1 binding-sid dynamic behavior ub6-encaps-reduced
```

```
Router(config-sr-te-policy)# color 10 end-point ipv6 FCBB:BB00:2::1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list p1_r8_1
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# exit
```

Running Configuration

```
Router# show running-config

. . .

segment-routing
 traffic-eng
  segment-lists
   srv6
    sid-format usid-f3216
   !
   segment-list p1_r8_1
    srv6
     index 10 sid FCBB:BB00:10:feff::
     index 15 sid FCBB:BB00:100:fe00::
     index 20 sid FCBB:BB00:2::
     index 30 sid FCBB:BB00:3:::
     index 40 sid FCBB:BB00:4::
     index 50 sid FCBB:BB00:5::
     index 60 sid FCBB:BB00:6::
    !
   !
  !
  policy POLICY1
   srv6
    locator loc1 binding-sid dynamic behavior ub6-encaps-reduced
   !
   color 10 end-point ipv6 fcbb:bb00:2::1
   candidate-paths
    preference 100
     explicit segment-list p1_r8_1
     !
    !
   !
  !
 !
!
```

Verification

Verify the SR-TE policy configuration using:

```
Router# show segment-routing traffic-eng policy name srte_c_10_ep_fcbb:bb00:2::1 d

SR-TE policy database
---------------------

Color: 10, End-point: fcbb:bb00:2::1
  Name: srte_c_10_ep_fcbb:bb00:2::1
  Status:
    Admin: up  Operational: down for 00:01:30 (since Oct 31 21:33:24.090)
  Candidate-paths:
    Preference: 100 (configuration) (inactive)
```

```
    Name: POLICY1
    Requested BSID: dynamic
    Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 13
    Explicit: segment-list p1_r8_1 (inactive)
      Weight: 1, Metric Type: TE
        SID[0]: FCBB:BB00:10:feff::
        SID[1]: FCBB:BB00:100:fe00::
        SID[2]: FCBB:BB00:1::
        SID[3]: FCBB:BB00:1:fe00::
        SID[4]: FCBB:BB00:fe00::
        SID[5]: FCBB:BB00:5::
        SID[6]: FCBB:BB00:6::
    SRv6 Information:
      Locator: loc1
      Binding SID requested: Dynamic
      Binding SID behavior: End.B6.Encaps.Red
  Attributes:
    Forward Class: 0
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
```

Candidate Paths with Weighted SID Lists

If a candidate path is associated with a set of segment lists, each segment list is associated with weight for weighted load balancing. Valid values for weight are from 1 to 4294967295; the default weight is 1.

If a set of segment lists is associated with the active path of the policy, then the steering is per-flow and weighted-ECMP (W-ECMP) based according to the relative weight of each segment list.

The fraction of the flows associated with a given segment list is *w*/*Sw*, where *w* is the weight of the segment list and *Sw* is the sum of the weights of the segment lists of the selected path of the SR policy.

When a composite candidate path is active, the fraction of flows steered into each constituent SR policy is equal to the relative weight of each constituent SR policy. Further load balancing of flows steered into a constituent SR policy is performed based on the weights of the segment list of the active candidate path of that constituent SR policy.

Configuration Example

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# segment-lists
Router(config-sr-te-segment-lists)# srv6
Router(config-sr-te-sl-global-srv6)# sid-format usid-f3216
Router(config-sr-te-sl-global-srv6)# exit
Router(config-sr-te-segment-lists)# segment-list p1_r8_3
Router(config-sr-te-sl)# srv6
Router(config-sr-te-sl-srv6)# index 10 sid FCBB:BB00:10:fe01::
Router(config-sr-te-sl-srv6)# index 20 sid FCBB:BB00:1::
Router(config-sr-te-sl-srv6)# index 30 sid FCBB:BB00:1:fe00::
Router(config-sr-te-sl-srv6)# index 40 sid FCBB:BB00:fe00::
Router(config-sr-te-sl-srv6)# index 50 sid FCBB:BB00:5::
Router(config-sr-te-sl-srv6)# index 60 sid FCBB:BB00:6::
Router(config-sr-te-segment-lists)# segment-list igp_ucmp1
Router(config-sr-te-sl)# srv6
```

```
Router(config-sr-te-sl-srv6)# index 10 sid FCBB:BB00:1::
Router(config-sr-te-sl-srv6)# index 20 sid FCBB:BB00:4::
Router(config-sr-te-sl-srv6)# index 30 sid FCBB:BB00:5::
Router(config-sr-te-sl-srv6)# exit
Router(config-sr-te-sl)# exit
Router(config-sr-te-segment-lists)# exit

Router(config-sr-te)# policy po_r8_1001
Router(config-sr-te-policy)# srv6 locator loc1 binding-sid dynamic behavior ub6-encaps-reduced
Router(config-sr-te-policy)# color 1001 end-point ipv6 FCBB:BB00:2::1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 1000
Router(config-sr-te-policy-path-pref)# explicit segment-list p1_r8_3
Router(config-sr-te-pp-info)# weight 4
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# explicit segment-list igp_ucmp1
Router(config-sr-te-pp-info)# weight 2
Router(config-sr-te-pp-info)# exit
```

Running Configuration

```
Router# show running-config

. . .

segment-routing
 traffic-eng
  segment-lists
   srv6
    sid-format usid-f3216
   !
   segment-list p1_r8_3
    srv6
     index 10 sid FCBB:BB00:10:fe01::
     index 20 sid FCBB:BB00:1::
     index 30 sid FCBB:BB00:1:fe00::
     index 40 sid FCBB:BB00:fe00::
     index 50 sid FCBB:BB00:5::
     index 60 sid FCBB:BB00:6::
    !
   !
   segment-list igp_ucmp1
    srv6
     index 10 sid FCBB:BB00:1::
     index 20 sid FCBB:BB00:4::
     index 30 sid FCBB:BB00:5::
    !
   !
  !
  policy po_r8_1001
   srv6
    locator loc1 binding-sid dynamic behavior ub6-encaps-reduced
   !
   color 1001 end-point ipv6 fcbb:bb00:2::1
   candidate-paths
    preference 1000
     explicit segment-list p1_r8_3
      weight 4
     !
     explicit segment-list igp_ucmp1
      weight 2
     !
```

```
    !
   !
  !
 !
!
```

SRv6-TE Explicit Segment List SID Validation

SRv6-TE explicit segment list SID validation evaluates whether the explicit segment list of an SR policy's candidate path is valid and therefore usable. The head-end node validates if the hops are in the SR-TE topology database.

When enabled, the segment list SID validation occurs before programming the SR policy and after an IGP topology change.

When the segment list validation fails, then the segment list is declared invalid. An SR policy candidate path is declared invalid when it has no valid segment lists. An SR policy is declared invalid when it has no valid candidate paths.

Usage Guidelines and Limitations

- Segment list SID validation can be enabled globally for all SRv6 explicit segment lists or for a specific SRv6 segment list.
- If SIDs in an explicit segment list are expected to not be found in the headend (for example, a multi-domain case), the topology check can be disabled for that segment list.
- The SR-TE topology should distributed from IGP (for intra-domain paths) or from BGP-LS (for multi-domain paths).
- The SRv6 segment list in an explicit candidate path of an SRv6 policy cannot be empty.
- All segments in a segment list must be of same data plane type: SRv6 or SR-MPLS.
- Top SID validation occurs by performing path resolution using the top SID.
- All SIDs in segment list are validated against local SID block and SID format.
- A segment list is validated against MSD.

Enable SID validation globally for all SRv6 explicit segment lists

Prior to enabling SRv6 explicit segment list SID validation, use the **show segment-routing traffic-eng topology** command to verify if the SR-TE topology is available on the headend PCC router.

To enable SID validation globally, use the **segment-routing traffic-eng segment-lists srv6 topology-check** command.

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# segment-lists
Router(config-sr-te-segment-lists)# srv6
Router(config-sr-te-sl-global-srv6)# topology-check
```

To enable SID validation for a specific explicit SID list, use the **segment-routing traffic-eng segment-lists segment-list** *name* **srv6 topology-check** command.

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# segment-lists
```

```
Router(config-sr-te-segment-lists)# segment-list p1_r8_1
Router(config-sr-te-sl)# srv6
Router(config-sr-te-sl-srv6)# topology-check
```

The following example shows how to enable SID validation globally and disable SID validation for a specific SRv6 explicit segment list:

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# segment-lists
Router(config-sr-te-segment-lists)# srv6
Router(config-sr-te-sl-global-srv6)# topology-check
Router(config-sr-te-sl-global-srv6)# exit
Router(config-sr-te-segment-lists)# segment-list p1_r8_1
Router(config-sr-te-sl)# srv6
Router(config-sr-te-sl-srv6)# no topology-check
```

Running Configuration

```
segment-routing
 traffic-eng
  segment-lists
   srv6
    topology-check
   !
   segment-list p1_r8_1
    srv6
     no topology-check
    !
   !
  !
 !
!
```

Verification

```
Router# show segment-routing traffic-eng policy name srte_c_10_ep_fcbb:bb00:2::1 detail

SR-TE policy database
---------------------

Color: 10, End-point: fcbb:bb00:2::1
  Name: srte_c_10_ep_fcbb:bb00:2::1
  Status:
    Admin: up  Operational: down for 00:04:12 (since Nov  7 19:24:21.396)
  Candidate-paths:
    Preference: 100 (configuration) (inactive)
      Name: POLICY1
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 13
      Explicit: segment-list p1_r8_1 (inactive)
        Weight: 1, Metric Type: TE
          SID[0]: fccc:0:10:feff::
          SID[1]: fccc:0:100:fe00::
          SID[2]: fccc:0:1::
          SID[3]: fccc:0:1:fe00::
          SID[4]: fccc:0:fe00::
          SID[5]: fccc:0:5::
```

```
        SID[6]: fccc:0:6::
    SRv6 Information:
      Locator: loc1
      Binding SID requested: Dynamic
      Binding SID behavior: End.B6.Encaps.Red
Attributes:
  Forward Class: 0
  Steering labeled-services disabled: no
  Steering BGP disabled: no
  IPv6 caps enable: yes
  Invalidation drop enabled: no
  Max Install Standby Candidate Paths: 0
```

Dynamic Paths

A **dynamic** path is based on an optimization objective and a set of constraints. The head-end computes a solution, resulting in a SID-list or a set of SID-lists. When the topology changes, a new path is computed. If the head-end does not have enough information about the topology, the head-end might delegate the computation to a Segment Routing Path Computation Element (SR-PCE). For information on configuring SR-PCE, see *Configure Segment Routing Path Computation Element* chapter.

An SRv6-TE policy initiates a single (selected) path in RIB/FIB. This is the preferred valid candidate path.

A candidate path has the following characteristics:

- It has a preference – If two policies have same {color, endpoint} but different preferences, the policy with the highest preference is selected.
- It is associated with a single Binding SID (uB6) – A uB6 SID conflict occurs when there are different SRv6 policies with the same uB6 SID. In this case, the policy that is installed first gets the uB6 SID and is selected.
- It is valid if it is usable.

A path is selected when the path is valid and its preference is the best among all candidate paths for that policy.

Note The protocol of the source is not relevant in the path selection logic.

For a dynamic path that traverses a specific interface between nodes (segment), the algorithm may encode this segment using an Adjacency uSID (uA SID).

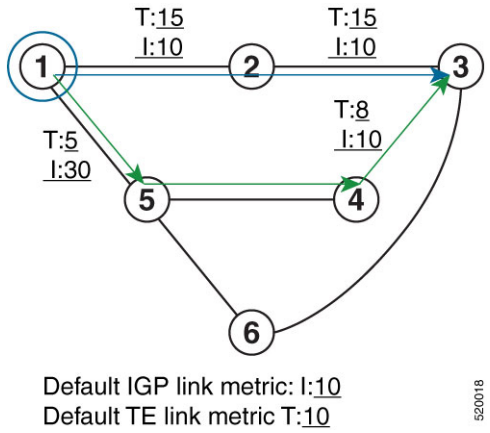
Optimization Objectives

Optimization objectives allow the head-end router to compute a uSID-list that expresses the shortest dynamic path according to the selected metric type:

- Hopcount — Use the least number of hops for path computation.
- IGP metric — Refer to the *Implementing IS-IS* and *Implementing OSPF* chapters in the *Routing Configuration Guide for Cisco 8000 Series Routers*
- TE metric — See the Configure Interface TE Metrics, on page 202 section for information about configuring TE metrics.

- Delay (latency) — See the Configure Performance Measurement, on page 539chapter for information about measuring delay for links or SRv6 policies.

This example shows a dynamic path from head-end router 1 to end-point router 3 that minimizes IGP or TE metric:



- The blue path uses the minimum IGP metric: Min-Metric (1 → 3, IGP) = uSID-list {cafe:0:**3**::}; cumulative IGP metric: 20
- The green path uses the minimum TE metric: Min-Metric (1 → 3, TE) = uSID-list {cafe:0:**5**:**4**:**3**::}; cumulative TE metric: 23

Configure Interface TE Metrics

To configure the TE metric for interfaces, use the **metric** *value* command. The *value* range is from 0 to 2147483647.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# interface type interface-path-id
Router(config-sr-te-if)# metric value
```

Running Configuration

The following configuration example shows how to set the TE metric for various interfaces:

```
segment-routing
 traffic-eng
  interface TenGigE0/0/0/0
   metric 100
  !
  interface TenGigE0/0/0/1
   metric 1000
  !
  interface TenGigE0/0/2/0
   metric 50
  !
 !
end
```

Constraints

Constraints allow the head-end router to compute a dynamic path according to the selected metric type:

Note For path-computation on PCE, configuring both affinity and disjoint-path is not supported.

- Affinity — You can apply a color or name to links or interfaces by assigning affinity bit-maps to them. You can then specify an affinity (or relationship) between an SRv6 policy path and link colors. SRv6-TE computes a path that includes or excludes links that have specific colors,or combinations of colors. See the Named Interface Link Admin Groups and SRv6-TE Affinity Maps, on page 206section for information on named interface link admin groups and SRv6-TE Affinity Maps.
- Disjoint — SRv6-TE computes a path that is disjoint from another path in the same disjoint-group. Disjoint paths do not share network resources. Path disjointness may be required for paths between the same pair of nodes, between different pairs of nodes, or a combination (only same head-end or only same end-point).
- Segment protection-type behavior — You can control whether protected or unprotected segments are used when encoding the SID-list of an SRv6 policy candidate path. The types of segments that could be used when building a SID-list include uSIDs and adjacency SIDs (uA SID). See the Segment Protection-Type Constraint, on page 208 for details and usage guidelines.

Configure SRv6 Policy with Dynamic Path

To configure a SRv6-TE policy with a dynamic path, optimization objectives, and affinity constraints, complete the following configurations:

- Create the SRv6-TE Policy and configure the SRv6-TE policy color and IPv6 end-point address.
- (optional) Specify Customized Locator and Source Address

Note
- If you do not specify a customized per-policy locator and BSID behavior, the policy will use the global locator and BSID behavior.
- If you do not specify a customized per-policy source address, the policy will use the local IPv6 source address.

- Enable Dynamic Path Computed by the SR-PCE
- Configure Dynamic Path Optimization Objectives
- Configure Dynamic Path Constraints

Note Disjoint-path and affinity constraints cannot be configured at the same time.

Configuration Example

The following example shows a configuration of an SRv6 policy at an SRv6-TE head-end router using the global locator and source address. The policy has a dynamic path with optimization objectives and affinity constraints computed by the SR-PCE.

```
Node1(config)# segment-routing
Node1(config-sr)# traffic-eng
Node1(config-sr-te)# srv6

/* Specify customized locator and source address */
Node1(config-sr-te-srv6)# locator Node1 binding-sid dynamic behavior ub6-encaps-reduced
Node1(config-sr-te-srv6)# exit
Node1(config-sr-te)# candidate-paths
Node1(config-sr-te-candidate-path)# all

/* Specify the customized IPv6 source address for candidate paths */
Node1(config-sr-te-candidate-path-type)# source-address ipv6 cafe:0:1::1
Node1(config-sr-te-candidate-path-type)# exit
Node1(config-sr-te-candidate-path)# exit

/* Create the SRv6-TE policy */
Node1(config-sr-te)# policy pol_node1_node4_te
Node1(config-sr-te-policy)# color 20 end-point ipv6 cafe:0:4::4

/* Enable dynamic path computed by the SR-PCE */
Node1(config-sr-te-policy)# candidate-paths
Node1(config-sr-te-policy-path)# preference 100
Node1(config-sr-te-policy-path-pref)# dynamic
Node1(config-sr-te-pp-info)# pcep
Node1(config-sr-te-path-pcep)# exit

/* Configure dynamic Path optimization objectives */
Node1(config-sr-te-pp-info)# metric type te
Node1(config-sr-te-pp-info)# exit

/* Configure dynamic path constraints*/
Node1(config-sr-te-policy-path-pref)# constraints
Node1(config-sr-te-path-pref-const)# affinity
Node1(config-sr-te-path-pref-const-aff)# exclude-any
Node1(config-sr-te-path-pref-const-aff-rule)# name brown
```

Running Confiuration

```
segment-routing
 traffic-eng
  srv6
   locator Node1 binding-sid dynamic behavior ub6-encaps-reduced
  !
  candidate-paths
   all
    source-address ipv6 cafe:0:1::1
   !
  !
  policy pol_node1_node4_te
   color 20 end-point ipv6 cafe:0:4::4
   candidate-paths
    preference 100
     dynamic
      pcep
      !
      metric
```

```
      type te
     !
    !
    constraints
     affinity
      exclude-any
       name brown
      !
     !
    !
   !
  !
 !
!
```

The following example shows a configuration of a manual SRv6 policy at an SRv6-TE head-end router with customized locator and source address. The policy has a dynamic path with optimization objectives and affinity constraints computed by the SR-PCE.

```
Node1(config)# segment-routing
Node1(config-sr)# traffic-eng
Node1(config-sr-te)# policy pol_node1_node4_te
Node1(config-sr-te-policy)# source-address ipv6 cafe:0:1::1
Node1(config-sr-te-policy)# srv6
Node1(config-sr-te-policy-srv6)# locator Node1 binding-sid dynamic behavior ub6-encaps-reduced
Node1(config-sr-te-policy-srv6)# exit
Node1(config-sr-te-policy)# color 20 end-point ipv6 cafe:0:4::4
Node1(config-sr-te-policy)# candidate-paths
Node1(config-sr-te-policy-path)# preference 100
Node1(config-sr-te-policy-path-pref)# dynamic
Node1(config-sr-te-pp-info)# pcep
Node1(config-sr-te-path-pcep)# exit
Node1(config-sr-te-pp-info)# metric type te
Node1(config-sr-te-pp-info)# exit
Node1(config-sr-te-policy-path-pref)# constraints
Node1(config-sr-te-path-pref-const)# affinity
Node1(config-sr-te-path-pref-const-aff)# exclude-any
Node1(config-sr-te-path-pref-const-aff-rule)# name brown
```

Running Config

```
segment-routing
 traffic-eng
  policy pol_node1_node4_te
   srv6
    locator Node1 binding-sid dynamic behavior ub6-encaps-reduced
   !
   source-address ipv6 cafe:0:1::1
   color 20 end-point ipv6 cafe:0:4::4
   candidate-paths
    preference 100
     dynamic
      pcep
      !
      metric
       type te
      !
     !
     constraints
      affinity
       exclude-any
        name brown
```

```
        !
       !
      !
     !
    !
   !
  !
 !
```

Verification

```
Node1# show segment-routing traffic-eng policy color 20

SR-TE policy database
---------------------
Color: 20, End-point: cafe:0:4::4
  Name: srte_c_20_ep_cafe:0:4::4
  Status:
    Admin: up  Operational: down for 00:00:09 (since Nov  5 20:10:26.158)
  Candidate-paths:
    Preference: 100 (configuration)
      Name: pol_node1_node4_te
      Requested BSID: dynamic
      PCC info:
        Symbolic name: cfg_pol_node1_node4_te_discr_100
        PLSP-ID: 1
        Protection Type: unprotected-preferred
        Maximum SID Depth: 13
      Dynamic (pce cafe:0:2::2) (valid)
        Metric Type: NONE,   Path Accumulated Metric: 0
      SRv6 Information:
        Locator: Node1
        Binding SID requested: Dynamic
        Binding SID behavior: End.B6.encaps.Red
  Attributes:
    Forward Class: 0
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
```

Named Interface Link Admin Groups and SRv6-TE Affinity Maps

Named Interface Link Admin Groups and SRv6-TE Affinity Maps provide a simplified and more flexible means of configuring link attributes and path affinities to compute paths for SRv6-TE policies.

In the traditional TE scheme, links are configured with attribute-flags that are flooded with TE link-state parameters using Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF).

Named Interface Link Admin Groups and SRv6-TE Affinity Maps let you assign, or map, up to 32 color names for affinity and attribute-flag attributes instead of 32-bit hexadecimal numbers. After mappings are defined, the attributes can be referred to by the corresponding color name in the CLI. Furthermore, you can define constraints using *include-any*, *include-all*, and *exclude-any* arguments, where each statement can contain up to 10 colors.

Note You can configure affinity constraints using attribute flags or the Flexible Name Based Policy Constraints scheme; however, when configurations for both schemes exist, only the configuration pertaining to the new scheme is applied.

Configure Named Interface Link Admin Groups and SRv6-TE Affinity Maps

Configure affinity maps on the following routers:

- Routers with interfaces that have an associated admin group attribute.
- Routers that act as SRv6-TE head-ends for SR policies that include affinity constraints.

Perform the following to configure affinity maps :

- Assign affinity to interfaces

  Define affinity maps

Configuration Example for Link Admin Group

The following example shows how to assign affinity to interfaces and to define affinity maps. This configuration is applicable to any router (SRv6-TE head-end or transit node) with colored interfaces.

To assign affinity to interfaces, use the **segment-routing traffic-eng interface** *interface* **affinity name** command .

Configure on routers with interfaces that have an associated admin group attribute.

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# interface HundredGigE0/0/0/0
Router(config-sr-if)# affinity
Router(config-sr-if-affinity)# name brown
Router(config-sr-if-affinity)# exit
Router(config-sr-if)# exit
```

To define affinity maps, use the **segment-routing traffic-eng affinity-map name bit-position** *bit-position* command. The *bit-position* range is from 0 to 255.

```
Router(config-sr-te)# affinity-map
Router(config-sr-te-affinity-map)# name brown bit-position 1
```

Running Configuration

```
segment-routing
 traffic-eng
  interface HundredGigE0/0/0/0
   affinity
    name brown
   !
  !
  affinity-map
   name brown bit-position 1
  !
end
```

Segment Protection-Type Constraint

This feature introduces the ability to control whether protected or unprotected segments are used when encoding the SID-list of an SRv6 policy candidate path. The types of segments that could be used when building a SID-list include uSIDs and adjacency SIDs (uA SID).

A prefix SID is a global segment representing a prefix that identifies a specific node. A prefix SID is programmed with a backup path computed by the IGP using TI-LFA.

An adjacency SID is a local segment representing an IGP adjacency. An adjacency SID can be programmed with or without protection. Protected adjacency SIDs are programmed with a link-protectant backup path computed by the IGP (TI-LFA) and are used if the link associated with the IGP adjacency fails.

Prefix SIDs and adjacency SIDs can be leveraged as segments in a SID-list in order to forward a packet along a path traversing specific nodes and/or over specific interfaces between nodes. The type of segment used when encoding the SID-list will determine whether failures along the path would be protected by TI-LFA. Depending on the offering, an operator may want to offer either unprotected or protected services over traffic engineered paths.

The following behaviors are available with the segment protection-type constraint:

- **protected-only** — The SID-list must be encoded using protected segments.
- **protected-preferred** — The SID-list should be encoded using protected segments if available; otherwise, the SID-list may be encoded using unprotected Adj-SIDs. This is the default behavior when no segment protection-type constraint is specified.
- **unprotected-only** — The SID-list must be encoded using unprotected Adj-SID.
- **unprotected-preferred** — The SID-list should be encoded using unprotected Adj-SID if available, otherwise SID-list may be encoded using protected segments.

Usage Guidelines and Limitations

Observe the following guidelines and limitations for the platform:

- This constraint applies to candidate-paths of manual SR policies with dynamically computed paths.
- PCEP has been augmented (vendor-specific object) to allow a PCC to indicate the segment protection-type constraint to the PCE.
- When the segment protection type constraint is protected-only or unprotected-only, the path computation must adhere to the constraint. If the constraint is not satisfied, the SRv6 policy will not come up on such candidate path.
- When the segment protection-type constraint is unprotected-only, the entire SID-list must be encoded with unprotected Adj-SIDs.
- When the segment protection-type constraint is protected-only, the entire SID-list must be encoded with protected Adj-SIDs or Prefix SIDs.

Configuring Segment Protection-Type Constraint

To configure the segment protection-type behavior, use the **constraints segments protection** command.

The following example shows how to configure the policy with a SID-list that must be encoded using protected segments:

```
Router(config-sr-te)# policy POLICY1
Router(config-sr-te-policy)# color 10 end-point ipv6 2:2::22
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# constraints
Router(config-sr-te-path-pref-const)# segments
Router(config-sr-te-path-pref-const-seg)# protection protected-only
```

The following example shows how to configure the SRv6 ODN policy that must be encoded using protected segments:

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# on-demand color 20
Router(config-sr-te-color)# constraints
Router(config-sr-te-color-const)# segments
Router(config-sr-te-color-const-seg)# protection protected-only
```

SRv6 Flexible Algorithm

SRv6 Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SRv6 locators to realize forwarding beyond link-cost-based shortest path. As a result, Flexible Algorithm provides a traffic-engineered path automatically computed by the IGP to any destination reachable by the IGP.

With SRv6 Flexible Algorithm, a PE can advertise BGP service routes (global, VPN) that include the SRv6 locator (Algo 0 or Flex Algo) of the intended transport SLA.

For information about configuring Flexible Algorithm, see Configuring SRv6 IS-IS Flexible Algorithm, on page 153.

In the example below, the SR domain has 2 network slices. Each slice is assigned a /32 uSID Locator Block.

A slice can be realized with a user-defined Flex-Algo instances (for example, Flex Algo 128 = min-delay)

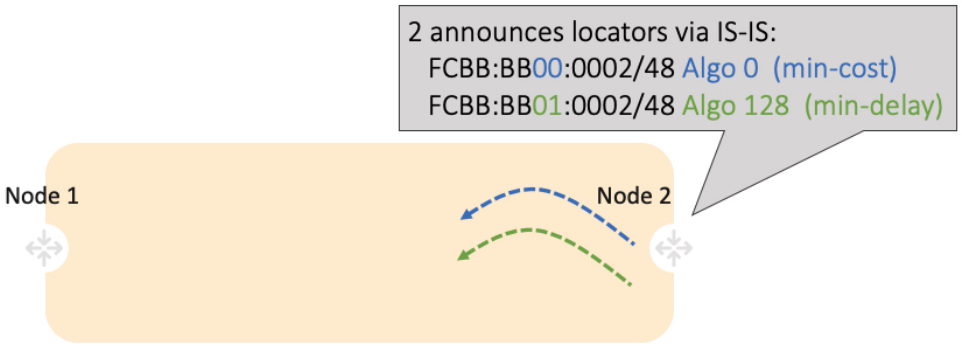
- Min-Cost Slice — FCBB:BB**00**/32
- Min-Delay Slice — FCBB:BB**01**/32

SR node2 gets a Shortest-Path Endpoint uSID (uN) from each slice:

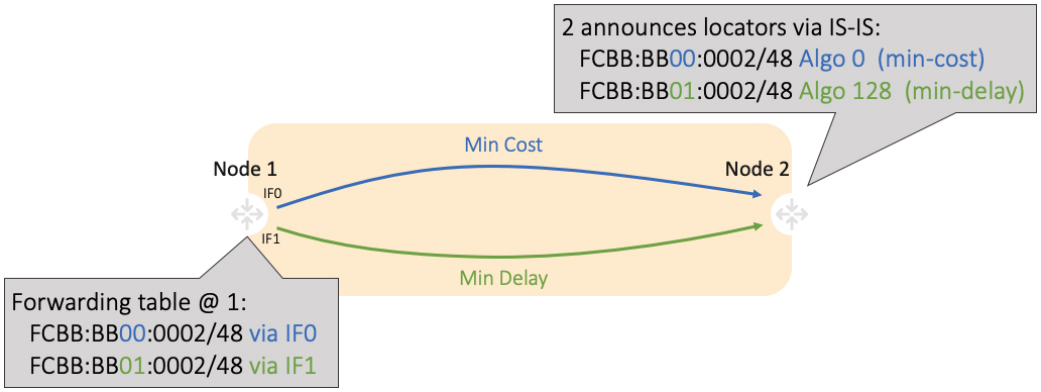
- uN **min-cost** of Node2 — FCBB:BB**00**:0002/48
- uN **min-delay** of Node2 — FCBB:BB**01**:0002/48

Node2 announces locators via IS-IS:

- FCBB:BB**00**:0002/48 Algo **0** (min-cost)
- FCBB:BB**01**:0002/48 Algo **128** (min-delay)



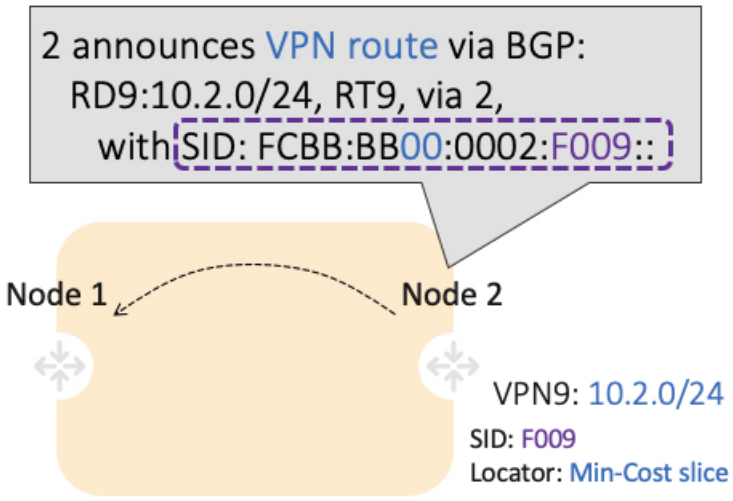
IS-IS in Node1 computes shortest paths for each locator and programs them in the FIB:



Node1 and Node2 are PEs of a common VPN. PEs advertise VPN routes via BGP with different transport SLAs. For example, traffic to a set of prefixes is to be delivered over the min-cost slice, while for another set of prefixes is to be delivered over the min-delay slice. To achieve this, the egress PE's service route advertisement includes the locator of the intended transport SLA type.

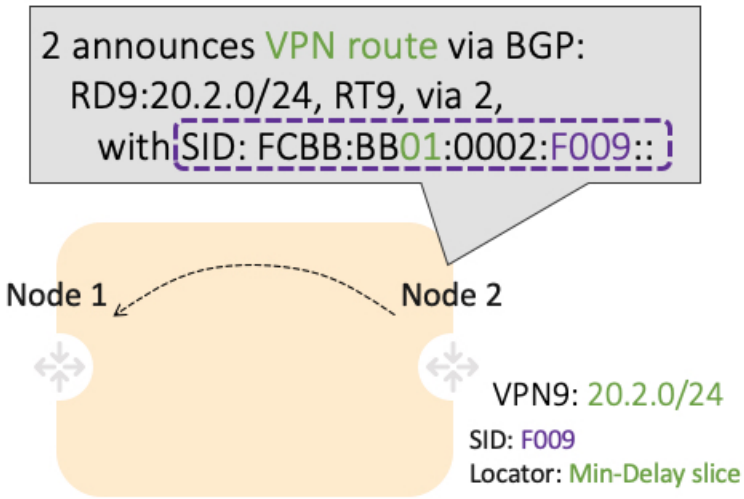
Use-case: VPN over Min-Cost Slice (Control Plane Behavior)

- Intuitive uSID program for routes advertised by Node2:
  - Within the Min-Cost Slice (FCBB:BB**00**)
  - Follow the shortest-path to Node2 (**0002**)
  - Execute VPN9 decapsulation function at Node2 (**F009**)
- Hardware Efficiency
  - Egress PE Node2 processes multiple uSIDs with a single /64 lookup
  - FCBB:BB**00**:**0002**:**F009**/64



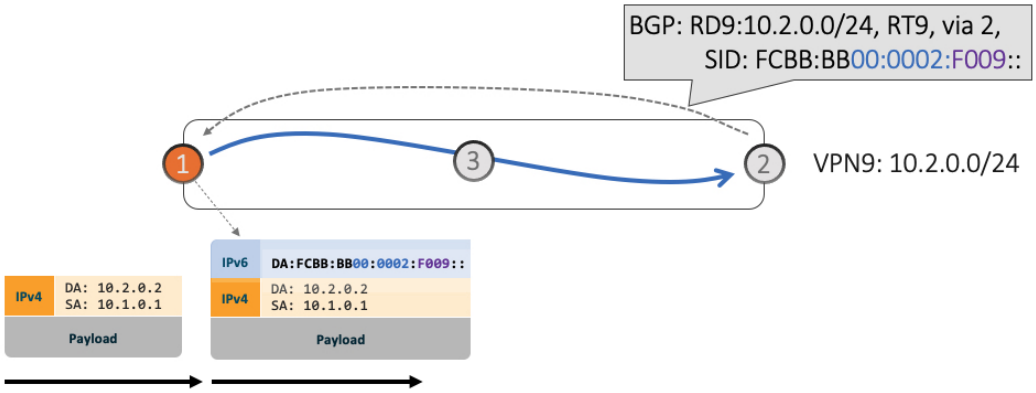
Use-case: VPN over Min-Delay Slice (Control Plane Behavior)

- Intuitive uSID program for routes advertised by Node2:
  - Within the Min-Delay Slice (FCBB:BB**01**)
  - Follow the shortest-path to Node2 (**0002**)
  - Execute VPN9 decapsulation at Node2 (**F009**)
- Hardware Efficiency
  - Egress PE 2 processes multiple uSIDs with a single /64 lookup
  - FCBB:BB**01**:**0002**:**F009**/64



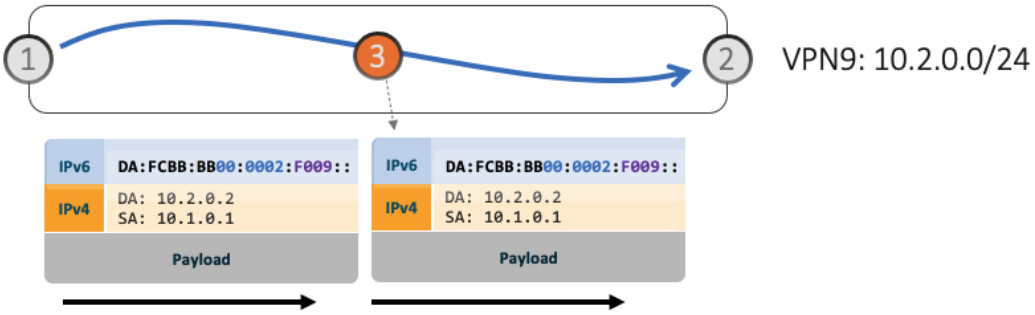
Use-case: VPN over Min-Cost Slice (Data Plane Behavior)

1. Ingress PE (Node 1) learns via BGP that prefix 10.2.0.0/24 in VPN9 is reachable via SID FCBB:BB**00**:**0002**:**F009**
2. Node 1 programs the prefix with "VPN Encaps" behavior
3. When receiving traffic with DA IP matching the prefix 10.2.0.0/24 FIB entry, Node 1 encapsulates the incoming packet into IPv6 with DA of FCBB:BB**00**:**0002**:**F009**

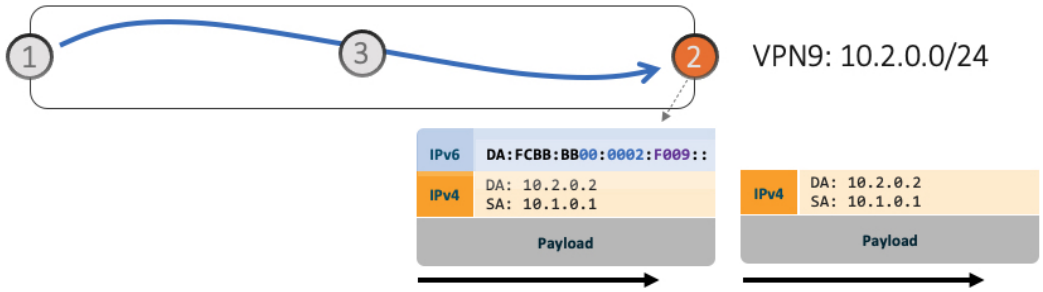


4. SRv6 allows for seamless deployment where any transit node (SRv6-capable or not) simply routes based on a /48 longest prefix match lookup.

   For example, transit node (Node 3) forwards traffic along the Algo 0 (min-cost) shortest path for the remote prefix FCBB:BB00:0002::/48.

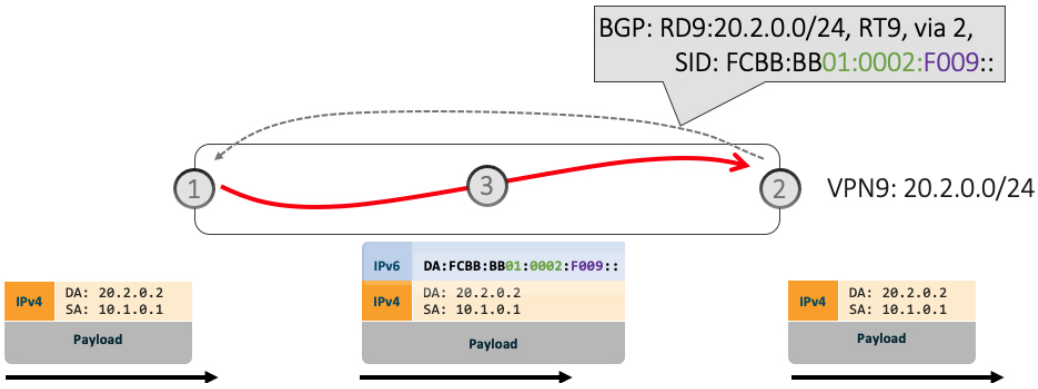


5. Egress PE (Node 2) matches local SID FCBB:BB00:0002:F009/64. Node 2 applies "VPN Decaps" behavior into VRF9 by removing the IPv6 encapsulation and looking up the payload's DA on the corresponding VPN table.



Use-case: VPN over Min-Delay Slice (Data Plane Behavior)

1. Ingress PE (Node 1) learns via BGP that prefix 20.2.0.0/24 in VPN9 is reachable via SID FCBB:BB**01**:**0002**:**F009**
2. Node 1 programs the prefix with "VPN Encaps" behavior
3. When receiving traffic with DA IP matching the prefix 20.2.0.0/24 FIB entry, Node 1 encapsulates the incoming packet into IPv6 with DA of FCBB:BB**01**:**0002**:**F009**
4. Transit node (Node 3) forwards traffic along the Algo 128 (min-delay) shortest path for the remote prefix FCBB:BB01:0002::/48.
5. Egress PE (Node 2) matches local SID FCBB:BB01:0002:F009/64. Node 2 applies "VPN Decaps" behavior into VRF9 by removing the IPv6 encapsulation and looking up the payload's DA on the corresponding VPN table.

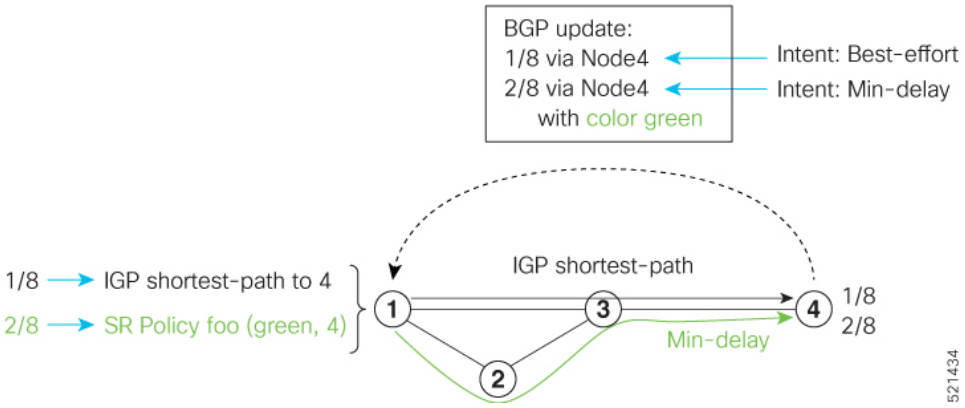


Automated Steering

Automated steering (AS) allows service traffic to be automatically steered onto the required transport SLA path programmed by an SRv6 policy.

With AS, BGP automatically steers traffic onto an SRv6 policy based on the next-hop and color of a BGP service route. The color of a BGP service route is specified by a color extended community attribute. This color is used as a transport SLA indicator, such as min-delay or min-cost.

When the next-hop and color of a BGP service route matches the end-point and color of an SRv6 policy, BGP automatically installs the route resolving onto the BSID of the matching SRv6 policy. Recall that an SRv6 policy on a head-end is uniquely identified by an end-point and color.



When a BGP route has multiple extended-color communities, each with a valid SRv6 policy, the BGP process installs the route on the SRv6 policy giving preference to the color with the highest numerical value.

The granularity of AS behaviors can be applied at multiple levels, for example:

- At a service level—When traffic destined to all prefixes in a given service is associated to the same transport path type. All prefixes share the same color.
- At a destination/prefix level—When traffic destined to a prefix in a given service is associated to a specific transport path type. Each prefix could be assigned a different color.
- At a flow level—When flows destined to the same prefix are associated with different transport path types.

Protocols

A segment routing path can be derived from various mechanisms. This section specifies extensions to the Path Computation Element Communication Protocol (PCEP) that allow a stateful PCE to compute Traffic Engineering (TE) paths as well as a PCC to request a path subject to certain constraints and optimization criteria in SR networks.

Path Computation Element Protocol

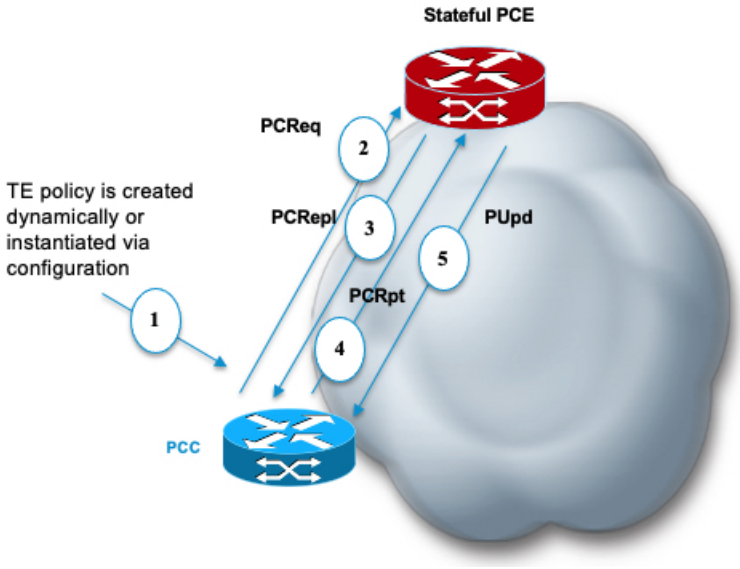
The path computation element protocol (PCEP) describes a set of procedures by which a path computation client (PCC) can report and delegate control of head-end label switched paths (LSPs) sourced from the PCC to a PCE peer. The PCE can request the PCC to update and modify parameters of LSPs it controls.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

A PCEP channel is established over TCP and has its own light-weight Keep-Alive (KA) mechanism

Upon configuring a PCE peer, a PCC opens a PCEP session to the PCE, and the PCE accepts the PCEP sessions if the following conditions are satisfied:

- The total number of PCEP sessions does not exceed the limit on the total number of PCEP sessions on the PCE.
- The KA interval indicated by PCC is acceptable to the PCE.

Sample Workflow with Stateful PCEP



1. The PCC is configured to instantiate an SRv6-TE policy.
2. The PCC sends a PCEP Path Computation Request (PCReq) to the PCE, requesting a path by specifying path attributes, optimization objectives, and constraints.
3. The PCE stores the request, computes a TE metric shortest-path, and returns the computed SID list in a PCEP Path Computation Reply (PCRepl).
4. The PCC allocates a BSID and activates the SR Policy using the SID list computed by the PCE. The PCC sends a Path Computation Report (PCRpt) to the PCE, delegating the SR Policy to the PCE and including BSID.
5. The PCE updates the paths when required (for example, following a multi-domain topology change that impacts connectivity).

Configure the Head-End Router as PCEP PCC

Configure the head-end router as PCEP Path Computation Client (PCC) to establish a connection to the PCE. The PCC and PCE addresses must be routable so that TCP connection (to exchange PCEP messages) can be established between PCC and PCE.

Perform the following to configure the Head-End Router as PCEP PCC:

- Configure the PCC to establish a Connection to the PCE
- Enable SR-TE related syslogs.
- Set the Maximum SID Depth (MSD) signaled during PCEP session establishment to 5.

- Enable PCEP reporting for all policies in the node.

Configuration Example

The following example shows how to configure an SR-TE head-end router with the following functionality:

```
/* Enable the SR-TE head-end router as a PCEP client (PCC) with 2 PCEP servers (PCE) with
different precedence values.*/
Node1(config-sr)# traffic-eng
Node1(config-sr-te)# pcc
Node1(config-sr-te-pcc)# source-address ipv6 cafe:0:1::1
Node1(config-sr-te-pcc)# pce address ipv6 cafe:0:2::2
Node1(config-pcc-pce)# precedence 10
Node1(config-pcc-pce)# exit
Node1(config-sr-te-pcc)# pce address ipv6 cafe:0:3::3
Node1(config-pcc-pce)# precedence 20
Node1(config-pcc-pce)# exit

/* Enable PCEP reporting for all policies in the node.*/
Node1(config-sr-te-pcc)# report-all
Node1(config-sr-te-pcc)# exit

/* Set the maximum SID Depth (MSD) */
Node1(config-sr-te)# srv6
Node1(config-sr-te-srv6)# maximum-sid-depth 5
Node1(config-sr-te-srv6)# exit

/* Enable SR-TE related syslogs */
Node1(config-sr-te)# logging
Node1(config-sr-te-log)# policy status
Node1(config-sr-te-log)# exit
Node1(config-sr-te)#
```

Show Running Configuration

```
segment-routing
 traffic-eng
  srv6
   maximum-sid-depth 5
  !
  logging
   policy status
  !
  pcc
   source-address ipv6 cafe:0:1::1
   pce address ipv6 cafe:0:2::2
    precedence 10
   !
   pce address ipv6 cafe:0:3::3
    precedence 20
   !
   report-all
  !
 !
!
```

Verification

```
Node1# show segment-routing traffic-eng pcc ipv6 peer brief

             Address   Precedence        State    Learned From
-------------------- ------------ ------------ ---------------
         cafe:0:2::2           10           up          config
         cafe:0:3::3           20           up          config


Node1# show segment-routing traffic-eng pcc ipv6 peer detail

PCC's peer database:
--------------------

Peer address: cafe:0:2::2
  Precedence: 10, (best PCE)
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation
  PCEP has been up for: 01:22:23
  Local keepalive timer is 30 seconds
  Remote keepalive timer is 30 seconds
  Local dead timer is 120 seconds
  Remote dead timer is 120 seconds
  Authentication: None
  Statistics:
    Open messages:      rx 1        |  tx 1
    Close messages:     rx 0        |  tx 0
    Keepalive messages: rx 164      |  tx 163
    Error messages:     rx 0        |  tx 0
    Report messages:    rx 0        |  tx 110
    Update messages:    rx 36       |  tx 0

Peer address: cafe:0:3::3
  Precedence: 20
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation
  PCEP has been up for: 01:21:48
  Local keepalive timer is 30 seconds
  Remote keepalive timer is 30 seconds
  Local dead timer is 120 seconds
  Remote dead timer is 120 seconds
  Authentication: None
  Statistics:
    Open messages:      rx 1        |  tx 1
    Close messages:     rx 0        |  tx 0
    Keepalive messages: rx 164      |  tx 162
    Error messages:     rx 0        |  tx 0
    Report messages:    rx 0        |  tx 82
    Update messages:    rx 0        |  tx 0
```

You can customize the Maximum SID Depth (MSD) signaled by PCC during PCEP session establishment.

For cases with path computation at PCE, a PCC can signal its MSD to the PCE in the following ways:

- During PCEP session establishment – The signaled MSD is treated as a node-wide property.

  MSD is configured under **segment-routing traffic-eng maximum-sid-depth** command.

  The MSD is expressed as a number uSIDs. The number of uSID is expressed as a number of carriers and the number of uSID per carrier.

- During PCEP LSP path request – The signaled MSD is treated as an LSP property.

- Local SR Policy: MSD is configured using the **segment-routing traffic-eng policy** command.

Note If the configured MSD values are different, the per-LSP MSD takes precedence over the per-node MSD.

After path computation, the resulting uSID stack size is verified against the MSD requirement.

- If the uSID stack size is larger than the MSD and path computation is performed by PCE, then the PCE returns a "no path" response to the PCC.
- If the uSID stack size is larger than the MSD and path computation is performed by PCC, then the PCC will not install the path.

Note A sub-optimal path (if one exists) that satisfies the MSD constraint could be computed in the following cases:

- For a dynamic path with TE metric, when the PCE is configured with the **pce segment-routing te-latency** command or the PCC is configured with the **segment-routing traffic-eng te-latency** command.
- For a dynamic path with LATENCY metric
- For a dynamic path with affinity constraints

For example, if the PCC MSD is 4 and the optimal path (with an accumulated metric of 100) requires 5 uSIDs, but a sub-optimal path exists (with accumulated metric of 110) requiring 4 uSIDs, then the sub-optimal path is installed.

Customize the SR-TE Path Calculation

To enable ECMP-aware path computation for TE metric, use the **te-latency** command.

Note ECMP-aware path computation is enabled by default for IGP and LATENCY metrics.

```
Router(config-sr-te)# te-latency
```

Configure PCEP Authentication

With PCEP authentication, you can establish a secure PCEP session with a PCE with one of the following methods:

- TCP Message Digest 5 (MD5) authentication: TCP Message Digest 5 (MD5) authentication is used for authenticating PCEP (TCP) sessions by using a clear text or encrypted password. This feature introduces support for TCP Authentication Option (TCP-AO), which replaces the TCP MD5 option.

  Any TCP segment coming from the PCC that does not contain a MAC matching the configured password will be rejected. Specify if the password is encrypted or clear text.

  ```
  Router(config-sr-te-pcc)# pce address ipv6 ipv6-PCE-address[password {clear | encrypted}
   LINE]
  ```

- TCP-AO: TCP-AO uses Message Authentication Codes (MACs), which provides the following:
  - Protection against replays for long-lived TCP connections
  - More details on the security association with TCP connections than TCP MD5
  - A larger set of MACs with minimal system and operational changes

  TCP-AO is compatible with Master Key Tuple (MKT) configuration. TCP-AO also protects connections when using the same MKT across repeated instances of a connection. TCP-AO protects the connections by using traffic key that are derived from the MKT, and then coordinates changes between the endpoints.

  Any TCP segment coming from the PCC that does not contain a MAC matching the configured key chain will be rejected. Use the **include-tcp-options** keyword to include other TCP options in the header for MAC calculation.

```
Router(config-sr-te-pcc)# pce address ipv6 ipv6-PCE-address tcp-ao key-chain
[include-tcp-options]
```

Note TCP-AO and TCP MD5 are never permitted to be used simultaneously. TCP-AO supports IPv6, and is fully compatible with the proposed requirements for the replacement of TCP MD5.

Configure PCEP-Related Timers

To better understand how the PCE-initiated SR policy timers operate, consider the following example:

- PCE A instantiates SR policy P at head-end N.
- Head-end N delegates SR policy P to PCE A and programs it in forwarding.
- If head-end N detects that PCE A is no longer reachable, then head-end N starts the PCE-initiated **orphan** and **state** timers for SR policy P.
- If PCE A reconnects before the **orphan** timer expires, then SR policy P is automatically delegated back to its original PCE (PCE A).
- After the **orphan** timer expires, SR policy P will be eligible for delegation to any other surviving PCE(s).
- If SR policy P is not delegated to another PCE before the **state** timer expires, then head-end N will remove SR policy P from its forwarding.

You can use the following PCEP-Related Timers:

- To specify how often keepalive messages are sent from PCC to its peers, use the **timers keepalive** command. The range is from 0 to 255 seconds; the default value is 30.

```
Router(config-sr-te-pcc)# timers keepalive seconds
```

- To specify how long the remote peers wait before bringing down the PCEP session if no PCEP messages are received from this PCC, use the **timers deadtimer** command. The range is from 1 to 255 seconds; the default value is 120.

```
Router(config-sr-te-pcc)# timers deadtimer seconds
```

- To specify how long a delegated SR policy can remain up without an active connection to a PCE, use the **timers delegation-timeout** command. The range is from 0 to 3600 seconds; the default value is 60.

```
Router(config-sr-te-pcc)# timers delegation-timeout seconds
```

- To specify the amount of time that a PCE-initiated SR policy will remain delegated to a PCE peer that is no longer reachable by the PCC, use the **timers initiated orphans** command. The range is from 10 to 180 seconds; the default value is 180.

```
Router(config-sr-te-pcc)# timers initiated orphans seconds
```

- To specify the amount of time that a PCE-initiated SR policy will remain programmed while not being delegated to any PCE, use the **timers initiated state** command . The range is from 15 to 14440 seconds (24 hours); the default value is 600.

```
Router(config-sr-te-pcc)# timers initiated state seconds
```

Note Multicast follows the same timer for Point-to-Multipoint (P2MP) policies:

- The default interval is 10 minutes.
- You can set the internal timer to '0' to ensure that forwarding states remain active indefinitely, even if the PCE is down for more than 5 minutes.
- If the PCE is down for more than 5 minutes, it is because the PCE is going down or the reachability to PCE going down.

Configure PCEP Redundancy Type

To enable PCC-centric high-availability model, use the **redundancy pcc-centric** command. The PCC-centric model changes the default PCC delegation behavior to the following:

- After LSP creation, LSP is automatically delegated to the PCE that computed it.
- If this PCE is disconnected, then the LSP is redelegated to another PCE.
- If the original PCE is reconnected, then the delegation fallback timer is started. When the timer expires, the LSP is redelegated back to the original PCE, even if it has worse preference than the current PCE.

```
Router(config-sr-te-pcc)# redundancy pcc-centric
```

SR-TE Application Programming Interface (API)

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
SR-TE Application Programming Interface (API)	Release 7.11.1	This feature introduces an API solution that simplifies the task of building SR-TE controllers and managing SRTE policies. It does so by defining gRPC API services that allow applications to request SR policy operations. The solution leverages the gRPC Service API and GPB Data models, providing a unified, scalable, and secure method for network programming. This feature introduces these changes: **New CLI** • grpc segment-routing traffic-eng policy-service **YANG Data Models:** EMSD Yang model is updated to have this config under "segment-routing" container. • Native model: `Cisco-IOS-XR-man-ems-cfg.yang` • UM model: `Cisco-IOS-XR-um-grpc-cfg.yang` (see GitHub, YANG Data Models Navigator)

SDN controllers and applications with SR traffic-engineering capabilities require a mechanism to create, monitor, and control SRv6-TE (IPv6) policies with different properties, such as optimization objectives, constraints, and segment-lists.

This feature introduces an API solution that simplifies building SR-TE controllers by defining gRPC API services to request SR policy operations. These services allow for SR policy operations, potentially including the creation, modification, or deletion of such policies. It's a more efficient and modernized approach to managing and controlling SR-TE policies.

Google-defined remote procedure call (gRPC) is an open-source RPC framework. It is based on Protocol Buffers (Protobuf), which is an open-source binary serialization protocol. gRPC provides a flexible, efficient,

automated mechanism for serializing structured data, like XML, but is smaller and simpler to use. You can define the structure using protocol buffer message types in `.proto` files. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs.

The client applications use this protocol to request information from the router and make configuration changes to the router. The process for using data models involves:

- Obtaining the data models.
- Establishing a connection between the router and the client using gRPC communication protocol.
- Managing the configuration of the router from the client using data models.

Note Refer to Use gRPC Protocol to Define Network Operations with Data Models in the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Usage Guidelines and Limitations

- The API is supported for SRv6-TE policies.
- The API is not supported on the SR PCE.

SR-TE gRPC API

The SR-TE gRPC API uses the protocol buffers definition interface language (IDL) to manage the lifecycle (creation, modification, deletion) of SR-TE policies signaled by a controller/PCE and programmed at a headend.

The gRPC requests are encoded and sent to the SR-TE headend router (gRPC server) using JSON. The router can invoke the RPC calls defined in the IDL to program the SR-TE policies.

The gRPC service specifications for SR-TE, including the definitions of messages and the data model, are housed in a proto file. This SR-TE gRPC API proto file and a sample client are available in the following Github repository: https://github.com/ios-xr/SR-TE

SR-TE gRPC API proto file provides the following RPCs:

gRPC Operation	Description
SRTEPolicyAdd	Create and modify an SR-TE policy and its identifiers and attributes (for ex: color, endpoint, head-end, candidate paths).
SRTEPolicyDelete	Delete an SR-TE policy or any of its candidate paths.

Enabling the SR-TE gRPC API on the Headend Router

Use the following commands to enable the SR-TE gRPC API:

```
RP/0/RP0/CPU0:ios(config)# grpc
RP/0/RP0/CPU0:ios(config-grpc)# segment-routing
RP/0/RP0/CPU0:ios(config-grpc-sr)# traffic-eng
RP/0/RP0/CPU0:ios(config-grpc-sr-te)# policy-service
RP/0/RP0/CPU0:ios(config-grpc-sr-te)# commit
```

Note Refer to Use gRPC Protocol to Define Network Operations with Data Models in the *Programmability Configuration Guide for Cisco 8000 Series Routers* for other gRPC parameters.

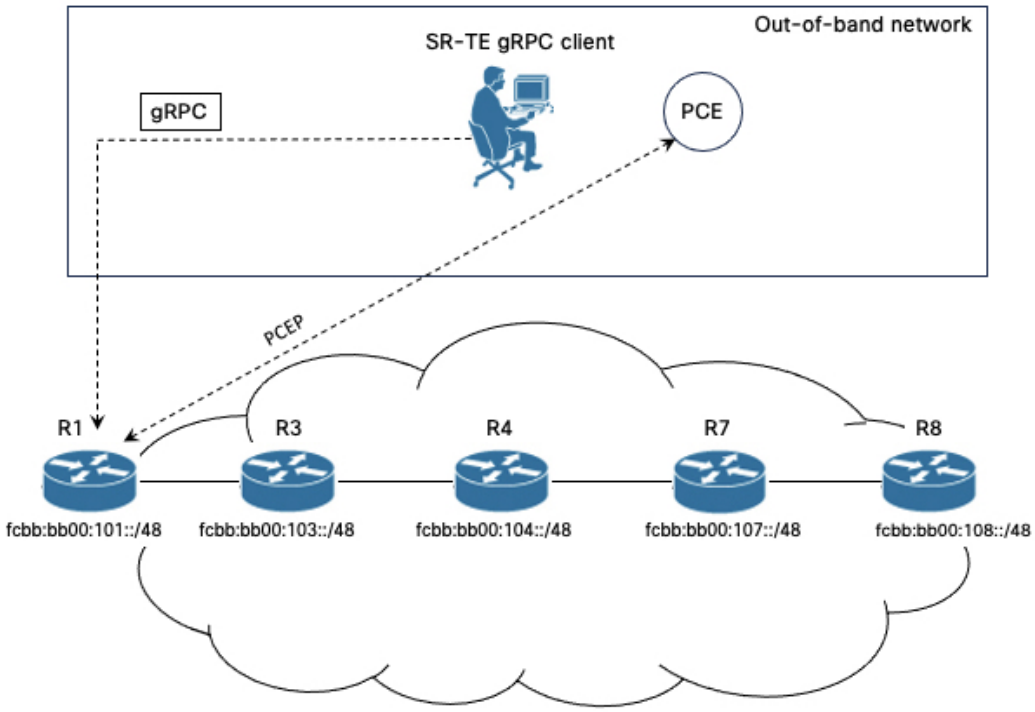
Verify

Use the show segment-routing traffic-eng service-api clients command to display the SR-TE gRPC API client and its status:

```
RP/0/RSP0/CPU0:ios# show segment-routing traffic-eng service-api clients
. . .

Client name: emsd (Protocol type: gRPC)
  Role: Active
  ID: 960268627
  State: Up
  Throttled: No
  Statistics:
    Client ever connected: Yes
    Connected: 00:00:12 (since Wed Nov 15 15:02:14 PST 2023)
    Number of add requests: 0
    Number of delete requests: 0
. . .
```

Examples

In the following examples, R1 is the SR-TE headend and the gRPC server. The headend router implements the server-side of the gRPC communication, handling requests from gRPC clients.



In these examples, an SR-TE gRPC client written in Python is used to emulate the controller node. These clients send commands to create, modify, or delete SRTE policies, and the headend router executes these commands and returns the responses.

Example 1: Programming an SRv6-TE Policy with Explicit Path via gRPC API

1. Configure the SRv6-TE policy.

   In this example, the SRv6-TE policy uses explicit paths with a segment list. The following JSON file is passed as an input to the SR-TE gRPC client to program an SRv6-TE policy with an explicit path at the head-end router.

```
controller:grpc$ more sample_srv6_explicit_path_req.json
{
 "policies": [
  {
   "key": {
    "color": 6,
    "endpoint": "2001:0:108::1",
    "headend": "2001:0:101::1"
   },
   "CPs": [
    {
     "explicit": [
      {
       "segmentList": {
        "name": "test-srv6",
        "segments": {
         "typeB": [
          "fcbb:bb00:104::",
          "fcbb:bb00:108::"
         ]
        }
       }
      }
     ],
     "preference": 10,
     "dataplane": 1,
     "key": {
      "originatorID": {
       "ASN": 64000,
       "nodeID": "1.1.1.101"
      },
      "discriminator": 100,
      "originatorProtocol": 40
     }
    }
   ],
   "bindingSIDAllocation": 1,
   "srv6BindingSID": {
    "locatorName": "LOC_BEST_EFFORT",
    "behavior": 71
   }
  }
 ]
}
```

   Execute the command/JSON file on the SR-TE gRPC client:

```
controller-grpc$ srte_client -u <user> -p <password> --policy-add
-j sample_srv6_explicit_path_req.json -a <sr-te-headend-ip-addr>:57400
Response: ['{"ReturnCode": 0, "Key": {"Color": 6, "Headend": "2001:0:101::1",
"Endpoint": "2001:0:108::1"}}']
```

Note Observe the console message indicating the policy state is “UP”:

```
RP/0/RP0/CPU0:Nov 15 15:11:23.784 PST: xtc_agent[1304]: %OS-XTC-5-SR_POLICY_UPDOWN :
SR policy 'srte_c_6_ep_2001:0:108::1' (color 6, end-point 2001:0:108::1) state changed to
UP
```

2. Verify the configuration.

   Use the following **show** commands to verify the configuration.

```
RP/0/RP0/CPU0:R1# show segment-routing traffic-eng policy color 6

SR-TE policy database
---------------------

Color: 6 End-point: 2001:0:108::1
  Name: srte_c_6_ep_2001:0:108::1
  Status:
    Admin: up  Operational: up for 00:01:00 (since Nov 15 15:11:23.784)
  Candidate-paths:
    Preference: 10 (gRPC) (active)
      Requested BSID: dynamic
      PCC info:
        Symbolic name: gRPC_srte_c_6_ep_2001:0:108::1_c_6_ep_2001:0:108::1_discr_100
        PLSP-ID: 1
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 7
      Explicit: segment-list grpc_sl_1 (valid)
        Weight: 1, Metric Type: TE
          SID[0]: fcbb:bb00:104::/48
                  Format: f3216
                  LBL:32 LNL:16 FL:0 AL:80
          SID[1]: fcbb:bb00:108::/48
                  Format: f3216
                  LBL:32 LNL:16 FL:0 AL:80
      SRv6 Information:
        Locator: LOC_BEST_EFFORT
        Binding SID requested: Dynamic
        Binding SID behavior: uB6 (Insert.Red)
  Attributes:
    Binding SID: fcbb:bb00:101:e005::
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0


RP/0/RP0/CPU0:R1# show cef ipv6 fcbb:bb00:101:e005::/64

fcbb:bb00:101:e005::/64, version 927, SRv6 Endpoint uB6 (Insert.Red), internal 0x1000001
 0x200 (ptr 0x8b1495e8) [1], 0x400 (0x8a460958), 0x0 (0xa0e205e8)
 Updated Nov 15 15:11:23.786
 local adjacency to TenGigE0/0/0/0

 Prefix Len 64, traffic index 0, precedence n/a, priority 0
  gateway array (0x8a2f3708) reference count 1, flags 0x400000, source rib (7), 0 backups

                [2 type 3 flags 0x8401 (0x8a396cf8) ext 0x0 (0x0)]
```

```
 LW-LDI[type=3, refc=1, ptr=0x8a460958, sh-ldi=0x8a396cf8]
 gateway array update type-time 1 Nov 15 15:11:23.786
LDI Update time Nov 15 15:11:23.787
LW-LDI-TS Nov 15 15:11:23.787
Accounting: Disabled
  via fe80::28a:96ff:feaa:ac00/128, TenGigE0/0/0/0, 8 dependencies, weight 1, class 0,
protected, ECMP-backup (Local-LFA) [flags 0x600]
   path-idx 0 bkup-idx 1 NHID 0x0 [0xa0ffa0a0 0x0]
   next hop fe80::28a:96ff:feaa:ac00/128
   SRv6 H.Insert.Red SID-list {fcbb:bb00:104::}
  via fe80::d66a:35ff:fef3:2000/128, TenGigE0/0/0/1, 8 dependencies, weight 1, class
0, protected, ECMP-backup (Local-LFA) [flags 0x600]
   path-idx 1 bkup-idx 0 NHID 0x0 [0xa0ffa190 0x0]
   next hop fe80::d66a:35ff:fef3:2000/128
   SRv6 H.Insert.Red SID-list {fcbb:bb00:104::}

   Weight distribution:
   slot 0, weight 1, normalized_weight 1, class 0
   slot 1, weight 1, normalized_weight 1, class 0
   Load distribution: 0 1 (refcount 2)

   Hash  OK  Interface                 Address
   0     Y   TenGigE0/0/0/0            fe80::28a:96ff:feaa:ac00
   1     Y   TenGigE0/0/0/1            fe80::d66a:35ff:fef3:2000
```

Example 2: Programming an SRv6-TE Policy with Dynamic Path (Delegated to PCE) via gRPC API

1. Configure the SRv6-TE policy.

   In this example, the SRv6-TE policy uses dynamic paths delegated to the PCE.

   The following JSON file is passed as an input to the SR-TE gRPC client to program an SRv6-TE policy with a dynamic path.

```
controller:grpc$ more sample_srv6_dynamic_path_req.json
{
 "policies": [
  {
   "key": {
    "color": 7,
    "endpoint": "2001:0:108::1",
    "headend": "2001:0:101::1"
   },
   "CPs": [
    {
     "dynamic": {
      "delegate": true,
      "ometric": 0
     },
     "preference": 10,
     "dataplane": 1,
     "key": {
      "originatorID": {
       "ASN": 64000,
       "nodeID": "1.1.1.101"
      },
      "discriminator": 100,
      "originatorProtocol": 40
     }
    }
   ],
   "bindingSIDAllocation": 1,
   "srv6BindingSID": {
```

```
    "locatorName": "LOC_BEST_EFFORT",
    "behavior": 71
   }
  }
 ]
}
```

Execute the command/JSON file on the SR-TE gRPC client:

```
controller-grpc$ srte_client -u <user> -p <password> --policy-add
-j sample_srv6_dynamic_path_req.json -a <sr-te-headend-ip-addr>:57400
Response: ['{"ReturnCode": 0, "Key": {"Color": 7, "Headend": "2001:0:101::1",
"Endpoint": "2001:0:108::1"}}']
```

Note Observe the console message indicating the policy state is “UP”:

```
RP/0/RP0/CPU0:Nov 15 15:25:23.671 PST: xtc_agent[1304]: %OS-XTC-5-SR_POLICY_UPDOWN :
SR policy 'srte_c_7_ep_2001:0:108::1' (color 7, end-point 2001:0:108::1) state changed to
UP
```

2. Verify the configuration.

   Use the following **show** commands to verify the configuration.

```
RP/0/RP0/CPU0:R1# show segment-routing traffic-eng pol color 7

SR-TE policy database
---------------------

Color: 7, End-point: 2001:0:108::1
  Name: srte_c_7_ep_2001:0:108::1
  Status:
    Admin: up  Operational: up for 00:01:06 (since Nov 15 15:25:23.671)
  Candidate-paths:
    Preference: 10 (gRPC) (active)
      Requested BSID: dynamic
      PCC info:
        Symbolic name: gRPC_srte_c_7_ep_2001:0:108::1_c_7_ep_2001:0:108::1_discr_100
        PLSP-ID: 3
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 7
      Dynamic (pce 2001:0:109::1) (valid)
        Metric Type: TE,   Path Accumulated Metric: 44
          SID[0]: fcbb:bb00:103::/48 Behavior: uN (PSP/USD) (48)
                  Format: f3216
                  LBL:32 LNL:16 FL:0 AL:80
                  Address: 2001:0:103::1
          SID[1]: fcbb:bb00:104::/48 Behavior: uN (PSP/USD) (48)
                  Format: f3216
                  LBL:32 LNL:16 FL:0 AL:80
                  Address: 2001:0:104::1
          SID[2]: fcbb:bb00:107::/48 Behavior: uN (PSP/USD) (48)
                  Format: f3216
                  LBL:32 LNL:16 FL:0 AL:80
                  Address: 2001:0:107::1
          SID[3]: fcbb:bb00:108::/48 Behavior: uN (PSP/USD) (48)
                  Format: f3216
                  LBL:32 LNL:16 FL:0 AL:80
                  Address: 2001:0:108::1
      SRv6 Information:
        Locator: LOC_BEST_EFFORT
```

```
        Binding SID requested: Dynamic
        Binding SID behavior: uB6 (Insert.Red)
  Attributes:
    Binding SID: fcbb:bb00:101:e006::
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0


RP/0/RP0/CPU0:R1# show cef ipv6 fcbb:bb00:101:e006::/64

fcbb:bb00:101:e006::/64, version 936, SRv6 Endpoint uB6 (Insert.Red), internal 0x1000001
 0x200 (ptr 0x8b149100) [1], 0x400 (0x8a460918), 0x0 (0xa0e20598)
 Updated Nov 15 15:25:23.672
 local adjacency to TenGigE0/0/0/1

 Prefix Len 64, traffic index 0, precedence n/a, priority 0
  gateway array (0x8a2f3618) reference count 1, flags 0x500000, source rib (7), 0 backups

                [2 type 3 flags 0x8401 (0x8a396e58) ext 0x0 (0x0)]
  LW-LDI[type=3, refc=1, ptr=0x8a460918, sh-ldi=0x8a396e58]
  gateway array update type-time 1 Nov 15 15:25:23.672
 LDI Update time Nov 15 15:25:23.672
 LW-LDI-TS Nov 15 15:25:23.672
 Accounting: Disabled
   via fe80::28a:96ff:feaa:ac00/128, TenGigE0/0/0/0, 10 dependencies, weight 1, class
0, backup (Local-LFA) [flags 0x300]
    path-idx 0 NHID 0x0 [0x8a0c5a00 0x0]
    next hop fe80::28a:96ff:feaa:ac00/128
    local adjacency
    SRv6 H.Insert.Red SID-list {fcbb:bb00:103:104:107::}
   via fe80::d66a:35ff:fef3:2000/128, TenGigE0/0/0/1, 10 dependencies, weight 1, class
0, protected [flags 0x400]
    path-idx 1 bkup-idx 0 NHID 0x0 [0xa0ffa190 0x0]
    next hop fe80::d66a:35ff:fef3:2000/128
    SRv6 H.Insert.Red SID-list {fcbb:bb00:103:104:107::}

    Weight distribution:
    slot 0, weight 1, normalized_weight 1, class 0
    Load distribution: 0 (refcount 2)

    Hash  OK  Interface                Address
    0     Y   TenGigE0/0/0/1           fe80::d66a:35ff:fef3:2000


RP/0/RP0/CPU0:R1# show segment-routing traffic-eng forwarding pol color 7

SR-TE Policy Forwarding database
--------------------------------

Color: 7, End-point: 2001:0:108::1
  Name: srte_c_7_ep_2001:0:108::1
  Binding SID: fcbb:bb00:101:e006::
  Active LSP:
    Candidate path:
      Preference: 10 (gRPC)
    Segment lists:
      SL[0]:
        Name: dynamic
        SL ID: 0xa000002
        Switched Packets/Bytes: ?/?
        Paths:
```

```
          Path[0]:
            Outgoing Interfaces: TenGigE0/0/0/0
            Next Hop: fe80::28a:96ff:feaa:ac00
            FRR Pure Backup: Yes
            ECMP/LFA Backup: Yes
            SID stack (Top -> Bottom): {fcbb:bb00:103::/48, fcbb:bb00:104::/48,
fcbb:bb00:107::/48}
          Path[1]:
            Outgoing Interfaces: TenGigE0/0/0/1
            Next Hop: fe80::d66a:35ff:fef3:2000
            FRR Pure Backup: No
            ECMP/LFA Backup: No
            SID stack (Top -> Bottom): {fcbb:bb00:103::/48, fcbb:bb00:104::/48,
fcbb:bb00:107::/48}

  Policy Packets/Bytes Switched: ?/?
```

Example 3: Delete the Policy

1. On the cRPC client, run the following command to delete the policy configuration:

```
controller-grpc$ srte_client -u <user> -p <password> --policy-delete
-j sample_srv6_explicit_path_req.json -a <sr-te-headend-ip-addr>:57400
Response: ['{"ReturnCode": 0, "Key": {"Color": 6, "Headend": "2001:0:101::1",
"Endpoint": "2001:0:108::1"}}']

controller-grpc$ srte_client -u <user> -p <password> --policy-delete
-j sample_srv6_explicit_path_req.json -a <sr-te-headend-ip-addr>:57400
Response: ['{"ReturnCode": 0, "Key": {"Color": 7, "Headend": "2001:0:101::1",
"Endpoint": "2001:0:108::1"}}']
```

Note Observe the console message indicating the policy state is “DOWN”:

```
RP/0/RP0/CPU0:Nov 15 15:30:14.180 PST: xtc_agent[1304]: %OS-XTC-5-SR_POLICY_UPDOWN :
SR policy 'srte_c_6_ep_2001:0:108::1' (color 6, end-point 2001:0:108::1) state changed to
DOWN (path invalidated)

RP/0/RP0/CPU0:Nov 15 15:30:27.181 PST: xtc_agent[1304]: %OS-XTC-5-SR_POLICY_UPDOWN :
SR policy 'srte_c_7_ep_2001:0:108::1' (color 7, end-point 2001:0:108::1) state changed to
DOWN (path invalidated)
```

2. Verify the delete operation.

```
RP/0/RSP0/CPU0:ios# show segment-routing traffic-eng service-api clients

Client name: emsd (Protocol type: gRPC)
  Role: Active
  ID: 440080357
  State: Up
  Throttled: No
  Statistics:
    Client ever connected: Yes
    Connected: 00:28:00 (since Wed Nov 15 15:02:14 PST 2023)
    Number of add requests: 2
    Number of delete requests: 2
```

Reporting of SR-TE Policies Using BGP- Link State

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
Reporting of SR-TE Policies Using BGP-Link State	Release 24.1.1	BGP- Link State (LS) is a mechanism by which LS and Traffic Engineering (TE) information can be collected from networks and shared with external components (such as, Segment Routing Path Computation Element (SR-PCE) or Crossword Optimization Engine (COE)) using the BGP routing protocol. This feature gathers the Traffic Engineering Policy information that is locally available in a node and advertises it in BGP-LS for SR-MPLS and SRv6. The operators can now take informed decisions based on the information that is gathered on their network's path computation, reoptimization, service placement, network visualization, and so on. The feature introduces these changes: **CLI:** • **distribute link-state** **YANG Data Model:** • New XPaths for `module Cisco-IOS-XR-infra-xtc-agent-cfg.yang` (see GitHub, YANG Data Models Navigator)

This function is achieved using a BGP Network Layer Reachability Information (NLRI) encoding format. BGP-LS consumes structured IGP data (for example, router-id, remote-IP-address of a link, local-IP address of a link, link identifier, and so on). and creates BGP-LS (NLRI) or attributes that BGP or other components like Cisco IOS XR Traffic Controller (XTC) can consume. Current implementation of BGP-LS can report topology using Nodes, Links, and Prefixes.

Modern Segment Routing (SR) networks often use SR Traffic Engineering (SR-TE) to influence the path that each specific traffic takes over the network. SR-TE tunnels can be provisioned manually on the tunnel head, but often they are calculated and provisioned by the central controller. Often operator of the network wants the ability to force the traffic over specific nodes and links.

Now the operators have the option to collect reports of the SR-TE and Policy information that is locally available in a node and advertise it into BGP-LS updates, which can be used by external components. Refer the IEFT for examples. This feature is implemented so that the operators have control over their network's path computation, reoptimization, service placement, network visualization, and so on.

Note Circuit Style (CS) SR policies are reported but without the CS policies' specific attributes, like the bidirectional constraints, per-hop behavior, and so on.

Restrictions to Reporting of SRTE Policies using BGP-LS

Some important points to be aware related to this feature are as follows:

- This feature has high availability, ensuring BGP-LS reporting at all times.
- This feature works with PCE State Sync. The policy information learned via the state-sync channel is not advertised in BGP-LS by the PCE.
- The feature works with PCE redundancy. Both PCEs advertise the BGP-LS policy information. The consumers can select only one policy based on the BGP best path logic.

Configure Reporting of SRTE Policies using BGP-LS

The reporting of policies to BGP-LS is disabled by default. Configuring the **distribute link-state** under the SR-PCE or SR-TE configuration enables this feature. Once enabled, all the existing SR policies or Candidate Path (CPs) are encoded to BGP-LS. You can disable this feature by removing the configuration and all the SR policies or CPs are withdrawn from BGP which deletes all of the previously encoded SR policies or CPs.

Note When SR policies that are reporting in BGP-LS are enabled by the operator, the Head End only reports the Active Candidate Path (CPs) (CPs installed in the forwarding). There are monitoring use cases that require reporting of inactive CPs. The following CLI is needed to report inactive CPs.

For both SR-TE and SR-PCE, you need to use the global CLI to enable the reporting of policies or Circuit Style (CS) to BGP-LS:

Configuration Example

Configure the following command to enable reporting and syncing of SR policies in BGP-LS at the Head End:

```
Router# config
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# distribute link-state
Router(config-sr-te-distribute-ls)# report-candidate-path-inactive
Router(config-sr-te-distribute-ls)# commit
Router(config-sr-te-distribute-ls)# exit
```

Note The **report-candidate-path-inactive** command is an optional command to report inactive CPs.

Running Configuration

This is a sample running configuration which shows that you have configured BGP-LS reporting feature.

```
segment-routing
 traffic-eng
  distribute link-state
   report-candidate-path-inactive
  !
 !
!
```

Verification

Use the **show pce segment-routing traffic-eng policy private** and **show pce distributed-ls events** to verify the configuration.

```
Router#show pce segment-routing traffic-eng policy private

PCE's policy database:
----------------------
PCC Address: 192.168.2.1
  Color: 100, Endpoint: 192.0.2.1
    Name: srte_c_100_ep_192.0.2.1
    Self Pointer: 0x317d7f0
    Active C-path Pointer: 0x317d9b0
    Candidate-paths:
      Symbolic-name: cfg_foo_discr_100 (Active)
        PLSP-ID: 1
        NB-API Notification pending flag: 0
        Self Pointer: 0x317d9b0
        Tunnel Pointer: 0x317d4e0
        Policy Pointer: 0x317d7f0
        Cached distribute LS element:
          Sense: TRUE
          Refcount: 1
          Is on queue: FALSE
          Node identifiers:
            Protocol: 9 router ID: 192.0.2.1
          Policy identifiers:
            Color: 100 Endpoint: 192.0.2.1
            Flags: 0x00
          Candidate path identifiers:
            Originator: 0.0.0.0 protocol: 3 ASN: 0
            Flags: 0x00
            Discriminator: 100
          Candidate path attributes:
            Name: foo
            Policy name: srte_c_100_ep_192.0.2.1
            State:
              Priority: 0 flags: 0x5800 (active, evaluated, valid-sid-list)
              Preference: 100
            BSID:
              Flags: 0x4000 (alloc)
              BSID: 24021
              Specified BSID: 0
            Constraints:
              Bitfield: 0x0020 flags: 0x4000 (protected-only) MT-ID: 0
              Algorithm: 0
              Bandwidth: 0 kbps
              Metric constraints[0]:
                Type: 0 flags: 0x80 (optimization)
```

```
              Margin: 0 bound: 0
            Metric constraints[1]:
              Type: 4 flags: 0x10 (bound)
              Margin: 0 bound: 10
          Segment lists:
            Segment list[0]:
              Flags: 0x3800 (computed, verified, first-seg-resolved) MT-ID: 0 algorithm:
 0 weight: 0
              Metric[0]:
                Type: 0 flags: 0x10 (value)
                Margin: 0 bound: 0 value: 10
              Segments:
                Segment[0]:
                  Bitfield: 0x0000 type: 3 flags: 0x8000 (sid-present)
                  SID: 102000
                  Descriptor:
                    Algorithm: 0
                    Local address: 192.0.2.1 [0] remote address: 0.0.0.0 [0]
```

```
Router#show pce distribute-ls events

Distribute LS events:
----------------
Event history (oldest first):
  Time                    Event
  Apr 11 01:50:48.985     [UID: 0] SR CP NLRI: node ID: SR RID 192.0.2.1, ls_id 0, asn 0,
bitf 0x00000200 policy ID: endpoint: 192.168.0.2 color: 100 CP ID: originator: config 0.0.0.0
 asn: 0 discriminator: 100 (oper: add)
  Apr 11 01:50:50.046     [UID: 1] SR CP NLRI: node ID: SR RID 192.0.2.1, ls_id 0, asn 0,
bitf 0x00000200 policy ID: endpoint: 192.168.0.2 color: 100 CP ID: originator: config 0.0.0.0
 asn: 0 discriminator: 100 (oper: add)
RP/0/0/CPU0:rtrX#show pce distribute-ls summary
Tue Apr 11 02:03:36.289 PDT
Distribution enabled: yes
Connected to LS-LIB: yes
Encode queue size: 0
Estimated encoding rate: 17280/s
NLRIs encoded to LS-LIB:
  SR candidate path: 2 added, 0 removed, 0 errored, 0 replaced
Element stats:
  SR candidate path: 1 (watermark: 2)
Throttle timer:
  Running: no
```

Below is the example show output for SRv6.

```
Router# show segment-routing traffic-eng policy color 200 private

SR-TE policy database
---------------------

Color: 200, End-point: 192::2 ID: 2
  Name: srte_c_200_ep_192::2
  Status:
    Admin: up  Operational: up for 00:01:07 (since Mar  6 11:17:42.580)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Originator: ASN 0 node-address <None> discriminator: 100
      Name: bar
      Requested BSID: dynamic
      PCC info:
        Symbolic name: cfg_bar_discr_100
        PLSP-ID: 2
        Is orphan: no
```

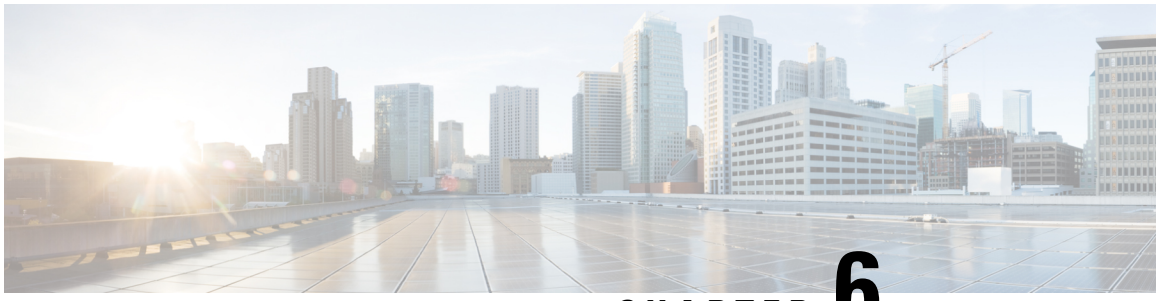
```
      State timer:
        Running: no
    Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 10
    ID: 1
    Source: 192::1
    Stale: no
    Checkpoint flags: 0x00000000
    Path Type: SRV6
    Performance-measurement:
      Reverse-path segment-list:
      Delay-measurement: Disabled
      Liveness-detection: Disabled
    Dynamic (pce 192.168.0.3) (valid)
      Metric Type: IGP,   Path Accumulated Metric: 10
      IGP area: 0
        SID[0]: fccc:ccc1:2::/48 Behavior: uN (PSP/USD) (48)
                Format: f3216
                LBL:32 LNL:16 FL:0 AL:80
                Address: 192::2
    SRv6 Information:
      Locator: loc1Algo0
      Binding SID requested: Dynamic
      Binding SID behavior: uB6 (Insert.Red)
    Cached distribute LS element:
      Sense: TRUE
      Refcount: 1
      Is on queue: FALSE
      Node identifiers:
        Protocol: 9 router ID: 192::1
      Policy identifiers:
        Color: 200 Endpoint: 192::2
        Flags: 0x80 (endpoint-v6)
      Candidate path identifiers:
        Originator: 0.0.0.0 protocol: 3 ASN: 0
        Flags: 0x00
        Discriminator: 100
      Candidate path attributes:
        Name: bar
        Policy name: srte_c_200_ep_192::2
        State:
          Priority: 0 flags: 0x5A00 (active, evaluated, valid-sid-list, delegated)
          Preference: 100
        SRv6 BSID:
          Flags: 0x8000 (alloc)
          BSID: fccc:ccc1:1:e018::
          Specified BSID: ::
          Endpoint:
            Endpoint function: 71 flags: 0x00 algorithm: 0
          Structure:
           Locator block length: 32 locator node length: 16 function length: 16 arguments
length: 0
        Constraints:
          Bitfield: 0x0020 flags: 0xC000 (dataplane-v6, protected) MT-ID: 0
          Algorithm: 0
          Bandwidth: 0 kbps
          Metric constraints[0]:
            Type: 0 flags: 0x80 (optimization)
            Margin: 0 bound: 0
          Metric constraints[1]:
            Type: 4 flags: 0x10 (bound)
            Margin: 0 bound: 10
```

```
          Segment lists:
            Segment list[0]:
              Flags: 0xB800 (dataplane-v6, computed, verified, first-seg-resolved) MT-ID:
0 algorithm: 0 weight: 1
              Metric[0]:
                Type: 0 flags: 0x10 (value)
                Margin: 0 bound: 0 value: 10
              Segments:
                Segment[0]:
                  Bitfield: 0x0003 type: 9 flags: 0x8000 (sid-present)
                  SID: fccc:ccc1:2::
                  Descriptor:
                    Algorithm: 0
                    Local address: 192::2 [0] remote address: :: [0]
                  Endpoint:
                    Endpoint function: 48 flags: 0x00 algorithm: 0
                  Structure:
                    Locator block length: 32 locator node length: 16 function length: 0
arguments length: 80
  LSPs:
    LSP[0]:
      LSP-ID: 2 policy ID: 2 (active)
      State: Programmed
      Binding SID: fccc:ccc1:1:e018::
      Install timer:
        Running: no
      Cleanup timer:
        Running: no
      Delete timer:
        Running: no
      Revert timer:
        Running: no
      SM chain:
        Init -> Egress paths
        Egress paths pending -> BSID RW
        BSID rewrite pending -> Success
      Forwarding flags: 0x00000008
      Candidate path ID: 1
      Flags:
      SL-ID:
        Sent/Received/Transferred: 1/1/0
      SLs:
        SL[0]:
          Name: dynamic
          Type: Dynamic PCE
          Checkpoint id: 1
          NH SRV6 SID: fccc:ccc1:2::
          SL ID: 0xa000001
          Flags:
          Normalized Weight: 1
          ENS: 1
          Paths:
            Path[0]:
              Interface version: 1
              Flags:
              Outgoing interface: Gi0/2/0/0
              Weight: 1
              SID stack:
              Total SID count: 0
              Underlay Normalized Weight: 1
            Path[1]:
              Interface version: 1
              Flags:
              Outgoing interface: Gi0/2/0/1
```

```
              Weight: 1
              SID stack:
              Total SID count: 0
              Underlay Normalized Weight: 1
            Path[2]:
              Interface version: 1
              Flags:
              Outgoing interface: Gi0/2/0/2
              Weight: 1
              SID stack:
              Total SID count: 0
              Underlay Normalized Weight: 1
  Attributes:
    Binding SID: fccc:ccc1:1:e018::
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
    Path Type: SRV6
  Notification to clients:
    Binding SID: fccc:ccc1:1:e018::
    Bandwidth : 0 Kbps (0 Kbps)
    State: UP
    Flags: [add] [ipv6_caps]
    Metric Type: IGP
    Metric Value: 10
    Admin Distance: 30
  ifhandle: 0x00000000
  Source: 192::1
  Transition count: 1
  LSPs created count: 1
  Reoptimizations completed count: 1
  Retry Action Flags: 0x00000000, ()
  Last Retry Timestamp: never (0 seconds ago)
  Policy reference: 0x1222c10
  Event history (oldest first):
    Time                    Event
    Mar  6 11:17:40.563     POLICY CREATE
    Mar  6 11:17:40.564     (x3) CP PCRPT: 1 hops:
    Mar  6 11:17:41.071     CP PCUPD: CP-ID: 1, path change: true, notify: true, hops:
fccc:ccc1:2::
    Mar  6 11:17:41.072     CP PCRPT: 1 hops: fccc:ccc1:2::
    Mar  6 11:17:41.072     LSP CREATE: 2, need BSID RW: true, BSID: ::
    Mar  6 11:17:41.072     CP CHANGE: PREF: 100 [PROTO: 30, ORIGIN: 0/<None>, DISC: 100]
    Mar  6 11:17:42.579     SRv6 BSID RW REQ: ID 2
    Mar  6 11:17:42.580     SRv6 BSID RW RES: ID 2 Status: success
    Mar  6 11:17:42.580     LSP PCRPT: 2, hops: fccc:ccc1:2::
    Mar  6 11:17:42.580     CP PCRPT REMOVE: 1
    Mar  6 11:17:42.580     IM STATE CHANGE: UNKNOWN to UP, count: 0


Router# show pce segment-routing traffic-eng policy private

PCE's policy database:
----------------------

PCC Address: 192.168.2.1
  Color: 200, Endpoint: 192::2
    Name: srte_c_200_ep_192::2
    Self Pointer: 0x26cd890
    Active C-path Pointer: 0x26cda00
```

```
  Candidate-paths:
    Symbolic-name: cfg_bar_discr_100 (Active)
      PLSP-ID: 2
      NB-API Notification pending flag: 0
      Self Pointer: 0x26cda00
      Tunnel Pointer: 0x26cd580
      Policy Pointer: 0x26cd890
      Cached distribute LS element:
        Sense: TRUE
        Refcount: 1
        Is on queue: FALSE
        Node identifiers:
          Protocol: 9 router ID: 192::1
        Policy identifiers:
          Color: 200 Endpoint: 192::2
          Flags: 0x80 (endpoint-v6)
        Candidate path identifiers:
          Originator: 0.0.0.0 protocol: 3 ASN: 0
          Flags: 0x00
          Discriminator: 100
        Candidate path attributes:
          Name: bar
          Policy name: srte_c_200_ep_192::2
          State:
            Priority: 0 flags: 0x5A00 (active, evaluated, valid-sid-list, delegated)
            Preference: 100
          SRv6 BSID:
            Flags: 0x8000 (alloc)
            BSID: fccc:ccc1:1:e018::
            Specified BSID: ::
            Endpoint:
              Endpoint function: 71 flags: 0x00 algorithm: 0
            Structure:
             Locator block length: 32 locator node length: 16 function length: 16 arguments
length: 0
          Constraints:
            Bitfield: 0x0020 flags: 0xC000 (dataplane-v6, protected) MT-ID: 0
            Algorithm: 0
            Bandwidth: 0 kbps
            Metric constraints[0]:
              Type: 0 flags: 0x80 (optimization)
              Margin: 0 bound: 0
            Metric constraints[1]:
              Type: 4 flags: 0x10 (bound)
              Margin: 0 bound: 10
          Segment lists:
            Segment list[0]:
             Flags: 0xB800 (dataplane-v6, computed, verified, first-seg-resolved) MT-ID:
0 algorithm: 0 weight: 0
              Metric[0]:
                Type: 0 flags: 0x10 (value)
                Margin: 0 bound: 0 value: 10
              Segments:
                Segment[0]:
                  Bitfield: 0x0003 type: 9 flags: 0x8000 (sid-present)
                  SID: fccc:ccc1:2::
                  Descriptor:
                    Algorithm: 0
                    Local address: 192::2 [0] remote address: :: [0]
                  Endpoint:
                    Endpoint function: 48 flags: 0x00 algorithm: 0
                  Structure:
```

```
                     Locator block length: 32 locator node length: 16 function length: 0
arguments length: 80


Router# show bgp link-state link-state | i \\[SP\\]

*>i[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.1][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
*>
[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.3][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792



Router# show bgp link-state link-state
[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.1][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
BGP routing table entry for
[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.1][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker               128          128
Last Modified: Mar  6 11:17:43.000 for 00:04:09
Last Delayed at: ---
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.168.2.1 (metric 20) from 192.168.2.1 (192.168.0.1)
      Origin IGP, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 128
      Link-state:
                  SRTE-CP-State: Priority: 0 Flags: 0x5a00 Preference: 100
                  SR-Policy-CP-name: bar
                  SR-Policy-name: srte_c_200_ep_192::2
                  SRTE-CP-Constraints: Flags: 0xc000 Mtid: 0 Algorithm: 0
                  SRTE-CP-Constraints-Metric: Type: 0 Flags: 0x80 Margin: 0
                  Bound: 0
                  SRTE-CP-Constraints-Metric: Type: 4 Flags: 0x10 Margin: 0
                  Bound: 10
                  SRTE-Segment-List: Flags: 0xb800 Mtid: 0 Algorithm: 0
                  Weight: 1
                 SRTE-Segment: Segment-Type: 9 Flags: 0x8000 SID: fccc:ccc1:2:: Algorithm:
 0
                   Local-node: 192::2
                  SRv6-Endpoint-Fn: 48 Flags: 0x0 Algo: 0
                  SRv6-SID-Struct: LBL: 32 LNL: 16 FL: 0 AL: 80
                  SRTE-Segment-List-Metric: Type: 0 Flags: 0x10 Margin: 0
                  Bound: 0 Value: 10
                  SRTE-CP-SRV6-BSID: Flags: 0x8000 BSID: fccc:ccc1:1:e018::
                   Specified BSID: ::
                  SRv6-Endpoint-Fn: 71 Flags: 0x0 Algo: 0
                  SRv6-SID-Struct: LBL: 32 LNL: 16 FL: 16 AL: 0
```



CHAPTER 6

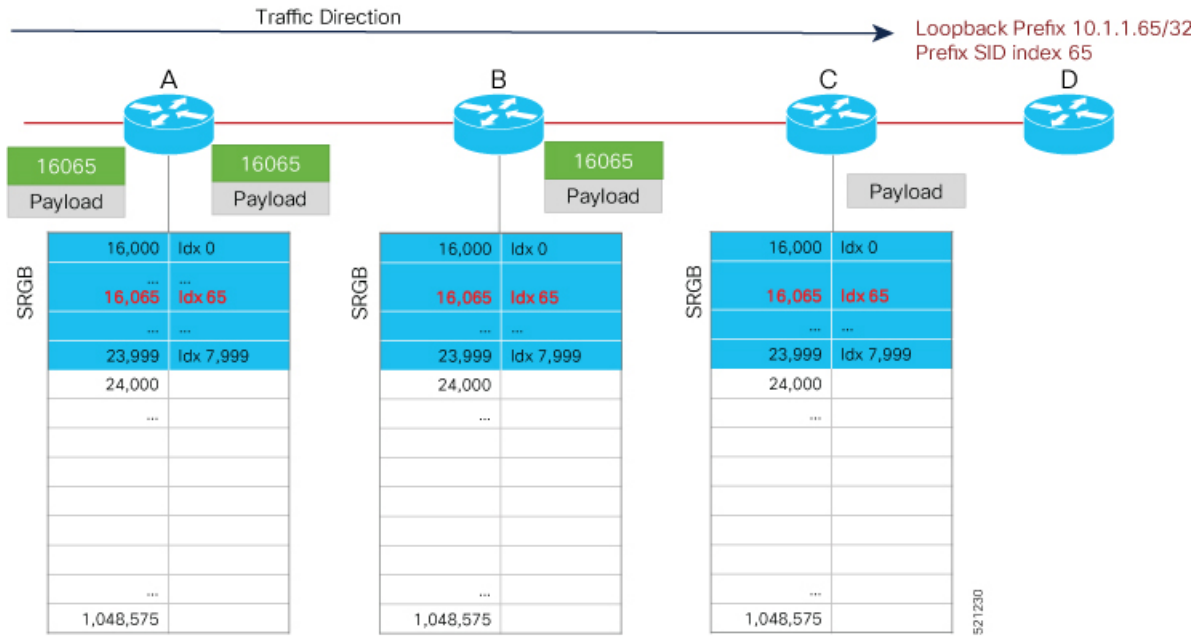
Configure Segment Routing Global Block and Segment Routing Local Block

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

- About the Segment Routing Global Block, on page 239
- About the Segment Routing Local Block, on page 241
- Understanding Segment Routing Label Allocation, on page 242
- Setup a Non-Default Segment Routing Global Block Range, on page 245
- Setup a Non-Default Segment Routing Local Block Range, on page 246

About the Segment Routing Global Block

The Segment Routing Global Block (SRGB) is a range of labels reserved for Segment Routing global segments. A prefix-SID is advertised as a domain-wide unique index. The prefix-SID index points to a unique label within the SRGB range. The index is zero-based, meaning that the first index is 0. The MPLS label assigned to a prefix is derived from the Prefix-SID index plus the SRGB base. For example, considering an SRGB range of 16,000 to 23,999, a prefix 1.1.1.65/32 with prefix-SID index of **65** is assigned the label value of **16065**.



To keep the configuration simple and straightforward, we strongly recommended that you use a homogenous SRGB (meaning, the same SRGB range across all nodes). Using a heterogenous SRGB (meaning, a different SRGB range of the same size across nodes) is also supported but is not recommended.

Behaviors and Limitations

- The default SRGB in IOS XR has a size of 8000 starting from label value 16000. The default range is 16000 to 23,999. With this size, and assuming one loopback prefix per router, an operator can assign prefix SIDs to a network with 8000 routers.
- There are instances when you might need to define a different SRGB range. For example:
  - Non-IOS XR nodes with a SRGB range that is different than the default IOS XR SRGB range.
  - The default SRGB range is not large enough to accommodate all required prefix SIDs.
- A non-default SRGB can be configured following these guidelines:
  - The SRGB starting value can be configured anywhere in the dynamic label range space (16,000 to 1,048,575).
  - The SRGB can be configured to any size value that fits within the dynamic label range space.
- Allocating an SRGB label range does not mean that all the labels in this range are programmed in the forwarding table. The label range is just reserved for SR and not available for other purposes. Furthermore, a platform may limit the number of local labels that can be programmed.
- We recommend that the non-default SRGB be configured under the **segment-routing** global configuration mode. By default, all IGP instances and BGP use this SRGB.
- You can also configure a non-default SRGB under the IGP, but it is not recommended.

SRGB Label Conflicts

When you define a non-default SRGB range, there might be a label conflict (for example, if labels are already allocated, statically or dynamically, in the new SRGB range). The following system log message indicates a label conflict:

```
%ROUTING-ISIS-4-SRGB_ALLOC_FAIL : SRGB allocation failed: 'SRGB reservation not
successful for [16000,80000], SRGB (16000 80000, SRGB_ALLOC_CONFIG_PENDING, 0x2)
(So far 16 attempts). Make sure label range is free'
```

To remove this conflict, you must reload the router to release the currently allocated labels and to allocate the new SRGB.

After the system reloads, LSD does not accept any dynamic label allocation before IS-IS/OSPF/BGP have registered with LSD. Upon IS-IS/OSPF/BGP registration, LSD allocates the requested SRGB (either the default range or the customized range).

After IS-IS/OSPF/BGP have registered and their SRGB is allocated, LSD starts serving dynamic label requests from other clients.

Note To avoid a potential router reload due to label conflicts, and assuming that the default SRGB size is large enough, we recommend that you use the default IOS XR SRGB range.

Note Allocating a non-default SRGB in the upper part of the MPLS label space increases the chance that the labels are available and a reload can be avoided.

Caution Modifying a SRGB configuration is disruptive for traffic and may require a reboot if the new SRGB is not available entirely.

About the Segment Routing Local Block

A local segment is automatically assigned an MPLS label from the dynamic label range. In most cases, such as TI-LFA backup paths and SR-TE explicit paths defined with IP addresses, this dynamic label allocation is sufficient. However, in some scenarios, it could be beneficial to allocate manually local segment label values to maintain label persistency. For example, an SR-TE policy with a manual binding SID that is performing traffic steering based on incoming label traffic with the binding SID.

The Segment Routing Local Block (SRLB) is a range of label values preserved for the manual allocation of local segments, such as adjacency segment identifiers (adj-SIDs), Layer 2 adj-SIDs, and binding SIDs (BSIDs). These labels are locally significant and are only valid on the nodes that allocate the labels.

Behaviors and Limitations

- The default SRLB has a size of 1000 starting from label value 15000; therefore, the default SRLB range goes from 15000 to 15,999.

- A non-default SRLB can be configured following these guidelines:
  - The SRLB starting value can be configured anywhere in the dynamic label range space (16,000 to 1,048,575).
  - The SRLB can be configured to any size value that fits within the dynamic label range space.

SRLB Label Conflicts

When you define a non-default SRLB range, there might be a label conflict (for example, if labels are already allocated, statically or dynamically, in the new SRLB range). In this case, the new SRLB range will be accepted, but not applied (pending state). The previous SRLB range (active) will continue to be in use.

To remove this conflict, you must reload the router to release the currently allocated labels and to allocate the new SRLB.

Caution You can use the **clear segment-routing local-block discrepancy all** command to clear label conflicts. However, using this command is disruptive for traffic since it forces all other MPLS applications with conflicting labels to allocate new labels.

Note To avoid a potential router reload due to label conflicts, and assuming that the default SRGB size is large enough, we recommend that you use the default IOS XR SRLB range.

Note Allocating a non-default SRLB in the upper part of the MPLS label space increases the chance that the labels are available and a reload can be avoided.

Understanding Segment Routing Label Allocation

In IOS XR, local label allocation is managed by the Label Switching Database (LSD). MPLS applications must register as a client with the LSD to allocate labels. Most MPLS applications (for example: LDP, RSVP, L2VPN, BGP [LU, VPN], IS-IS and OSPF [Adj-SID], SR-TE [Binding-SID]) use labels allocated dynamically by LSD.

With Segment Routing-capable IOS XR software releases, the LSD *preserves* the default SRLB label range (15,000 to 15,999) and default SRGB label range (16,000 to 23,999), even if Segment Routing is not enabled.

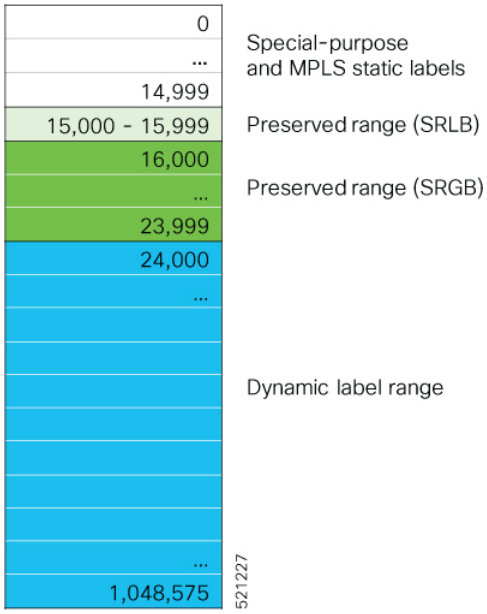
This preservation of the default SRLB/SRGB label range makes future Segment Routing activation possible without a reboot. No labels are allocated from this preserved range. When you enable Segment Routing with the default SRLB/SRGB in the future, these label ranges will be available and ready for use.

The LSD allocates dynamic labels starting from 24,000.

Note If an MPLS label range is configured and it overlaps with the default SRLB/SRGB label ranges (for example, **mpls label range 15000 1048575**), then the default SRLB/SRGB preservation is disabled.

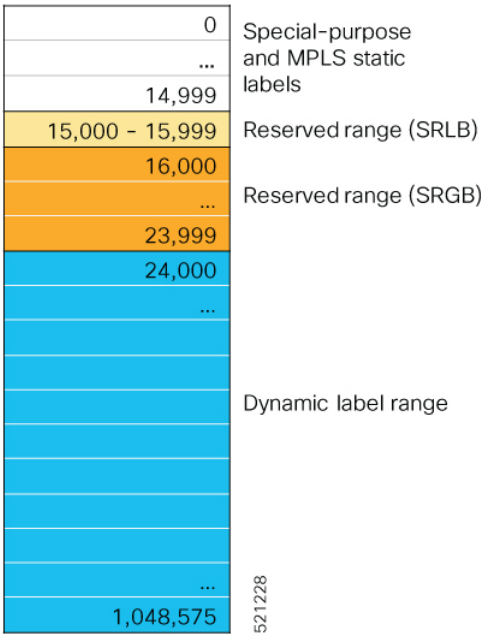
Example 1: LSD Label Allocation When SR is not Configured

- Special use: 0-15
- MPLS static: 16 to 14,999
- SRLB (preserved): 15,000 to 15,999
- SRGB (preserved): 16,000 to 23,999
- Dynamic: 24,000 to max



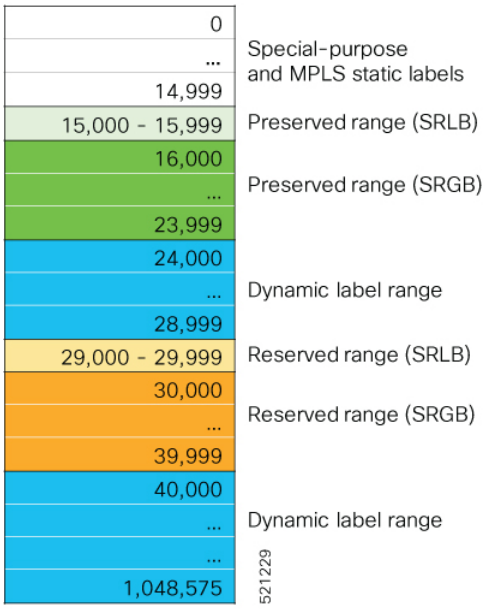
Example 2: LSD Label Allocation When SR is Configured with Default SRGB and Default SRLB

- Special use: 0-15
- MPLS static: 16 to 14,999
- SRLB (reserved): 15,000 to 15,999
- SRGB (reserved): 16,000 to 23,999
- Dynamic: 24,000 to max



Example 3: LSD Label Allocation When SR is Configured with Non-default SRGB and Non-default SRLB

- Special use: 0-15
- MPLS static: 16 to 14,999
- SRLB (preserved): 15,000 to 15,999
- SRGB (preserved): 16,000 to 23,999
- Dynamic: 24000 to 28,999
- SRLB (reserved): 29,000 to 29,999
- SRGB (reserved): 30,000 to 39,999
- Dynamic: 40,000 to max



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. **segment-routing global-block** *starting_value ending_value*
3. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**segment-routing global-block** *starting_value ending_value* **Example:** `RP/0/RP0/CPU0:router(config)# segment-routing global-block 16000 80000`	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.

	Command or Action	Purpose
Step 3	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Use the **show mpls label table** [**label** *label-value*] command to verify the SRGB configuration:

```
Router# show mpls label table label 16000 detail
Table Label   Owner                           State  Rewrite
----- ------- ------------------------------- ------ -------
0     16000   ISIS(A):1                       InUse  No
  (Lbl-blk SRGB, vers:0, (start_label=16000, size=64001)
```

What to do next

Configure prefix SIDs and enable segment routing.

Setup a Non-Default Segment Routing Local Block Range

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

SUMMARY STEPS

1. **configure**
2. **segment-routing local-block** *starting_value ending_value*
3. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

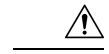
	Command or Action	Purpose
Step 2	**segment-routing local-block** *starting_value ending_value* **Example:** `RP/0/RP0/CPU0:router(config)# segment-routing local-block 30000 30999`	Enter the lowest value that you want the SRLB range to include as the starting value. Enter the highest value that you want the SRLB range to include as the ending value.
Step 3	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Use the **show mpls label table** [**label** *label-value*] [**detail**] command to verify the SRLB configuration:

```
Router# show mpls label table label 30000 detail

Table Label   Owner                              State  Rewrite
----- ------- ---------------------------------- ------ -------
0     30000   LSD(A)                             InUse  No
  (Lbl-blk SRLB, vers:0, (start_label=30000, size=1000, app_notify=0)

Router# show segment-routing local-block inconsistencies

No inconsistencies
```

The following example shows an SRLB label conflict in the range of 30000 and 30999. Note that the default SRLB is active and the configured SRLB is pending:

```
Router(config)# segment-routing local-block 30000 30999

%ROUTING-MPLS_LSD-3-ERR_SRLB_RANGE : SRLB allocation failed: 'SRLB reservation not successfull

for [30000,30999]. Use with caution 'clear segment-routing local-block discrepancy all'
command
to force srlb allocation'
```



Caution You can use the **clear segment-routing local-block discrepancy all** command to clear label conflicts. However, using this command is disruptive for traffic since it forces all other MPLS applications with conflicting labels to allocate new labels.

```
Router# show mpls label table label 30000 detail
```

```
Table Label   Owner                            State  Rewrite
----- ------- -------------------------------- ------ -------
0     30000   LSD(A)                           InUse  No
  (Lbl-blk SRLB, vers:0, (start_label=30000, size=1000, app_notify=0)

Router# show segment-routing local-block inconsistencies
SRLB inconsistencies range: Start/End: 30000/30999

Router# show mpls lsd private | i SRLB

SRLB Lbl Mgr:
   Current Active SRLB block      = [15000, 15999]
   Configured Pending SRLB block  = [30000, 30999]
```

Reload the router to release the currently allocated labels and to allocate the new SRLB:

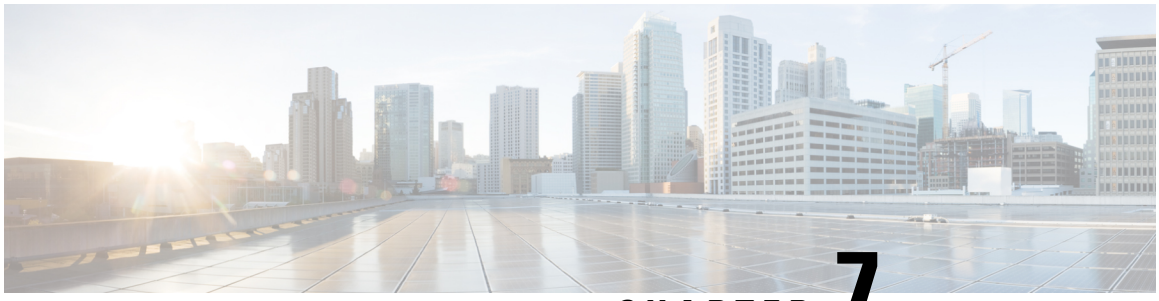
```
Router# reload

Proceed with reload? [confirm]yes
```

After the system is brought back up, verify that there are no label conflicts with the SRLB configuration:

```
Router# show mpls lsd private | i SRLB

SRLB Lbl Mgr:
   Current Active SRLB block      = [30000, 30999]
   Configured Pending SRLB block  = [0, 0]

Router# show segment-routing local-block inconsistencies

No inconsistencies
```

What to do next

Configure adjacency SIDs and enable segment routing.



CHAPTER 7

Configure Segment Routing for IS-IS Protocol

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

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

Note For additional information on implementing IS-IS on your Cisco 8000 Series Router, see the *Implementing IS-IS* module in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

- Enabling Segment Routing for IS-IS Protocol, on page 249
- Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface, on page 252
- Configuring an Adjacency SID, on page 263
- IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability, on page 269
- IS-IS Unreachable Prefix Announcement, on page 273
- IS-IS Partition Detection and Leakage of Specific Route Advertisement, on page 275
- Conditional Prefix Advertisement, on page 279

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
- 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. **router-id loopback** *loopback interface used for prefix-sid*
6. **segment-routing mpls** [**sr-prefer**]
7. **exit**
8. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis isp`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. **Note** You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**address-family** { **ipv4**	**ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 4	**metric-style wide** [**level** { **1**	**2** }] **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# metric-style wide level 1`	Configures a router to generate and accept only wide link metrics in the Level 1 area.

	Command or Action	Purpose
Step 5	**router-id loopback** *loopback interface used for prefix-sid* **Example:** `RP/0/(config-isis-af)#router-id loopback0`	Configures router ID for each address-family (ipv4/ipv6).

Step 6	**segment-routing mpls** [**sr-prefer**]
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)#`
	`segment-routing mpls`
		Segment routing is enabled by the following actions:
		• MPLS forwarding is enabled on all interfaces where IS-IS is active.
		• All known prefix-SIDs in the forwarding plain are programmed, with the prefix-SIDs advertised by remote routers or learned through local or remote mapping server.
		• The prefix-SIDs locally configured are advertised.
		Use the **sr-prefer** keyword to set the preference of segment routing (SR) labels over label distribution protocol (LDP) labels.

Step 7	**exit**
	Example:
	`RP/0/RP0/CPU0:router(config-isis-af)# exit`
	`RP/0/RP0/CPU0:router(config-isis)# exit`	

Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Configure the prefix SID.

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

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
Disable Penultimate Hop Popping	Release 7.5.4	You can now disable the penultimate hop popping (PHP) without adding an explicit-Null label. In earlier releases, you could disable PHP only by adding an explicit-Null label using the explicit-null keyword. The feature introduces the **php-disable** keyword under the **prefix-sid** command.

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

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

Strict-SPF SIDs are used to forward traffic strictly along the SPF path. 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). Strict-SPF SIDs are also used to program the backup paths for prefixes, node SIDs, and adjacency SIDs.

Penultimate-Hop-Popping (PHP) can be disabled for the Prefix SID. In this case, the penultimate hop does not pop the Prefix-SID before delivering the packet to the node that advertised the Prefix-SID; it is forwarded intact to the next hop. This can be useful in situations where the label needs to be retained for certain purposes, such as for traffic engineering or QoS policies.

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** [**algorithm** *algorithm-number*] {**index** *SID-index* | **absolute** *SID-value*} [**n-flag-clear**] [**explicit-null**] [**php-disable**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis 1`	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.
Step 3	**interface Loopback** *instance* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface Loopback0`	Specifies the loopback interface and instance.
Step 4	**address-family** { **ipv4**	**ipv6** } [**unicast**] **Example:** The following is an example for ipv4 address family: `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

Step 5	**prefix-sid** [**algorithm** *algorithm-number*] {**index** *SID-index*	**absolute** *SID-value*} [**n-flag-clear**] [**explicit-null**] [**php-disable**]
		Example:
		`RP/0/RP0/CPU0:router(config-isis-if-af)# prefix-sid index 1001`
		`RP/0/RP0/CPU0:router(config-isis-if-af)# prefix-sid absolute 17001`
			Configures the prefix-SID index or absolute value for the interface.
			Specify **algorithm** *algorithm-number* to configure SR Flexible Algorithm. See Enabling Segment Routing Flexible Algorithm, on page 489.
			Specify **index** *SID-index* for each node to create a prefix SID based on the lower boundary of the SRGB + the index.
			Specify **absolute** *SID-value* for each node to create a specific prefix SID within the SRGB.
			By default, the n-flag is set on the prefix-SID, indicating that it is a node SID. For specific prefix-SID (for example,

	Command or Action	Purpose
		Anycast prefix-SID), enter the **n-flag-clear** keyword. IS-IS does not set the N flag in the prefix-SID sub Type Length Value (TLV). To disable penultimate-hop-popping (PHP) and add explicit-Null label, enter **explicit-null** keyword. IS-IS sets the E flag in the prefix-SID sub TLV. Any upstream neighbor of the Prefix-SID originator replaces the Prefix-SID with a Prefix-SID having an Explicit NULL value. To disable penultimate-hop-popping (PHP), enter **php-disable** keyword. IS-IS sets the P flag in the prefix-SID sub TLV. The penultimate hop will not pop the Prefix-SID before delivering the packet to the node that advertised the Prefix-SID.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Verify the prefix-SID configuration:

```
RP/0/RP0/CPU0:router# show isis database verbose

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

Overriding MPLS Imposition (IP-to-MPLS) via Service Layer API (SL-API)

Table 27: Feature History Table

Feature Name	Release Information	Feature Description
Overriding MPLS Imposition (IP-to-MPLS) via Service Layer API (SL-API)	Release 24.2.11	In scenarios where SR-prefer is enabled, this feature allows you to specify SR prefixes through an Access Control List where their imposition forwarding entry (IP-to-MPLS) gives preference to SL-API, instead of the SR native LSP. The labeled forwarding entries (MPLS-to-MPLS or MPLS-to-IP) continue to follow the SR native LSP. This feature introduces the following command under Router RIB AF configuration mode: **segment-routing mpls preserve-label-forwarding access-list***acl_name* **[apply-inverse]**

Note For detailed information about Service Layer API (SL-API), refer to "Use Service Layer API to Bring your Controller on Cisco IOS XR Router" of the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- This feature is applicable when an SR prefix destination is also programmed via SL-API and “sr-prefer” is also enabled due to the presence of other prefixes with both SR and LDP LSPs.
- If the feature is configured for selected (allowed) prefixes, the “sr-prefer” configuration is ignored and the imposition forwarding entry follows the SL-API path instead of the SR native LSP.
- If the feature is not configured, or if a prefix is not allowed for SL-API steering, the “sr-prefer” configuration is honored and the imposition forwarding entry follows the SR native LSP.
- When there is a single source of programming for a destination (SR or SL-API), this feature has no impact on the forwarding.

- This feature is supported for programming of IPv4 SR prefixes.
- This feature is supported for programming of IPv6 SR prefixes.
- This feature does not support forwarding of traffic to IPv4 destinations recursing onto IPv6 next-hops steered over SL-API paths (BGPv4 over SRMPLS-v6).
- Redistribution of SL-API imposition route into another protocol is not supported.
- SR native and SL-API paths must always be labelled.
- The set of prefixes allowed for SR and LDP must be disjointed from the set of prefixes allowed for SR and SL-API. SR/SL-API and SR/LDP can co-exist across different SR prefixes.

Use Case

Assume a node is part of a network with SR and LDP enabled concurrently (ships-in-the-night) with preference to SR over LDP when both LSPs are present (sr-prefer).

The network operator relies on a controller to program a desired traffic-engineered path for specific prefix destinations using SL-API. The following forwarding behaviors are expected at the node programmed via SL-API:

- Imposition forwarding entry (IP-to-MPLS) gives preference to the SL-API LSP
- Labeled forwarding entries (MPLS-to-MPLS or MPLS-to-IP) follow the SR native LSP

Transport Without SL-API Injection

Consider the following :

- A network with SR and LDP enabled concurrently
- Nodes are configured with SR-prefer enabled
- Prefix 10.1.1.2/32 (SR prefix SID 16002) is programmed with SR native LSP
- Prefix 10.1.1.3/32 (SR prefix SID 16003) is programmed with SR native LSP
- Prefix 10.1.1.4/32 (SR prefix SID 16004) is programmed with both SR native and LDP LSPs
- When required, a controller is used to program a desired traffic-engineered path for allowed destination prefixes via an SL-API:
  - Allowed prefixes for controller steering: 10.1.1.2/32, 10.1.1.3/32
  - Not allowed prefix for controller steering: 10.1.1.4/32

When the controller does not trigger an SL-API path for allowed prefixes, the imposition forwarding entry follows the SR native LSP instead of the LDP LSP (as a result of configuring sr-prefer). The Swap/Pop forwarding entries are programmed to follow both SR native and LDP LSPs.

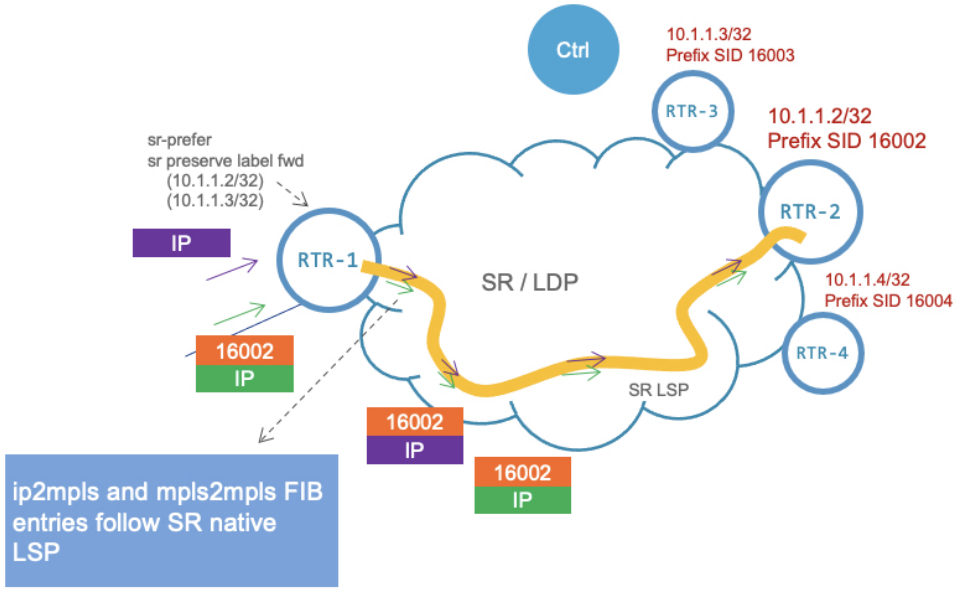
For SR prefixes 10.1.1.2/32 and 10.1.1.3/32:

- Imposition (IP2MPLS):
  - SR Prefix SID push forwarding entry programmed by IGP
- Swap/Pop (MPLS2MPLS/MPLS2IP):

- SR prefix SID local label swap/pop forwarding entry programmed by IGP

For SR prefix 10.1.1.4/32:

- Imposition (IP2MPLS):
  - SR Prefix SID push forwarding entry programmed by IGP
- Swap/Pop (MPLS2MPLS/MPLS2IP):
  - SR prefix SID local label swap/pop forwarding entry programmed by IGP
  - LDP local label swap/pop forwarding entry programmed by LDP



Transport After SL-API Injection

When the controller triggers an SL-API path for an allowed destination prefix (for example 10.1.1.2), the imposition forwarding entry will follow the SL-API LSP instead of the SR native LSP.

The imposition forwarding entry for allowed prefixes but not programmed by SL-API (for example 10.1.1.3/32), or not allowed prefixes (for example, 10.1.1.4), will follow the SR native LSP.

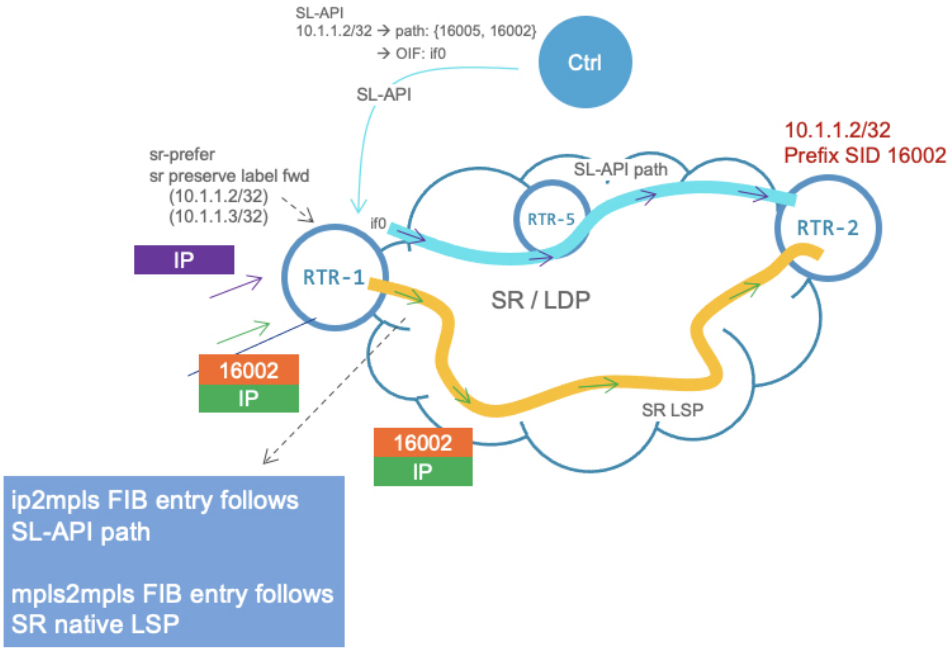
The Swap/Pop forwarding entries for the SR-only prefixes (10.1.1.2/32, 10.1.1.3/32) continue to be programmed by SR.

The Swap/Pop forwarding entries for the SR/LDP prefix (10.1.1.4/32) continue to be programmed by both SR and LDP.

For Prefix 10.1.1.2/32:

- Imposition (IP2MPLS):
  - Label stack push forwarding entry programed by SL-API
- Swap/Pop (MPLS2MPLS/MPLS2IP):
  - SR prefix SID local label swap/pop forwarding entry programmed by IGP

There are no changes to the forwarding entries for prefixes 10.1.1.3/32 and 10.1.1.4/32.



Configuration

To enable overriding of MPLS label imposition via SL-API, use the **segment-routing mpls preserve-label-forwarding access-list** *acl_name* **[apply-inverse]** command under **router rib address-family** {**ipv4**|**ipv6**} configuration mode.

Example

The following example shows how to allow IPv4 SR prefix 10.1.1.2/32 to have it MPLS imposition forwarding entry to be override via SL-API:

```
RP/0/RP0/CPU0:ios(config)# ipv4 access-list SL-API-PREFER-ALLOWED-PFX
RP/0/RP0/CPU0:ios(config-ipv4-acl)# 10 permit 10.1.1.2
RP/0/RP0/CPU0:ios(config-ipv4-acl)# exit

RP/0/RP0/CPU0:ios(config)# router rib address-family ipv4
RP/0/RP0/CPU0:ios(config-rib-afi)# segment-routing mpls preserve-label-forwarding access-list
 SL-API-PREFER-ALLOWED-PFX
```

The following example shows how to allow any IPv4 SR prefix except 10.1.1.2/32 to have it MPLS imposition forwarding entry to be override via SL-API:

```
RP/0/RP0/CPU0:ios(config)# ipv4 access-list SL-API-PREFER-DISALLOWED-PFX
RP/0/RP0/CPU0:ios(config-ipv4-acl)# 10 permit 10.1.1.2
RP/0/RP0/CPU0:ios(config-ipv4-acl)# exit

RP/0/RP0/CPU0:ios(config)# router rib address-family ipv4
RP/0/RP0/CPU0:ios(config-rib-afi)# segment-routing mpls preserve-label-forwarding access-list
 SL-API-PREFER-DISALLOWED-PFX apply-inverse
```

Verification

Consider the following SR prefix as allowed to be steered over an SL-API path:

- Prefix: 10.1.1.1/32
- Prefix SID: 20000

The SR native LSP programmed at a node in the network is as follows:

- Local label: 20000
- ECMP:
  - Path0 – out label: 20000; out int: Bundle-Ether20131
  - Path1 – out label: 20000; out int: Bundle-Ether20132
  - Path2 – out label: 20000; out int: Bundle-Ether20133
  - Path3 – out label: 20000; out int: Bundle-Ether20134

The SL-API LSP to be programmed by the controller at the same node is as follows:

- Local label: 100051 (dynamically allocated)
- Weighted ECMP:
  - Path0 – out label: 24000, 18001; out int: Bundle-Ether2012; weight: 1
  - Path1 – out label: 24000, 18001; out int: Bundle-Ether2013; weight: 2
  - Path2 – out label: 24001, 18001; out int: Bundle-Ether2014; weight: 4
  - Path3 – out label: 24001, 18001; out int: Bundle-Ether2015; weight: 8

The following sequence of show command outputs can be used to verify the programming of the imposition forwarding entry when an SL-API path is present.

The following output shows the RIB entry for an allowed prefix highlighting the fields to indicate that overriding of MPLS label imposition via SL-API is enabled:

```
Router# show route 10.1.1.1/32 detail

Routing entry for 10.1.1.1/32
  Known via "isis 0", distance 115, metric 500, labeled SR (label forwarding preserve),
type level-2
  Installed Jul 30 21:23:50.539 for 01:43:09
  Routing Descriptor Blocks
    100.201.201.2, from 199.1.0.2, via Bundle-Ether20131
      Route metric is 500
      Label: 0x4e20 (20000)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:4       Path ref count:0
      NHID:0x0(Ref:0)
      MPLS eid:0x109c700000001
    101.201.201.2, from 199.1.0.2, via Bundle-Ether20132
      Route metric is 500
      Label: 0x4e20 (20000)
      Tunnel ID: None
```

```
      Binding Label: None
      Extended communities count: 0
      Path id:3       Path ref count:0
      NHID:0x0(Ref:0)
      MPLS eid:0x109c700000001
    102.201.201.2, from 199.1.0.2, via Bundle-Ether20133
      Route metric is 500
      Label: 0x4e20 (20000)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:2       Path ref count:0
      NHID:0x0(Ref:0)
      MPLS eid:0x109c700000001
    103.201.201.2, from 199.1.0.2, via Bundle-Ether20134
      Route metric is 500
      Label: 0x4e20 (20000)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:0
      NHID:0x0(Ref:0)
      MPLS eid:0x109c700000001
  Route version is 0x47 (71)
  Local Label: 0x4e20 (20000)
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_NON_RECURSIVE_MEDIUM (7) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 1, Download Version 1793240
  Route eid: 0x109c700000001
  No advertising protos.
```

The following output shows the imposition forwarding entry programmed via SL-API (rewrite owner) in LSD. Observe the programmed paths and their parameters (output interface, output labels, and weight).

```
Router# show mpls lsd forwarding ipv4 detail | begin 10.1.1.1/32

'default':4U, 10.1.1.1/32, (100051)[SR Merge], 4 Paths,
       Owner=Static(A):Service-layer
   1/4: IPv4_STACK, 'default':4U, BE2012, BSID: NO_LABEL, nh=100.201.200.2, lbls={ 24000,
18001 }
            lbl flags= {0x0 0x0} ()}, ext_flags=0x0 path_flags=0x0
            nh-id=0x0, path-id=0, backup-path-id=0, load-metric=32, parent-intf=None,
path-set-id=0, path-priority=0
            Inner Stack Flags:  { 0x0}
            MPLS eid: N/A
   2/4: IPv4_STACK, 'default':4U, BE2013, BSID: NO_LABEL, nh=101.201.200.2, lbls={ 24000,
18001 }
            lbl flags= {0x0 0x0} ()}, ext_flags=0x0 path_flags=0x0
            nh-id=0x0, path-id=0, backup-path-id=0, load-metric=64, parent-intf=None,
path-set-id=0, path-priority=0
            Inner Stack Flags:  { 0x0}
            MPLS eid: N/A
   3/4: IPv4_STACK, 'default':4U, BE2014, BSID: NO_LABEL, nh=102.201.200.2, lbls={ 24001,
18001 }
            lbl flags= {0x0 0x0} ()}, ext_flags=0x0 path_flags=0x0
            nh-id=0x0, path-id=0, backup-path-id=0, load-metric=128, parent-intf=None,
path-set-id=0, path-priority=0
            Inner Stack Flags:  { 0x0}
            MPLS eid: N/A
   4/4: IPv4_STACK, 'default':4U, BE2015, BSID: NO_LABEL, nh=103.201.200.2, lbls={ 24001,
18001 }
```

```
            lbl flags= {0x0 0x0} ()}, ext_flags=0x0 path_flags=0x0
            nh-id=0x0, path-id=0, backup-path-id=0, load-metric=256, parent-intf=None,
path-set-id=0, path-priority=0
            Inner Stack Flags:  { 0x0}
            MPLS eid: N/A
      BCDL priority:1, LSD queue:9, version:178429,
      flags: 0x8, fwd_flags: 0x100 (sr_lbl_fwd_preserve),
      Installed Jul 30 21:26:25.111 (01:42:45 ago)
      Prefix eid: 0x1275100000001

. . .
```

The following output shows that the CEF imposition forwarding entry prefers the SL-API path.

```
Router# show cef 10.1.1.1/32 detail location 0/0/CPU0

10.1.1.1/32, version 178429, internal 0x1000001 0x110 (ptr 0xa0ea1428) [3], 0x0 (0x1182a378),
 0xa28 (0x202c3ba8)
 Updated Jul 30 21:26:25.635
 local adjacency to Bundle-Ether2012

 Prefix Len 32, traffic index 0, precedence n/a, priority 1, encap-id 0x1275100000001
  gateway array (0x1d140ef8) reference count 750, flags 0x68, source lsd (5), 1 backups
                [251 type 5 flags 0x8401 (0x8c18bda0) ext 0x0 (0x0)]
  LW-LDI[type=5, refc=3, ptr=0x1182a378, sh-ldi=0x8c18bda0]
  gateway array update type-time 1 Jul 30 21:26:25.532
 LDI Update time Jul 30 21:26:25.532
 LW-LDI-TS Jul 30 21:26:25.635
   via 100.201.200.2/32, Bundle-Ether2012, 7 dependencies, weight 32, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x90fa8028 0x0]
    next hop 100.201.200.2/32
    local adjacency
     local label 100051      labels imposed {24000 18001}
   via 101.201.200.2/32, Bundle-Ether2013, 7 dependencies, weight 64, class 0 [flags 0x0]
    path-idx 1 NHID 0x0 [0x90fa85c8 0x0]
    next hop 101.201.200.2/32
    local adjacency
     local label 100051      labels imposed {24000 18001}
   via 102.201.200.2/32, Bundle-Ether2014, 7 dependencies, weight 128, class 0 [flags 0x0]
    path-idx 2 NHID 0x0 [0x90fa82f8 0x0]
    next hop 102.201.200.2/32
    local adjacency
     local label 100051      labels imposed {24001 18001}
   via 103.201.200.2/32, Bundle-Ether2015, 7 dependencies, weight 256, class 0 [flags 0x0]
    path-idx 3 NHID 0x0 [0x90fa8190 0x0]
    next hop 103.201.200.2/32
    local adjacency
     local label 100051      labels imposed {24001 18001}

    Weight distribution:
    slot 0, weight 32, normalized_weight 1, class 0
    slot 1, weight 64, normalized_weight 2, class 0
    slot 2, weight 128, normalized_weight 4, class 0
    slot 3, weight 256, normalized_weight 8, class 0
    Load distribution: 0 1 1 2 2 2 2 3 3 3 3 3 3 3 3 (refcount 251)

    Hash  OK  Interface                 Address
    0     Y   Bundle-Ether2012          100.201.200.2
    1     Y   Bundle-Ether2013          101.201.200.2
    2     Y   Bundle-Ether2013          101.201.200.2
    3     Y   Bundle-Ether2014          102.201.200.2
    4     Y   Bundle-Ether2014          102.201.200.2
    5     Y   Bundle-Ether2014          102.201.200.2
```

```
    6     Y   Bundle-Ether2014          102.201.200.2
    7     Y   Bundle-Ether2015          103.201.200.2
    8     Y   Bundle-Ether2015          103.201.200.2
    9     Y   Bundle-Ether2015          103.201.200.2
    10    Y   Bundle-Ether2015          103.201.200.2
    11    Y   Bundle-Ether2015          103.201.200.2
    12    Y   Bundle-Ether2015          103.201.200.2
    13    Y   Bundle-Ether2015          103.201.200.2
    14    Y   Bundle-Ether2015          103.201.200.2
```

The following output shows the backup CEF imposition forwarding entry if the SL-API path is not present. Observe that it follows the SR native LSP.

```
Router# show cef 10.1.1.1/32 backup detail location 0/0/CPU0

10.1.1.1/32, version 1793240, priority 1, flags 0x200000, flags2 0x81, extn_flags 0x2100,
source rib (7), ctx-flags 0xc1
 Updated Jul 30 21:23:50.819
 Prefix Len 32
 Label count = 1, src = 7, label = 20000
   via  BE20131 (0xf013954) 100.201.201.2, weight 0, class 0 [flags 0x0]
    next hop VRF - 'default', table - 0xe0000000
    Output labels {20000}
   via  BE20132 (0xf01395c) 101.201.201.2, weight 0, class 0 [flags 0x0]
    next hop VRF - 'default', table - 0xe0000000
    Output labels {20000}
   via  BE20133 (0xf013964) 102.201.201.2, weight 0, class 0 [flags 0x0]
    next hop VRF - 'default', table - 0xe0000000
    Output labels {20000}
   via  BE20134 (0xf01396c) 103.201.201.2, weight 0, class 0 [flags 0x0]
    next hop VRF - 'default', table - 0xe0000000
    Output labels {20000}
```

The following output shows the labeled forwarding entry (MPLS-to-MPLS) for the SR prefix SID local label (20000). Observe that it follows the SR native LSP.

```
Router# show mpls forwarding labels 20000 location 0/0/CPU0

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
20000  20000       SR Pfx (idx 4000)  BE20131      100.201.201.2   0
       20000       SR Pfx (idx 4000)  BE20132      101.201.201.2   0
       20000       SR Pfx (idx 4000)  BE20133      102.201.201.2   0
       20000       SR Pfx (idx 4000)  BE20134      103.201.201.2   0
```

The following output shows the details for the labeled forwarding entry (MPLS-to-MPLS) for the SR prefix SID local label (20000).

```
Router# show mpls forwarding labels 20000 detail location 0/0/CPU0

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
20000  20000       SR Pfx (idx 4000)  BE20131      100.201.201.2   0
     Updated: Jul 30 21:26:25.158
     Version: 1793240, Priority: 15
     Label Stack (Top -> Bottom): { 20000 }
     NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 0, Weight: 0
     MAC/Encaps: 14/18, MTU: 1500
     Outgoing Interface: Bundle-Ether20131 (ifhandle 0x0f013954)
```

```
    Packets Switched: 0

      20000       SR Pfx (idx 4000)  BE20132      101.201.201.2   0
    Updated: Jul 30 21:26:25.158
    Version: 1793240, Priority: 15
    Label Stack (Top -> Bottom): { 20000 }
    NHID: 0x0, Encap-ID: N/A, Path idx: 1, Backup path idx: 0, Weight: 0
    MAC/Encaps: 14/18, MTU: 1500
    Outgoing Interface: Bundle-Ether20132 (ifhandle 0x0f01395c)
    Packets Switched: 0

      20000       SR Pfx (idx 4000)  BE20133      102.201.201.2   0
    Updated: Jul 30 21:26:25.158
    Version: 1793240, Priority: 15
    Label Stack (Top -> Bottom): { 20000 }
    NHID: 0x0, Encap-ID: N/A, Path idx: 2, Backup path idx: 0, Weight: 0
    MAC/Encaps: 14/18, MTU: 1500
    Outgoing Interface: Bundle-Ether20133 (ifhandle 0x0f013964)
    Packets Switched: 0

      20000       SR Pfx (idx 4000)  BE20134      103.201.201.2   0
    Updated: Jul 30 21:26:25.158
    Version: 1793240, Priority: 15
    Label Stack (Top -> Bottom): { 20000 }
    NHID: 0x0, Encap-ID: N/A, Path idx: 3, Backup path idx: 0, Weight: 0
    MAC/Encaps: 14/18, MTU: 1500
    Outgoing Interface: Bundle-Ether20134 (ifhandle 0x0f01396c)
    Packets Switched: 0

Traffic-Matrix Packets/Bytes Switched: 0/0
```

Configuring an Adjacency SID

An adjacency SID (Adj-SID) is associated with an adjacency to a neighboring node. The adjacency SID steers the traffic to a specific adjacency. Adjacency SIDs have local significance and are only valid on the node that allocates them.

An adjacency SID can be allocated dynamically from the dynamic label range or configured manually from the segment routing local block (SRLB) range of labels.

Adjacency SIDs that are dynamically allocated do not require any special configuration, however there are some limitations:

- A dynamically allocated Adj-SID value is not known until it has been allocated, and a controller will not know the Adj-SID value until the information is flooded by the IGP.
- Dynamically allocated Adj-SIDs are not persistent and can be reallocated after a reload or a process restart.
- Each link is allocated a unique Adj-SID, so the same Adj-SID cannot be shared by multiple links.

Manually allocated Adj-SIDs are persistent over reloads and restarts. They can be provisioned for multiple adjacencies to the same neighbor or to different neighbors. You can specify that the Adj-SID is protected. If the Adj-SID is protected on the primary interface and a backup path is available, a backup path is installed. By default, manual Adj-SIDs are not protected.

Adjacency SIDs are advertised using the existing IS-IS Adj-SID sub-TLV. The S and P flags are defined for manually allocated Adj-SIDs.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|F|B|V|L|S|P|   |
+-+-+-+-+-+-+-+-+
```

Table 28: Adjacency Segment Identifier (Adj-SID) Flags Sub-TLV Fields

Field	Description
S (Set)	This flag is set if the same Adj-SID value has been provisioned on multiple interfaces.
P (Persistent)	This flag is set if the Adj-SID is persistent (manually allocated).

Manually allocated Adj-SIDs are supported on point-to-point (P2P) interfaces.

This task explains how to configure an Adj-SID on an interface.

Before you begin

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

Use the **show mpls label table detail** command to verify the SRLB range.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface** *type interface-path-id*
4. **point-to-point**
5. **address-family** { **ipv4** | **ipv6** } [**unicast**]
6. **adjacency-sid** {**index** *adj-SID-index* | **absolute** *adj-SID-value* } [**protected**]
7. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis 1`	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.

	Command or Action	Purpose
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet0/0/0/7`	Specifies the interface and enters interface configuration mode.
Step 4	**point-to-point** **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# point-to-point`	Specifies the interface is a point-to-point interface.
Step 5	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** The following is an example for ipv4 address family: `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

Step 6	**adjacency-sid** {**index** *adj-SID-index* | **absolute** *adj-SID-value* } [**protected**]
	Example:
	`RP/0/RP0/CPU0:router(config-isis-if-af)# adjacency-sid index 10`
	`RP/0/RP0/CPU0:router(config-isis-if-af)# adjacency-sid absolute 15010`
		Configures the Adj-SID index or absolute value for the interface.
		Specify **index** *adj-SID-index* for each link to create an Ajd-SID based on the lower boundary of the SRLB + the index.
		Specify **absolute** *adj-SID-value* for each link to create a specific Ajd-SID within the SRLB.
		Specify if the Adj-SID is **protected**. For each primary path, if the Adj-SID is protected on the primary interface and a backup path is available, a backup path is installed. By default, manual Adj-SIDs are not protected.

Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Verify the Adj-SID configuration:

```
RP/0/RP0/CPU0:router# show isis segment-routing label adjacency persistent
```

```
Mon Jun 12 02:44:07.085 PDT

IS-IS 1 Manual Adjacency SID Table

15010 AF IPv4
      GigabitEthernet0/0/0/3: IPv4, Protected 1/65/N, Active
      GigabitEthernet0/0/0/7: IPv4, Protected 2/66/N, Active

15100 AF IPv6
      GigabitEthernet0/0/0/3: IPv6, Not protected 255/255/N, Active
```

Verify the labels are added to the MPLS Forwarding Information Base (LFIB):

```
RP/0/RP0/CPU0:router# show mpls forwarding labels 15010
Mon Jun 12 02:50:12.172 PDT
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
15010  Pop         SRLB (idx 10)      Gi0/0/0/3    10.0.3.3        0
       Pop         SRLB (idx 10)      Gi0/0/0/7    10.1.0.5        0
       16004       SRLB (idx 10)      Gi0/0/0/7    10.1.0.5        0           (!)
       16004       SRLB (idx 10)      Gi0/0/0/3    10.0.3.3        0           (!)
```

Manually Configure a Layer 2 Adjacency SID

Typically, an adjacency SID (Adj-SID) is associated with a Layer 3 adjacency to a neighboring node, to steer the traffic to a specific adjacency. If you have Layer 3 bundle interfaces, where multiple physical interfaces form a bundle interface, the individual Layer 2 bundle members are not visible to IGP; only the bundle interface is visible.

You can configure a Layer 2 Adj-SID for the individual Layer 2 bundle interfaces. This configuration allows you to track the availability of individual bundle member links and to verify the segment routing forwarding over the individual bundle member links, for Operational Administration and Maintenance (OAM) purposes.

A Layer 2 Adj-SID can be allocated dynamically or configured manually.

- IGP dynamically allocates Layer 2 Adj-SIDs from the dynamic label range for each Layer 2 bundle member. A dynamic Layer 2 Adj-SID is not persistent and can be reallocated as the Layer 3 bundle link goes up and down.
- Manually configured Layer 2 Adj-SIDs are persistent if the Layer 3 bundle link goes up and down. Layer 2 Adj-SIDs are allocated from the Segment Routing Local Block (SRLB) range of labels. However, if the configured value of Layer 2 Adj-SID does not fall within the available SRLB, a Layer 2 Adj-SID will not be programmed into forwarding information base (FIB).

Restrictions

- Adj-SID forwarding requires a next-hop, which can be either an IPv4 address or an IPv6 address, but not both. Therefore, manually configured Layer 2 Adj-SIDs are configured per address-family.
- Manually configured Layer 2 Adj-SID can be associated with only one Layer 2 bundle member link.
- A SID value used for Layer 2 Adj-SID cannot be shared with Layer 3 Adj-SID.
- SR-TE using Layer 2 Adj-SID is not supported.

This task explains how to configure a Layer 2 Adj-SID on an interface.

Before you begin

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

Use the **show mpls label table detail** command to verify the SRLB range.

SUMMARY STEPS

1. **configure**
2. **segment-routing**
3. **adjacency-sid**
4. **interface** *type interface-path-id*
5. **address-family** { **ipv4** | **ipv6** } [**unicast**]
6. **l2-adjacency sid** {**index** *adj-SID-index* | **absolute** *adj-SID-value* } [**next-hop** {*ipv4_address* | *ipv6_address* }]
7. Use the **commit** or **end** command.
8. **end**
9. **router isis** *instance-id*
10. **address-family** { **ipv4** | **ipv6** } [**unicast**]
11. **segment-routing bundle-member-adj-sid**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**segment-routing** **Example:** `RP/0/RP0/CPU0:Router(config)# segment-routing`	Enters segment routing configuration mode.
Step 3	**adjacency-sid** **Example:** `RP/0/RP0/CPU0:Router(config-sr)# adjacency-sid`	Enters adjacency SID configuration mode.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:Router(config-sr-adj)# interface GigabitEthernet0/0/0/3`	Specifies the interface and enters interface configuration mode.

	Command or Action	Purpose
Step 5	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:Router(config-sr-adj-intf)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 6	**l2-adjacency sid** {**index** *adj-SID-index* | **absolute** *adj-SID-value* } [**next-hop** { *ipv4_address* | *ipv6_address* }] **Example:** `RP/0/RP0/CPU0:Router(config-sr-adj-intf-af)# l2-adjacency sid absolute 15015 next-hop 10.1.1.4`	Configures the Adj-SID index or absolute value for the interface. Specify **index** *adj-SID-index* for each link to create an Ajd-SID based on the lower boundary of the SRLB + the index. Specify **absolute** *adj-SID-value* for each link to create a specific Ajd-SID within the SRLB. For point-to-point interfaces, you are not required to specify a next-hop. However, if you do specify the next-hop, the Layer 2 Adj-SID will be used only if the specified next-hop matches the neighbor address. For LAN interfaces, you must configure the next-hop IPv4 or IPv6 address. If you do not configure the next-hop, the Layer 2 Adj-SID will not be used for LAN interface.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**end**	
Step 9	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:Router(config)# router isis isp`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.
Step 10	**address-family** { **ipv4** | **ipv6** } [**unicast**] **Example:** `RP/0/RP0/CPU0:Router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

	Command or Action	Purpose
Step 11	**segment-routing bundle-member-adj-sid** **Example:** `RP/0/RP0/CPU0:Router(config-isis-af)#` **`segment-routing bundle-member-adj-sid`**	Programs the dynamic Layer 2 Adj-SIDs, and advertises both manual and dynamic Layer 2 Adj-SIDs. **Note** This command is not required to program manual L2 Adj-SID, but is required to program the dynamic Layer 2 Adj-SIDs and to advertise both manual and dynamic Layer 2 Adj-SIDs.

Verify the configuration:

```
Router# show mpls forwarding detail | i "Pop|Outgoing Interface|Physical Interface"
Tue Jun 20 06:53:51.876 PDT
. . .
15001  Pop         SRLB (idx 1)       BE1          10.1.1.4        0
     Outgoing Interface: Bundle-Ether1 (ifhandle 0x000000b0)
     Physical Interface: GigabitEthernet0/0/0/3 (ifhandle 0x000000b0)


Router# show running-config segment-routing
Tue Jun 20 07:14:25.815 PDT
segment-routing
 adjacency-sid
  interface GigabitEthernet0/0/0/3
   address-family ipv4 unicast
    l2-adjacency-sid absolute 15001
   !
  !
 !
!
```

Associated Commands

- l2-adjacency sid
- segment-routing bundle-member-adj-sid

IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability

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

- Prefix Attribute Flags
- IPv4 and IPv6 Source Router ID

Prefix Attribute Flags

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

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

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

Prefix Attribute Flags Sub-TLV Format

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

Prefix Attribute Flags Sub-TLV Fields

Field	Description
X (External Prefix Flag)	This flag is set if the prefix has been redistributed from another protocol. The value of the flag is preserved when the prefix is propagated to another level.
R (Re-advertisement Flag)	This flag is set to 1 by the Level 1-2 router when the prefix is propagated between IS-IS levels (from Level 1 to Level 2, or from Level 2 to Level 1). 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).
N (Node Flag)	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 **prefix-attributes n-flag-clear** is configured (see Configuring Prefix Attribute N-flag-clear, on page 271). 2. Set to 0 if **prefix-sid** {**index***SID-index* | **absolute** *SID-value*} {**n-flag-clear**] is configured (see Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface, on page 252). 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 29: Source Router Sub-TLV Format

IPv4 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. **router isis** *instance-id*
3. **interface Loopback** *instance*
4. **prefix-attributes n-flag-clear** [**Level-1** | **Level-2**]
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

	Command or Action	Purpose
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis 1`	
Step 3	**interface Loopback** *instance* **Example:** `RP/0/RP0/CPU0:router(config)# interface Loopback0`	Specifies the loopback interface.
Step 4	**prefix-attributes n-flag-clear** [**Level-1** | **Level-2**] **Example:** `RP/0/RP0/CPU0:router(config-if)# isis prefix-attributes n-flag-clear`	Clears the prefix attribute N-flag explicitly.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Verify the prefix attribute configuration:

```
RP/0/RP0/CPU0:router# show isis database verbose

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

```
  Metric: 10         IP-Extended 10.0.0.2/32
    Prefix-SID Index: 1002, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:0 N:1
    Source Router ID: 10.0.0.2
<...>
```

IS-IS Unreachable Prefix Announcement

Table 30: Feature History Table

Feature Name	Release	Description
IS-IS Unreachable Prefix Announcement	Release 7.8.1	The Unreachable Prefix Announcement (UPA) notifies the loss of prefix reachability between areas or domains, for prefixes that are covered by the summary address range during inter-area or inter-domain summarization. This feature helps in identifying the routers that are facing prefix unreachability issues faster and fix it. The new commands introduced for this feature are: • summary-prefix • prefix-unreachable

The organization of networks into levels or areas and/or IGP domains helps to limit the scope of link-state information within certain boundaries. However, the state that is related to prefix reachability often requires propagation across these areas (Level1/Level2) or domains (Autonomous System Boundary Router (ASBR)). An Autonomous System Boundary Router (ASBR) is a router that is running multiple protocols and serves as a gateway to routers outside the Open Shortest Path First (OSPF) domain and those operating with different protocols.

Route summarization, also known as route aggregation, is a method to minimize the number of routing tables in an IP network. It consolidates selected multiple routes into a single route advertisement.

Techniques such as summarization address the scale challenges associated with the advertizement of the individual prefix state outside of local area/domain. MPLS architecture did not allow for the effective use of the summarization due to its end-to-end Label Switched Path (LSP) requirement. With the introduction of the SRv6, which does not have such requirement, the use of summarization has become important again.

Summarization results in suppression of the individual prefix state that is useful for triggering fast-convergence mechanisms outside of the Interior Gateway Routing Protocols (IGPs (for example - Border Gateway Protocol - Prefix Independent Convergence (BGP PIC) Edge).

This feature enables the notification of the individual prefixes becoming unreachable in its area/domain, when the summarization is used between areas/domains to advertise the reachability for these prefixes.

There are existing SRv6 deployments that use summarization and require fast detection of the egress Provider Edge (PE) going down. To address these deployments in timely manner, we use the existing Protocol Data Units (PDUs) and Tag-Length-Values (TLVs), which is based on the Prefix Unreachability Advertisement (UPA).

Configuration Steps

The configuration steps that are required to set up the Unreachable Prefix Announcement (UPA) feature are as follows:

- **UPA Advertisement**

  An existing IS-IS address-family submode **summary-prefix** command was extended for UPA advertisement.

```
Router(config)#router isis 1
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#summary-prefix beef:10::/32 level 2 adv-unreachable
Router(config-isis-af)#summary-prefix beef:11::/32 level 2 algorithm 128 adv-unreachable
 unreachable-component-tag 777
Router(config-isis-af)#commit
```

- **Prefix Unreachable**

  The new **prefix-unreachable** command includes new commands that control the UPA advertisement such as, lifetime, metric, limit the maximum number if UPAs and UPA processing. For more details see, prefix-unreachable

```
Router(config)#router isis 1
Router(config-isis)#address-family ipv6
Router(config-isis-af)#prefix-unreachable
Router(config-isis-prefix-unreachable)#adv-lifetime 500
Router(config-isis-prefix-unreachable)#adv-metric 4261412866
Router(config-isis-prefix-unreachable)#adv-maximum 77
Router(config-isis-prefix-unreachable)#rx-process-enable
Router(config-isis-prefix-unreachable)#commit
```

Running Configuration

Execute the following show commands to review the L1/L2 (area) or ASBR (domain) running configuration:

Run the **show run router isis 1 address-family ipv6 unicast** command to view the summary prefix under as well as UPA parameters under it.

```
Router#sh run router isis 1 address-family ipv6 unicast
router isis 1
 address-family ipv6 unicast
  advertise application lfa link-attributes srlg
  advertise link attributes
  prefix-unreachable
   adv-lifetime 300
  !
  summary-prefix 10::/64
  summary-prefix beef:10::/32 adv-unreachable
  summary-prefix beef:11::/32 algorithm 128 adv-unreachable
  summary-prefix ceef:10::/32 adv-unreachable
  propagate level 2 into level 1 route-policy L2_TO_L1
  segment-routing srv6
   locator USID_ALG0
   !
   locator USID_ALG128
   !
  !
 !
!
```

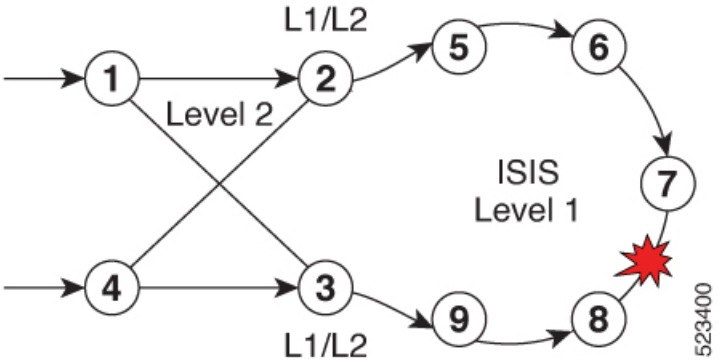
IS-IS Partition Detection and Leakage of Specific Route Advertisement

Table 31: Feature History Table

Feature Name	Release	Description
IS-IS Partition Detection and Leakage of Specific Route Advertisements	Release 7.10.1	In an open ring topology, a single fiber cut may partition the area or domain into two pieces. With summarization enabled, the area (domain) partition may result in traffic drops. Depending on the configuration in the Area Border Routers (ABRs) or Autonomous System Boundary Routers (ASBRs) that is picked as an entry point to the partitioned area (domain), the traffic is delivered to its destination or dropped as unreachable at ABR or ASBR. IS-IS partition detection and leakage of specific route advertisements features are introduced to retain connectivity for the partitioned area (domain) when summarization is used. The ABRs or ASBRs detect a network partition within an area (domain) and upon detection, ensure that the summary route is replaced with specific route advertisements in IS-IS. The following command and keyword are introduced: The feature introduces these changes: **New Command:** • **partition-detect** **Modified Command:** • The **partition-repair** keyword is introduced in the **summary-prefix** command. **YANG Data Model:** • New XPaths for `Cisco-IOS-XR-um-router-isis-cfg.yang` (see GitHub, YANG Data Models Navigator)

In service provider networks, the Layer 1 (L1) area is often represented by a set of routers connected in a ring. Sometimes the ring is not closed (for example, Area Border Routers (ABRs) are not connected directly inside the L1 area). In such cases, a single fiber cut partitions the area into two pieces. Route Summarization is basically advertising many routes into one route, also called route aggregation. When the partition is detected, summarization is suppressed, and all previously summarized prefixes are advertised in IS-IS. The individual prefix advertisements preserve the connectivity end-to-end.

Figure 22: Interarea Topology with L1 Area in an Open Ring



This feature addresses the following summarization problems:

- **Area Partition Detection and Avoidance**: Area partition detection only works for summarization from L1 to L2. It is not supported for summarization from L2 to L1. Ensure you have router-id that is configured for the address-family (IPv4 or IPv6) for which you are enabling the partition detection and avoidance.

Note Router-id must be enabled for this feature to work. Area partition is tracked for each algorithm, algo 0, and any enabled flex-algo, independently. For flex-algo,the Area Border Router must participate in the algo for the tracking to work for such flex-algo

- **Domain Partition Detection and Avoidance**: Networks use muti-domain design, where they split their network into multiple IGP domains. They redistribute between domains and summarize during the redistribution. An IGP domain may represent an open ring and a single link cut may split it into two parts.

Configuration Steps

Configure IS-IS Partition Detection and Leakage of Specific as follows:

- **Area Partition Detection and Avoidance**: To configure the area partition detection and avoidance there are two configuration steps:

  1. For each summary prefix that you want the area partition and avoidance to work, enable it with the summary-prefix command:

```
Router(config)#router isis 1
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#summary-prefix 2001:DB8::/32 level 2 partition-repair
Router(config-isis-af)#summary-prefix 2001:DB9::/32 level 2 algorithm 128
partition-repair
Router(config-isis-af)#commit
```

  2. Configure the tracking of the Area Border Router (ABR) reachability. Here, the example is for two ABRs, but you can enable the partition for as many ABRs in the area.

```
Router(config)#router isis 1
Router(config-isis)#address-family ipv6 unicast
```

```
Router(config-isis-af)#router-id 2001:DB8:1::1
Router(config-isis-af)#summary-prefix 2001:DB8::/32 level 2 partition-repair
Router(config-isis-af)#summary-prefix 2001:DB9::/32 level 2 algorithm 128
partition-repair
Router(config-isis-af)#partition-detect
Router(config-isis-af)#track 2001:DB8:4::4
Router(config-isis-af)#commit
```

```
Router(config)#router isis 1
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#router-id 2001:DB8:4::4
Router(config-isis-af)#summary-prefix 2001:DB8::/32 level 2 partition-repair
Router(config-isis-af)#summary-prefix 2001:DB9::/32 level 2 algorithm 128
partition-repair
Router(config-isis-af)#partition-detect
Router(config-isis-af)#track 2001:DB8:1::1
Router(config-isis-af)#commit
```

- **Domain Partition Detection and Avoidance**: It is similar to Area Partition and requires two configuration steps:

  1. For each summary prefix that you want the area partition and avoidance to work, enable it with the summary-prefix command:

```
Router(config)#router isis 1
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#summary-prefix 2001:DB8::/32 level 2 partition-repair
Router(config-isis-af)#summary-prefix 2001:DB9::/32 level 2 algorithm 128
partition-repair
Router(config-isis-af)#commit
```

  2. Configure the tracking of the Area Border Router reachability. The example is for two ABRs, but you can enable the partition for as many ABRs in the area.

     To track ASBR, two identifiers are required:

     - First is the internal router-id in the instance under which the configuration is done (similar to area partition).
     - Second is the address of the ASBR in the other domain that is redistributed to the instance where the configuration is done.

Note The implementation of the feature ensures that the reachability of the external-address is only tracked in algorithm 0. The loss of the external-address is used for algo 0 and all flex-algos. The reachability of the internal address is kept per flex-algo.

```
Router(config)#router isis 2
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#router-id 2001:DB8:1::1
Router(config-isis-af)#summary-prefix 2001:DB8::/32 level 2 partition-repair
Router(config-isis-af)#summary-prefix 2001:DB9::/32 level 2 algorithm 128
partition-repair
Router(config-isis-af)#partition-detect
```

```
Router(config-isis-af)#track 2001:DB8:4::4 external-address 2001:DB8:10::4
Router(config-isis-af)#commit


Router(config)#router isis 2
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#router-id 2001:DB8:4::4
Router(config-isis-af)#summary-prefix 2001:DB8::/32 level 2 partition-repair
Router(config-isis-af)#summary-prefix 2001:DB9::/32 level 2 algorithm 128
partition-repair
Router(config-isis-af)#partition-detect
Router(config-isis-af)#track 2001:DB8:1::1external-address 2001:DB8:10::1
Router(config-isis-af)#commit
```

Verification

- **For Area Partition Detection and Avoidance**

  Use the show command **show isis instance 1 flex-algo 128** to check if the area partition is detected:

```
Router# show isis instance 1 flex-algo 128
IS-IS 1 Flex-Algo Database
Flex-Algo 128:
Level-2:
Definition Priority: 128
Definition Source: plzen.00, (Local)
Definition Equal to Local: Yes
Definition Metric Type: IGP
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
Reverse Exclude Any Affinity Bit Positions:
Reverse Include Any Affinity Bit Positions:
Reverse Include All Affinity Bit Positions:
Exclude SRLGs:
Disabled: No

Level-1:
Definition Priority: 128
Definition Source: plzen.00, (Local)
Definition Equal to Local: Yes
Definition Metric Type: IGP
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
Reverse Exclude Any Affinity Bit Positions:
Reverse Include Any Affinity Bit Positions:
Reverse Include All Affinity Bit Positions:
Exclude SRLGs:
Disabled: No

Topologies supported:
IPv4 Unicast
Partition-Detect:
ABR: Internal-Address: 10.4.4.4
ASBR: Internal-Address: 10.4.4.4 External-Address: 10.10.10.4
ABR: Internal-Address: 10.5.5.5
IPv6 Unicast
Partition-Detect:
ABR: Internal-Address: 2001:DB8:4::4
```

```
Local Priority: 128
FRR Disabled: No
Microloop Avoidance Disabled: No
Data Plane Segment Routing: Yes
Data Plane IP: No
```

- **Domain Partition Detection and Avoidance**

  Usethe show command **show isis instance 1 flex-algo 128** to check if the domain partition is detected:

```
Router# show isis instance 1 flex-algo 128
IS-IS 1 Flex-Algo Database
Flex-Algo 128:

Level-2:
Definition Priority: 128
Definition Source: plzen.00, (Local)
Definition Equal to Local: Yes
Definition Metric Type: IGP
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
Reverse Exclude Any Affinity Bit Positions:
Reverse Include Any Affinity Bit Positions:
Reverse Include All Affinity Bit Positions:
Exclude SRLGs:
Disabled: No

Level-1:
Definition Priority: 128
Definition Source: plzen.00, (Local)
Definition Equal to Local: Yes
Definition Metric Type: IGP
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
Reverse Exclude Any Affinity Bit Positions:
Reverse Include Any Affinity Bit Positions:
Reverse Include All Affinity Bit Positions:
Exclude SRLGs:
Disabled: No

Topologies supported:
IPv4 Unicast
Partition-Detect:
ABR: Internal-Address: 10.4.4.4 (Active)
ASBR: Internal-Address: 10.4.4.4 External-Address: 10.10.10.4 (Active)
ABR: Internal-Address: 10.5.5.5
IPv6 Unicast
Partition-Detect:
ABR: Internal-Address: 2001:DB8:4::4 (Active)
ASBR: Internal-Address: 2001:DB8:4::4 External-Address: 2001:DB8:10::4 (Active)
```

Conditional Prefix Advertisement

In some situations, it's beneficial to make the IS-IS prefix advertisement conditional. For example, an Area Border Router (ABR) or Autonomous System Boundary Router (ASBR) that has lost its connection to one of the areas or autonomous systems (AS) might keep advertising a prefix. If an ABR or ASBR advertises the

Segment Routing (SR) SID with this prefix, the label stack of the traffic routed toward the disconnected area or AS might use this SID, which would result in dropped traffic at the ABR or ASBR.

ABRs or ASBRs are often deployed in pairs for redundancy and advertise a shared Anycast prefix SID. Conditional Prefix Advertisement allows an ABR or an ASBR to advertise its Anycast SID only when connected to a specific area or domain. If an ABR or ASBR becomes disconnected from the particular area or AS, it stops advertising the address for a specified interface (for example, Loopback).

Configure the conditional prefix advertisement under a specific interface. The prefix advertisement on this interface is associated with the route-policy that tracks the presence of a set of prefixes (prefix-set) in the Routing Information Base (RIB).

For faster convergence, the route-policy used for conditional prefix advertisement uses the new event-based **rib-has-route async** condition to notify IS-IS of the following situations:

- When the last prefix from the prefix-set is removed from the RIB.
- When the first prefix from the prefix-set is added to the RIB.

Configuration

To use the conditional prefix advertisement in IS-IS, create a prefix-set to be tracked. Then create a route policy that uses the prefix-set.

```
Router(config)# prefix-set prefix-set-name
Router(config-pfx)# prefix-address-1/length[, prefix-address-2/length,,,
prefix-address-16/length]
Router(config-pfx)# end-set

Router(config)# route-policy rpl-name
Router(config-rpl)# if rib-has-route async prefix-set-name then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
```

To advertise the loopback address in IS-IS conditionally, use the **advertise prefix route-policy** command under IS-IS interface address-family configuration sub-mode.

```
Router(config)# router isis 1
Router(config-isis)# interface Loopback0
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# advertise prefix route-policy rpl-name
Router(config-isis-if-af)# commit
```

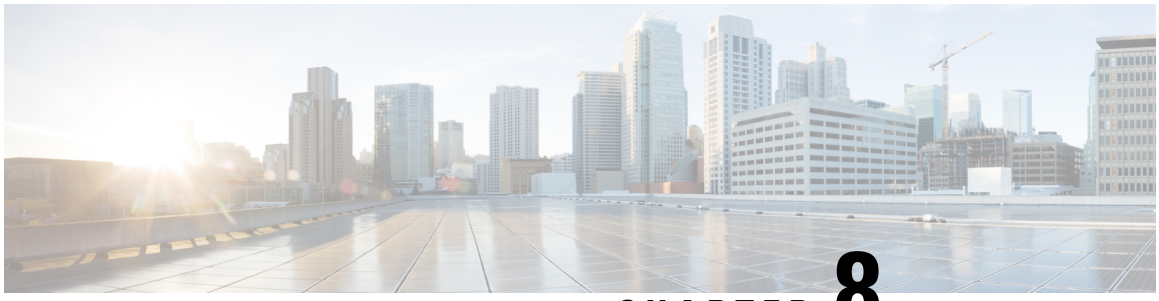
Example

```
Router(config)# prefix-set domain_2
Router(config-pfx)# 2.3.3.3/32, 2.4.4.4/32
Router(config-pfx)# end-set
Router(config)# route-policy track_domain_2
Router(config-rpl)# if rib-has-route async domain_2 then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config)# router isis 1
Router(config-isis)# interface Loopback0
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-if-af)# advertise prefix route-policy track_domain-2
```

```
Router(config-isis-if-af)# commit
```

Running Configuration

```
prefix-set domain_2
  2.3.3.3/32,
  2.4.4.4/32
end-set
!
route-policy track_domain_2
  if rib-has-route async domain_2 then
    pass
  endif
end-policy
!
router isis 1
 interface Loopback0
  address-family ipv4 unicast
   advertise prefix route-policy track_domain_2
  !
 !
!
```



CHAPTER 8

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.

- Enabling Segment Routing for OSPF Protocol, on page 283
- Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface, on page 285

Enabling Segment Routing for OSPF Protocol

Segment routing on the OSPF control plane supports the following:

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

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

Before you begin

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

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

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **segment-routing mpls**
4. **segment-routing sr-prefer**
5. **area 0**
6. **segment-routing mpls**
7. **exit**
8. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enables OSPF routing for the specified routing process and places the router in router configuration mode.
Step 3	**segment-routing mpls** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# segment-routing mpls`	Enables segment routing using the MPLS data plane on the routing process and all areas and interfaces in the routing process. Enables segment routing fowarding on all interfaces in the routing process and installs the SIDs received by OSPF in the forwarding table.
Step 4	**segment-routing sr-prefer** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# segment-routing sr-prefer`	Sets the preference of segment routing (SR) labels over label distribution protocol (LDP) labels.
Step 5	**area 0** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode.
Step 6	**segment-routing mpls** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# segment-routing mpls`	(Optional) Enables segment routing using the MPLS data plane on the area and all interfaces in the area. Enables segment routing fowarding on all interfaces in the area and installs the SIDs received by OSPF in the forwarding table.

	Command or Action	Purpose

Step 7	**exit**
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# exit`
	`RP/0/RP0/CPU0:router(config-ospf)# exit`	

Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Configure the prefix SID.

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

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

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

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

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

Before you begin

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

SUMMARY STEPS

1. **configure**

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

DETAILED STEPS

Procedure

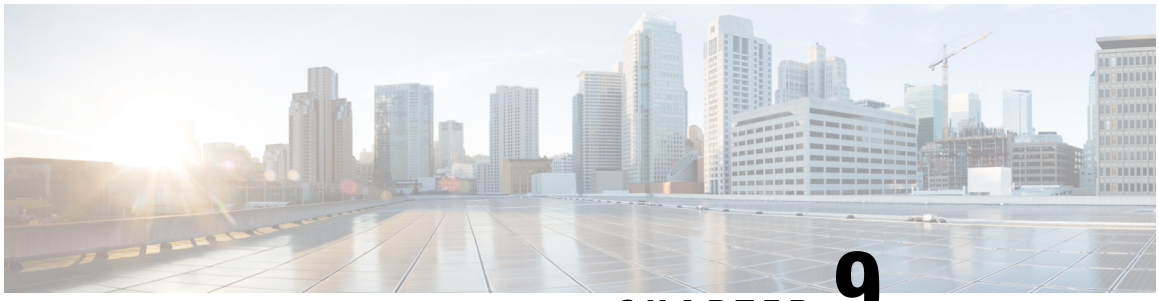
	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
Step 3	**area** *value* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 0`	Enters area configuration mode.
Step 4	**interface Loopback** *interface-instance* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface Loopback0 passive`	Specifies the loopback interface and instance.

Step 5	**prefix-sid**{**index** *SID-index* | **absolute** *SID-value* } [**n-flag-clear**] [**explicit-null**]
	Example:
	`RP/0/RP0/CPU0:router(config-ospf-ar)# prefix-sid index 1001`
	`RP/0/RP0/CPU0:router(config-ospf-ar)# prefix-sid absolute 17001`
		Configures the prefix-SID index or absolute value for the interface.
		Specify **index** *SID-index* for each node to create a prefix SID based on the lower boundary of the SRGB + the index.
		Specify **absolute** *SID-value* for each node to create a specific prefix SID within the SRGB.
		By default, the n-flag is set on the prefix-SID, indicating that it is a node SID. For specific prefix-SID (for example, Anycast prefix-SID), enter the `n-flag-clear` keyword. OSPF does not set the `N` flag in the prefix-SID sub Type Length Value (TLV).
		To disable penultimate-hop-popping (PHP) and add an explicit-Null label, enter the `explicit-null` keyword. OSPF sets the `E` flag in the prefix-SID sub TLV.

	Command or Action	Purpose
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Verify the prefix-SID configuration:

```
RP/0/RP0/CPU0:router# show ospf database opaque-area 7.0.0.1 self-originate
 OSPF Router with ID (10.0.0.1) (Process ID 1)
                Type-10 Opaque Link Area Link States (Area 0)
<...>
    Extended Prefix TLV: Length: 20
      Route-type: 1
      AF        : 0
      Flags     : 0x40
      Prefix    : 10.0.0.1/32

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



CHAPTER 9

Configure Segment Routing for BGP

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

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

Note For additional information on implementing BGP on your router, see the *BGP Configuration Guide for Cisco 8000 Series Routers*.

- Segment Routing for BGP, on page 289
- Configure BGP Prefix Segment Identifiers, on page 290
- Segment Routing Egress Peer Engineering, on page 291
- Configure BGP Link-State, on page 327
- Configurable Filters for IS-IS advertisements to BGP-Link State, on page 332
- Configure BGP Proxy Prefix SID, on page 334
- BGP Best Path Computation using SR Policy Paths, on page 337
- SRv6 Double Recursion for Multi-Layer BGP Underlay, on page 343

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, on page 239 section for information about the SRGB.

Note You must enable SR and explicitly configure the SRGB before configuring SR BGP. The SRGB must be explicitly configured, even if you are using the default range (16000 – 23999). BGP uses the SRGB and the index in the BGP prefix-SID attribute of a learned BGP-LU advertisement to allocate a local label for a given destination.

If SR and the SRGB are enabled after configuring BGP, then BGP is not aware of the SRGB, and therefore it allocates BGP-LU local labels from the dynamic label range instead of from the SRGB. In this case, restart the BGP process in order to allocate BGP-LU local labels from 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 *Routing Configuration Guide for Cisco 8000 Series Routers*.

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/RP0/CPU0:router(config)# segment-routing global-block 16000 23999

RP/0/RP0/CPU0:router(config)# route-policy SID($SID)
RP/0/RP0/CPU0:router(config-rpl)# set label-index $SID
RP/0/RP0/CPU0:router(config-rpl)# end policy

RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1
```

```
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# network 1.1.1.3/32 route-policy SID(3)
RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all
RP/0/RP0/CPU0:router(config-bgp-af)# commit
RP/0/RP0/CPU0:router(config-bgp-af)# end
```

```
RP/0/RP0/CPU0:router# show bgp 1.1.1.3/32
BGP routing table entry for 1.1.1.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 74          74
    Local Label: 16003
Last Modified: Sep 29 19:52:18.155 for 00:07:22
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  3
    99.3.21.3 from 99.3.21.3 (1.1.1.3)
      Received Label 3
      Origin IGP, metric 0, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 74
      Origin-AS validity: not-found
      Label Index: 3
```

Segment Routing Egress Peer Engineering

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
BGP PeerSet SID	Release 7.3.2	BGP peer SIDs are used to express source-routed interdomain paths and are of two types: Peer Node SIDs and Peer Adjacency SIDs. This release supports a new type of BGP peering SID, called BGP Peer Set SID. It is a group or set of BGP peer SIDs, that can provide load balancing over BGP neighbors (nodes) or links (adjacencies). The BGP peer Set SID can be associated with any combination of Peer Node SIDs or Peer Adjacency SIDs.

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

Below are the BGP-EPE peering SID types:

- PeerNode SID—To an eBGP peer. Pops the label and forwards the traffic on any interface to the peer.
- PeerAdjacency SID—To an eBGP peer via interface. Pops the label and forwards the traffic on the related interface.
- PeerSet SID—To a set of eBGP peers. Pops the label and forwards the traffic on any interface to the set of peers. All the peers in a set might not be in the same AS.

  Multiple PeerSet SIDs can be associated with any combination of PeerNode SIDs or PeerAdjacency SIDs.

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.

Usage Guidelines and Limitations

- When enabling BGP EPE, you must enable MPLS encapsulation on the egress interface connecting to the eBGP peer. This can be done by enabling either BGP labeled unicast (BGP-LU) address family or MPLS static for the eBGP peer.

  For information about BGP-LU, refer to the Implementing BGP chapter in the *BGP Configuration Guide for Cisco 8000 Series Routers*.

  For information about MPLS static, refer to the Implementing MPLS Static Labeling chapter in the *MPLS Configuration Guide for Cisco 8000 Series Routers*.

- Note the following points related to the IP-lookup backup support for EPEs:
  - This feature works only when you enable the **epe backup enable**, under the Global Address Family ID (AFI).
  - With this feature, an IP-Lookup backup is installed for each Egress Peer Engineering. This means, when all the paths of that EPE go down, the Forwarding Information Base (FIB) table searches in the IP table for the destination IP address in the data packet and forwards them accordingly.
  - The peer-set EPEs have a backup installed only when the mentioned CLI knob is enabled.

Configure Segment Routing Egress Peer Engineering

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

SUMMARY STEPS

1. **router bgp** *as-number*
2. **neighbor** *ip-address*
3. **remote-as** *as-number*
4. **egress-engineering**
5. **exit**
6. **mpls static**
7. **interface** *type interface-path-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 1`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 2	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.10.10.2`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 3	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 3`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 4	**egress-engineering** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# egress-engineering`	Configures the egress node with EPE for the eBGP peer.

Step 5	**exit**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# exit`
	`RP/0/RP0/CPU0:router(config-bgp)# exit`
	`RP/0/RP0/CPU0:router(config)#`	

Step 6	**mpls static** **Example:** `RP/0/RP0/CPU0:router(config)# mpls static`	Configure MPLS static on the egress interface connecting to the eBGP peer.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-static)# interface`	Specifies the egress interface connecting to the eBGP peer.

	Command or Action	Purpose
	`GigabitEthernet0/0/1/2`	

```
router bgp 1
 neighbor 10.10.10.2
  remote-as 3
  egress-engineering
 !
!
mpls static
 interface GigabitEthernet0/0/1/2
!
```

Configuring Manual BGP-EPE Peering SIDs

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
Manual BGP-EPE Peer SIDs	Release 7.3.2	BGP Peering SIDs that are allocated dynamically are not persistent and can be reallocated after a reload or a process restart. This feature allows you to manually configure BGP Egress Peer Engineering (EPE) Peering SIDs. This functionality provides predictability, consistency, and reliability if there are system reloads or process restarts.

Configuring manual BGP-EPE Peer SIDs allows for persistent EPE label values. Manual BGP-EPE SIDs are advertised through BGP-LS and are allocated from the Segment Routing Local Block (SRLB). See Configure Segment Routing Global Block and Segment Routing Local Block, on page 239 for information about the SRLB.

Each PeerNode SID, PeerAdjacency SID, and PeerSet SID is configured with an index value. This index serves as an offset from the configured SRLB start value and the resulting MPLS label (SRLB start label + index) is assigned to these SIDs. This label is used by CEF to perform load balancing across the individual BGP PeerSet SIDs, BGP PeerNode SID, or ultimately across each first-hop adjacency associated with that BGP PeerNode SID or BGP PeerSet SID.

Configuring Manual PeerNode SID

Each eBGP peer will be associated with a PeerNode SID index that is configuration driven.

```
RP/0/0/CPU0:PE1(config)# router bgp 10
RP/0/0/CPU0:PE1(config-bgp)# neighbor 10.10.10.2
RP/0/0/CPU0:PE1(config-bgp-nbr)# remote-as 20
RP/0/0/CPU0:PE1(config-bgp-nbr)# egress-engineering
RP/0/0/CPU0:PE1(config-bgp-nbr)# peer-node-sid index 600
```

Configuring Manual PeerAdjacency SID

Any first-hop for which an adjacency SID is configured needs to be in the resolution chain of at least one eBGP peer that is configured for egress-peer engineering. Otherwise such a kind of "orphan" first-hop with regards to BGP has no effect on this feature. This is because BGP only understands next-hops learnt by the BGP protocol itself and in addition only the resolving IGP next-hops for those BGP next-hops.

```
RP/0/0/CPU0:PE1(config)# router bgp 10
RP/0/0/CPU0:PE1(config-bgp)# adjacencies
RP/0/0/CPU0:PE1(config-bgp-adj)# 1.1.1.2
RP/0/0/CPU0:PE1(config-bgp-adj)# adjacency-sid index 500
```

Configuring Manual PeerSet SID

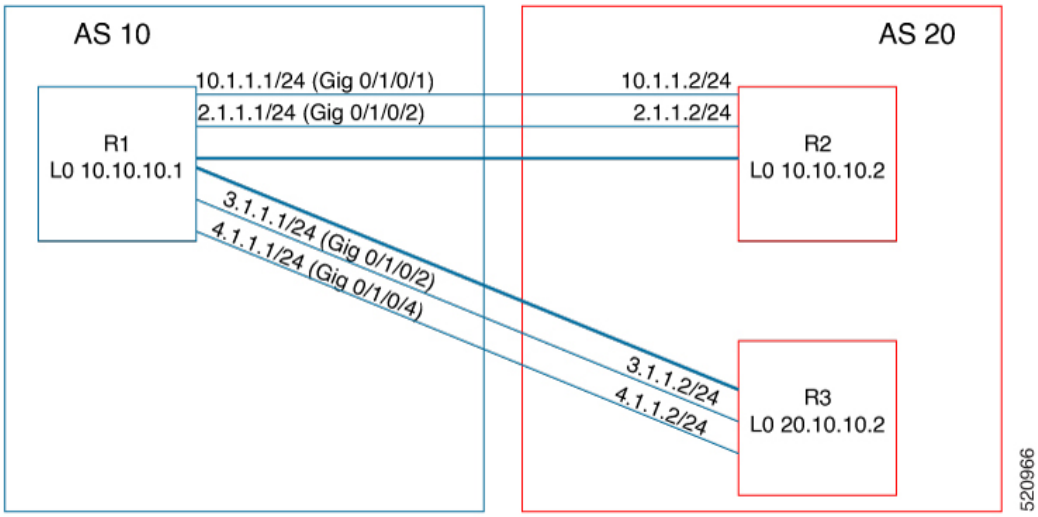
The PeerSet SID is configured under global Address Family. This configuration results in the creation of a Peer-Set SID EPE object.

```
RP/0/0/CPU0:PE1(config)# router bgp 10
RP/0/0/CPU0:PE1(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:PE1(config-bgp-afi)# peer-set-id 1
RP/0/0/CPU0:PE1(config-bgp-peer-set)# peer-set-sid 300
```

Example

Topology

The example in this section uses the following topology.



In this example, BGP-EPE peer SIDs are allocated from the default SRLB label range (15000 – 15999). The BGP-EPE peer SIDs are configured as follows:

- PeerNode SIDs to 10.10.10.2 with index 600 (label 15600), and for 20.10.10.2 with index 700 (label 15700)
- PeerAdj SID to link 1.1.1.2 with index 500 (label 15500)
- PeerSet SID 1 to load balance over BGP neighbors 10.10.10.1 and 20.10.10.2 with SID index 300 (label 15300)

- PeerSet SID 2 to load balance over BGP neighbor 20.10.10.2 and link 1.1.1.2 with SID index 400 (label 15400)

Configuration on R1

```
router bgp 10
 address-family ipv4 unicast
  peer-set-id 1
   peer-set-sid index 300
  !
  peer-set-id 2
   peer-set-sid index 400
  !
 !
 adjacencies
  1.1.1.2
   adjacency-sid index 500
   peer-set 2
  !
 !
 neighbor 10.10.10.2
  remote-as 20
  egress-engineering
  peer-node-sid index 600
  peer-set 1
 !
 neighbor 20.10.10.2
  egress-engineering
  peer-node-sid index 700
  peer-set 1
  peer-set 2
 !
```

To further show the load balancing of this example:

- 15600 is load balanced over {1.1.1.1 and 2.1.1.1}
- 15700 is load balanced over {3.1.1.1 and 4.1.1.1}
- 15500 is load balanced over {1.1.1.1}
- 15300 is load balanced over {1.1.1.1, 2.1.1.1, 3.1.1.1 and 4.1.1.1}
- 15400 is load balanced over {1.1.1.1, 3.1.1.1 and 4.1.1.1}

Advertising EPE-Enabled BGP Neighbors via BGP-LU

Table 34: Feature History Table

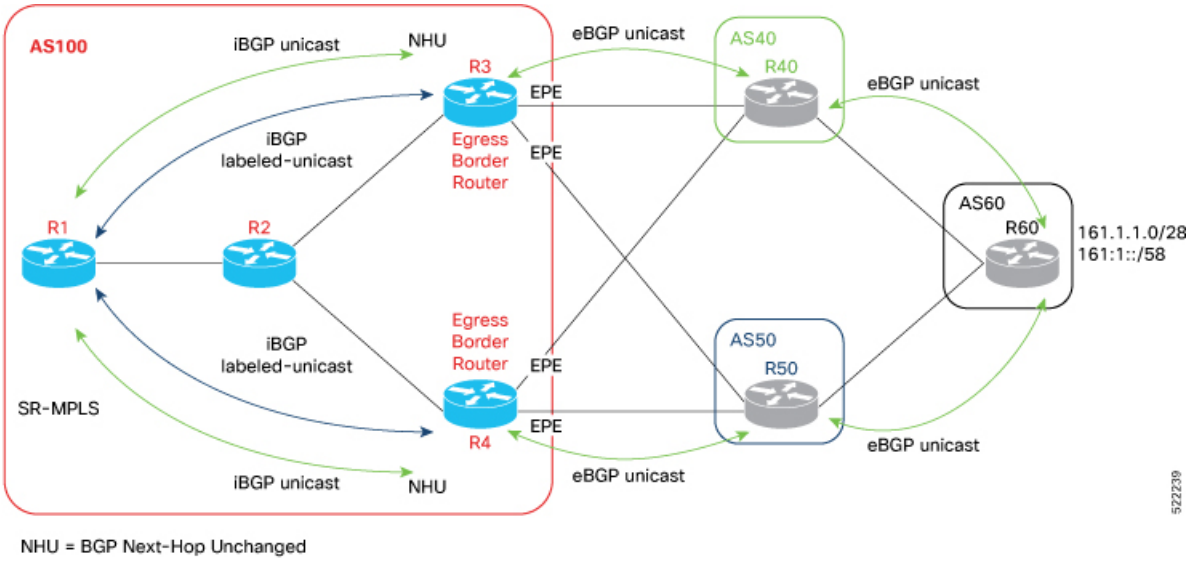
Feature Name	Release	Description
Advertising EPE-Enabled BGP Neighbors via BGP-LU	Release 7.3.3	BGP peering segments/SIDs are part of the Segment Routing Centralized BGP Egress Peer Engineering solution (BGP-EPE). A BGP-EPE-enabled border router allocates and programs BGP peering SIDs (EPE labels) to steer traffic over a specific external interface/BGP neighbor to reach a particular destination. This feature provides an alternate BGP-EPE solution leveraging BGP peering segments. It allows a BGP-EPE-enabled border router to use BGP Labeled Unicast (BGP-LU) to advertise the IP address of a neighbor with an LU label equal to the EPE label assigned to that neighbor.

BGP peering segments/SIDs are part of the Segment Routing Centralized BGP Egress Peer Engineering solution (BGP-EPE), as described in IETF RFC 9087. A BGP-EPE-enabled border router allocates and programs BGP peering SIDs (EPE labels) to steer traffic over a specific external interface/BGP neighbor to reach a particular destination.

This feature provides an alternate BGP-EPE solution leveraging BGP peering segments. It allows a BGP-EPE-enabled border router to use BGP Labeled Unicast (BGP-LU) to advertise the IP address of a neighbor with an LU label equal to the EPE label assigned to that neighbor.

The following figure illustrates a Segment Routing network (AS100) connected to a pair of transit Autonomous Systems. The egress border routers (R3 and R4) have BGP Peering segments (EPE) enabled on their eBGP neighbors in AS40 and AS50. Prefixes are propagated inside AS100 via BGP. R3 and R4 maintain the BGP next-hops unchanged. In addition, BGP labeled unicast is enabled inside AS100 to advertise the IP address of these eBGP neighbors.

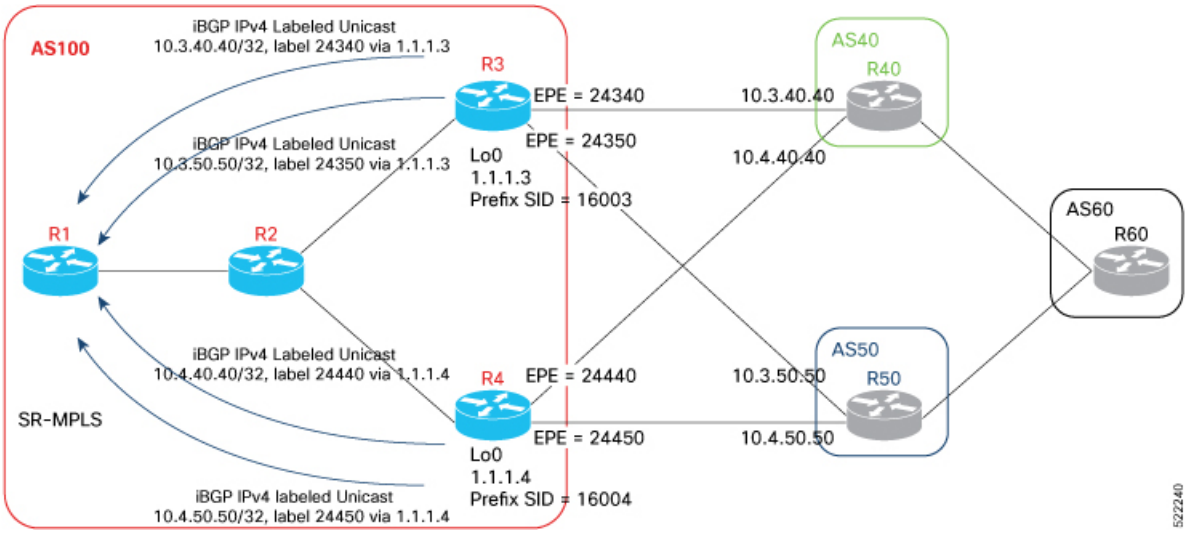
Figure 23: Solution Overview



NHU = BGP Next-Hop Unchanged

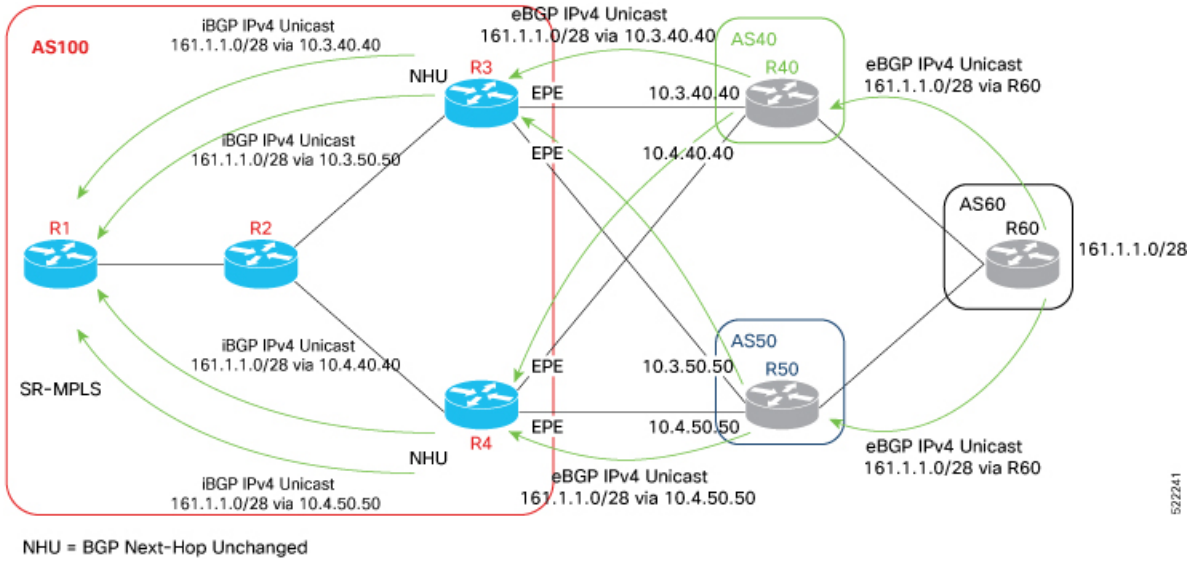
The figure below depicts the BGP-LU advertisements originated by R3 and R4 for the IP addresses of their eBGP neighbors. The figure also indicates the EPE label values assigned to each eBGP neighbor. Note that the local BGP-LU label on the egress border router is equal to the EPE label assigned to that neighbor.

Figure 24: Advertising EPE-Enabled BGP Neighbors via BGP-LU



In the following figure, an overlay prefix 161.1.1.0/28 originating at AS60 is advertised inside AS100. Egress border routers are configured to advertise all of their paths. Note that the BGP next-hops are not modified. In this example, the ingress router in AS100 (R1) learns the overlay prefix via 4 paths (one for each eBGP neighbor).

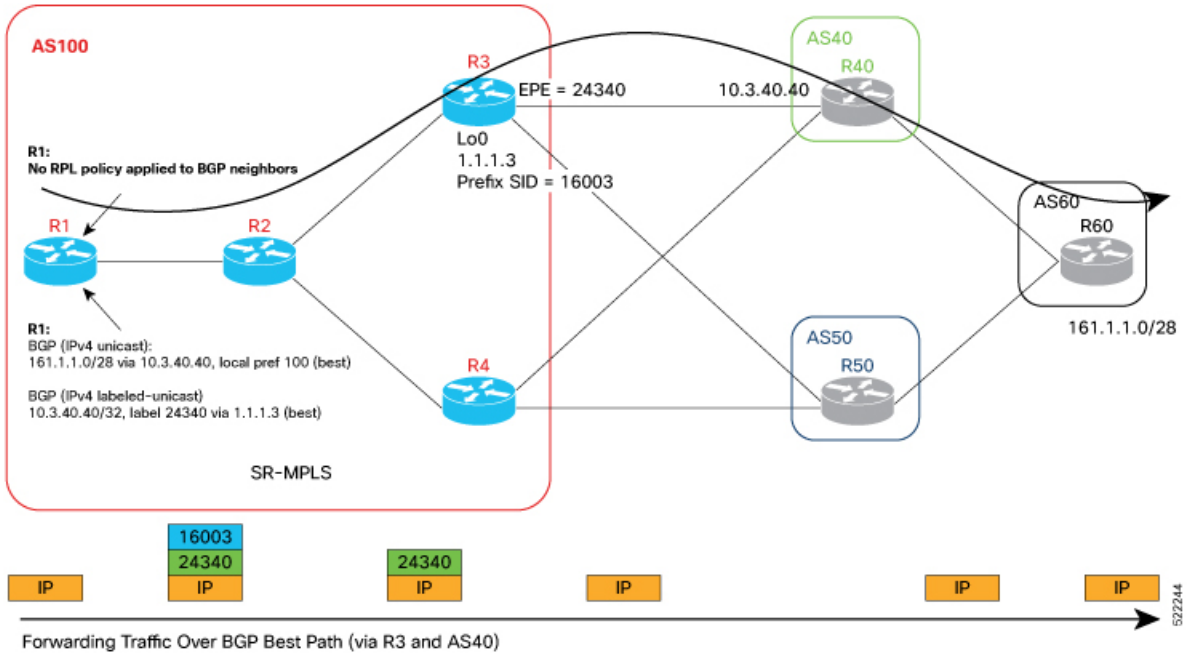
Figure 25: Advertising Overlay Prefixes



NHU = BGP Next-Hop Unchanged

On ingress border router R1, and without any BGP policy modification, assume that BGP selects the path corresponding to AS40 as best path for the overlay prefix 161.1.1.0/28. Its BGP next-hop (10.3.40.40) is learned via BGP-LU from egress border router R3 (1.1.1.3) and with LU label 24340. This label is the EPE label assigned at R3 for the eBGP neighbor to AS40. The EPE local label is programmed as a POP-and-forward toward the interface connecting to AS40. Lastly, R3's loopback (1.1.1.3) and its prefix label (16003) are learned via IS-IS with SR extensions. As a result, incoming traffic matching the 161.1.1.0/28 route is encapsulated at R1 with two MPLS labels (bottom-of-stack label **24340** and top label **16003**) in order to send the traffic to R3 and then to AS40.

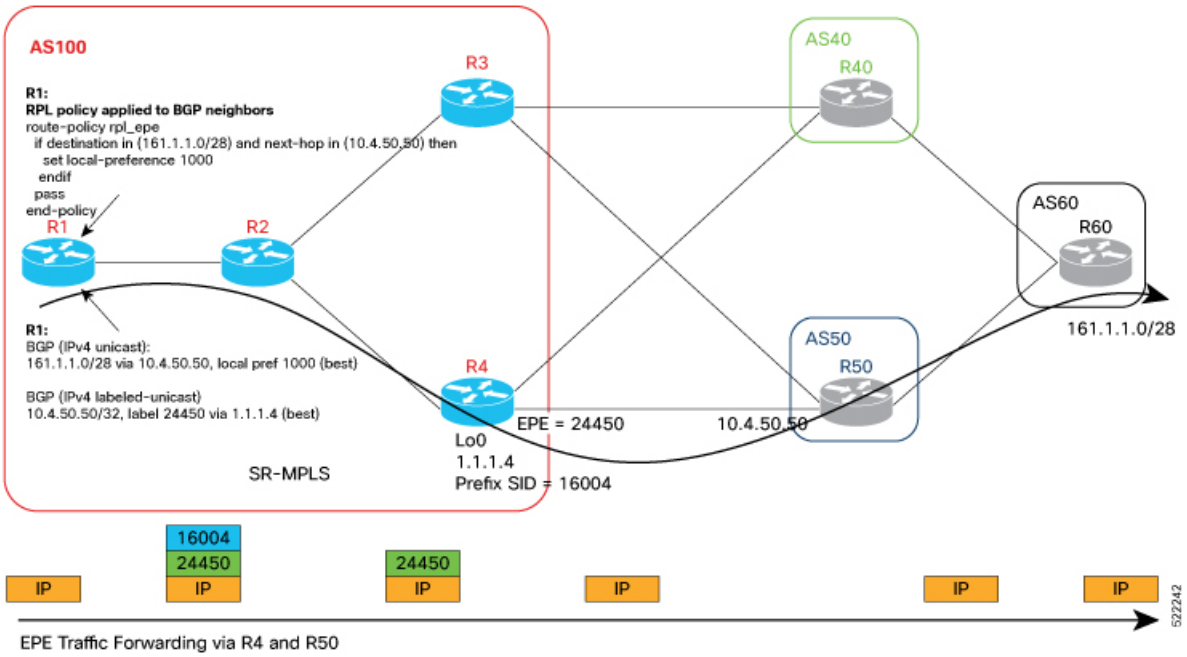
Figure 26: Forwarding Traffic Over BGP Best Path (via R3 and AS40)



Forwarding Traffic Over BGP Best Path (via R3 and AS40)

When the operator wants to modify the exit egress border router and/or an exit AS for a given overlay prefix, a BGP policy can be applied at the ingress border router to influence the best-path selection. In our example, consider that the desired egress path to 161.1.1.0/28 is via R4 and then AS50 (instead of R3 and AS40, as shown in the previous figure). An RPL policy, for example, can be used to assign a higher BGP local preference to the desired path. As a result, incoming traffic matching the 161.1.1.0/28 route is now encapsulated at R1 with two MPLS labels (bottom-of-stack label **24450** and top label **16004**) in order to send the traffic to R4 and then to AS50.

Figure 27: Forwarding Traffic Over EPE Path (via R4 and AS50)



Usage Guidelines and Limitations

The following usage guidelines and limitations apply for this feature:

- BGPv4 and BGPv6 EPE-enabled neighbors are supported.
- BGP peering SIDs (EPE Peer-Node SIDs and Peer-Adjacencies SIDs) allocated dynamically or configured manually can be used as BGP-LU labels when advertising the IP address of an EPE-enabled BGP neighbor via BGP-LU.
- BGP Peer-Set SIDs are not supported.

Enabling Advertisement of EPE-Enabled BGP Neighbors via BGP-LU

To enable advertisement of EPE-enabled BGP neighbors via BGP-LU, use the **advertise epe-bgp labeled-unicast** command in router BGP address family configuration mode.

The following example shows how to enable advertisement of EPE-enabled BGP neighbors via BGP-LU:

```
RP/0/RP0/CPU0:R3(config)# router bgp 100
RP/0/RP0/CPU0:R3(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-bgp-af)# advertise epe-bgp labeled-unicast
```

Running Config

```
router bgp 100
  address-family ipv4 unicast
   advertise epe-bgp labeled-unicast
```

Use Case:

This section provides the router configuration and show command outputs of the scenario described in the overview above

Egress Border Router R3 Configuration

Configure the SRGB:

```
RP/0/RP0/CPU0:R3(config)# segment-routing
RP/0/RP0/CPU0:R3(config-sr)# global-block 16000 23999
RP/0/RP0/CPU0:R3(config-sr)# exit
```

Configure the Loopback address:

```
RP/0/RP0/CPU0:R3(config)# interface Loopback0
RP/0/RP0/CPU0:R3(config-if)# ipv4 address 1.1.1.3 255.255.255.255
RP/0/RP0/CPU0:R3(config-if)# exit
```

Configure MPLS Static on the egress interface connecting to the eBGP peer:

```
RP/0/RP0/CPU0:R3(config)# mpls static
RP/0/RP0/CPU0:R3(config-mpls-static)# interface HundredGigE0/0/0/0
RP/0/RP0/CPU0:R3(config-mpls-static)# exit
```

Enable SR MPLS under IS-IS:

```
RP/0/RP0/CPU0:R3(config)# router isis 1
RP/0/RP0/CPU0:R3(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-isis-af)# segment-routing mpls
RP/0/RP0/CPU0:R3(config-isis-af)# metric-style wide
RP/0/RP0/CPU0:R3(config-isis-af)# exit
```

Configure prefix segment identifier (SID) value on the IS-IS enabled Loopback interface:

```
RP/0/RP0/CPU0:R3(config-isis)# interface Loopback0
RP/0/RP0/CPU0:R3(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-isis-if-af)# prefix-sid absolute 16003
RP/0/RP0/CPU0:R3(config-isis-if-af)# exit
RP/0/RP0/CPU0:R3(config-isis-if)# exit
```

Enable IS-IS in core-facing interface:

```
RP/0/RP0/CPU0:R3(config-isis)# interface HundredGigE0/0/0/0
RP/0/RP0/CPU0:R3(config-isis-if)# point-to-point
RP/0/RP0/CPU0:R3(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-isis-if-af)# exit
RP/0/RP0/CPU0:R3(config-isis-if)# exit
RP/0/RP0/CPU0:R3(config-isis)# exit
RP/0/RP0/CPU0:R3(config)#
```

Configure a route policy to advertise all BGP paths:

```
RP/0/RP0/CPU0:R3(config)# route-policy rpl_advertise_all_paths
RP/0/RP0/CPU0:R3(config-rpl)# set path-selection all advertise
RP/0/RP0/CPU0:R3(config-rpl)# set path-selection backup 1 install multipath-protect
RP/0/RP0/CPU0:R3(config-rpl)# end-policy
```

Enable advertisement of EPE-enabled BGP neighbors via BGP-LU:

```
RP/0/RP0/CPU0:R3(config)# router bgp 100
RP/0/RP0/CPU0:R3(config-bgp)# bgp router-id 1.1.1.3
RP/0/RP0/CPU0:R3(config-bgp)# ibgp policy out enforce-modifications
RP/0/RP0/CPU0:R3(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-bgp-af)# advertise epe-bgp labeled-unicast
RP/0/RP0/CPU0:R3(config-bgp-af)# additional-paths receive
RP/0/RP0/CPU0:R3(config-bgp-af)# additional-paths send
RP/0/RP0/CPU0:R3(config-bgp-af)# additional-paths selection route-policy
rpl_advertise_all_paths
RP/0/RP0/CPU0:R3(config-bgp-af)# allocate-label all
RP/0/RP0/CPU0:R3(config-bgp-af)# exit
```

Enable IPv4 unicast and IPv4 labeled unicast address families on iBGP peer:

```
RP/0/RP0/CPU0:R3(config-bgp)# neighbor 1.1.1.1
RP/0/RP0/CPU0:R3(config-bgp-nbr)# remote-as 100
RP/0/RP0/CPU0:R3(config-bgp-nbr)# update-source Loopback0
RP/0/RP0/CPU0:R3(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# advertise local-labeled-route disable
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R3(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R3(config-bgp-nbr)# exit
```

Enable EPE for the eBGP peers:

```
RP/0/RP0/CPU0:R3(config-bgp)# neighbor 10.3.40.40
RP/0/RP0/CPU0:R3(config-bgp-nbr)# remote-as 40
RP/0/RP0/CPU0:R3(config-bgp-nbr)# egress-engineering
RP/0/RP0/CPU0:R3(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# route-policy pass_all in
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# route-policy pass_all out
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R3(config-bgp-nbr)# exit

RP/0/RP0/CPU0:R3(config-bgp)# neighbor 10.3.50.50
RP/0/RP0/CPU0:R3(config-bgp-nbr)# remote-as 50
RP/0/RP0/CPU0:R3(config-bgp-nbr)# egress-engineering
RP/0/RP0/CPU0:R3(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# route-policy pass_all in
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# route-policy pass_all out
RP/0/RP0/CPU0:R3(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R3(config-bgp-nbr)# exit
RP/0/RP0/CPU0:R3(config-bgp)# exit
RP/0/RP0/CPU0:R3(config)# commit
```

Egress Border Router R3 Running Configuration

```
segment-routing
 global-block 16000 23999
!

interface Loopback0
 ipv4 address 1.1.1.3 255.255.255.255
```

```
mpls static
 interface GigabitEthernet0/0/0/0
!

router isis 1
 is-type level-2-only
 net 47.0000.0000.0003.00
 address-family ipv4 unicast
  metric-style wide
  segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16003
  !
 !
 interface HundredGigE0/0/0/0
  point-to-point
  address-family ipv4 unicast
  !
 !
!

route-policy rpl_advertise_all_paths
  set path-selection all advertise
  set path-selection backup 1 install multipath-protect
end-policy
!

router bgp 100
 bgp router-id 1.1.1.3
 ibgp policy out enforce-modifications
 address-family ipv4 unicast
  advertise epe-bgp labeled-unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy rpl_advertise_all_paths
  allocate-label all
 !
 neighbor 1.1.1.1
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
   advertise local-labeled-route disable
  !
  address-family ipv4 labeled-unicast
  !
 !
 neighbor 10.3.40.40
  remote-as 40
  egress-engineering
  address-family ipv4 unicast
   route-policy pass_all in
   route-policy pass_all out
  !
 !
 neighbor 10.3.50.50
  remote-as 50
  egress-engineering
  address-family ipv4 unicast
   route-policy pass_all in
   route-policy pass_all out
  !
```

```
 !
!
```

Egress Border Router R4 Configuration

The configuration of egress border router R4 follows the configuration of R3:

```
RP/0/RP0/CPU0:R4(config)# segment-routing
RP/0/RP0/CPU0:R4(config-sr)# global-block 16000 23999
RP/0/RP0/CPU0:R4(config-sr)# exit

RP/0/RP0/CPU0:R4(config)# interface Loopback0
RP/0/RP0/CPU0:R4(config-if)# ipv4 address 1.1.1.4 255.255.255.255
RP/0/RP0/CPU0:R4(config-if)# exit

RP/0/RP0/CPU0:R4(config)# mpls static
RP/0/RP0/CPU0:R4(config-mpls-static)# interface HundredGigE0/0/0/0
RP/0/RP0/CPU0:R4(config-mpls-static)# exit

RP/0/RP0/CPU0:R4(config)# router isis 1
RP/0/RP0/CPU0:R4(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-isis-af)# segment-routing mpls
RP/0/RP0/CPU0:R4(config-isis-af)# metric-style wide
RP/0/RP0/CPU0:R4(config-isis-af)# exit

RP/0/RP0/CPU0:R4(config-isis)# interface Loopback0
RP/0/RP0/CPU0:R4(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-isis-if-af)# prefix-sid absolute 16004
RP/0/RP0/CPU0:R4(config-isis-if-af)# exit
RP/0/RP0/CPU0:R4(config-isis-if)# exit

RP/0/RP0/CPU0:R4(config-isis)# interface HundredGigE0/0/0/0
RP/0/RP0/CPU0:R4(config-isis-if)# point-to-point
RP/0/RP0/CPU0:R4(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-isis-if-af)# exit
RP/0/RP0/CPU0:R4(config-isis-if)# exit
RP/0/RP0/CPU0:R4(config-isis)# exit

RP/0/RP0/CPU0:R4(config)# route-policy rpl_advertise_all_paths
RP/0/RP0/CPU0:R4(config-rpl)# set path-selection all advertise
RP/0/RP0/CPU0:R4(config-rpl)# set path-selection backup 1 install multipath-protect
RP/0/RP0/CPU0:R4(config-rpl)# end-policy

RP/0/RP0/CPU0:R4(config)# router bgp 100
RP/0/RP0/CPU0:R4(config-bgp)# bgp router-id 1.1.1.4
RP/0/RP0/CPU0:R4(config-bgp)# ibgp policy out enforce-modifications
RP/0/RP0/CPU0:R4(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-bgp-af)# advertise epe-bgp labeled-unicast
RP/0/RP0/CPU0:R4(config-bgp-af)# additional-paths receive
RP/0/RP0/CPU0:R4(config-bgp-af)# additional-paths send
RP/0/RP0/CPU0:R4(config-bgp-af)# additional-paths selection route-policy
rpl_advertise_all_paths
RP/0/RP0/CPU0:R4(config-bgp-af)# allocate-label all
RP/0/RP0/CPU0:R4(config-bgp-af)# exit

RP/0/RP0/CPU0:R4(config-bgp)# neighbor 1.1.1.1
RP/0/RP0/CPU0:R4(config-bgp-nbr)# remote-as 100
RP/0/RP0/CPU0:R4(config-bgp-nbr)# update-source Loopback0
RP/0/RP0/CPU0:R4(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# advertise local-labeled-route disable
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R4(config-bgp-nbr)# address-family ipv4 labeled-unicast
```

```
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R4(config-bgp-nbr)# exit

RP/0/RP0/CPU0:R4(config-bgp)# neighbor 10.4.40.40
RP/0/RP0/CPU0:R4(config-bgp-nbr)# remote-as 40
RP/0/RP0/CPU0:R4(config-bgp-nbr)# egress-engineering
RP/0/RP0/CPU0:R4(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# route-policy pass_all in
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# route-policy pass_all out
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R4(config-bgp-nbr)# exit

RP/0/RP0/CPU0:R4(config-bgp)# neighbor 10.4.50.50
RP/0/RP0/CPU0:R4(config-bgp-nbr)# remote-as 50
RP/0/RP0/CPU0:R4(config-bgp-nbr)# egress-engineering
RP/0/RP0/CPU0:R4(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# route-policy pass_all in
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# route-policy pass_all out
RP/0/RP0/CPU0:R4(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R4(config-bgp-nbr)# exit
RP/0/RP0/CPU0:R4(config-bgp)# exit
RP/0/RP0/CPU0:R4(config)# commit
```

Egress Border Router R4 Running Configuration

```
segment-routing
 global-block 16000 23999

interface Loopback0
 ipv4 address 1.1.1.4 255.255.255.255

mpls static
 interface GigabitEthernet0/0/0/0
!

router isis 1
 is-type level-2-only
 net 47.0000.0000.0004.00
 address-family ipv4 unicast
  metric-style wide
  segment-routing mpls
 !

 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16004
  !
 !
 interface HundredGigE0/0/0/0
  point-to-point
  address-family ipv4 unicast
  !
 !

route-policy rpl_advertise_all_paths
  set path-selection all advertise
  set path-selection backup 1 install multipath-protect
end-policy
!

router bgp 100
 bgp router-id 1.1.1.4
 ibgp policy out enforce-modifications
```

```
 address-family ipv4 unicast
  advertise epe-bgp labeled-unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy rpl_advertise_all_paths
  allocate-label all
 !
 neighbor 1.1.1.1
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
   advertise local-labeled-route disable
  !
  address-family ipv4 labeled-unicast
  !
 !
 neighbor 10.4.40.40
  remote-as 40
  egress-engineering
  address-family ipv4 unicast
   route-policy pass_all in
   route-policy pass_all out
  !
 !
 neighbor 10.4.50.50
  remote-as 50
  egress-engineering
  address-family ipv4 unicast
   route-policy pass_all in
   route-policy pass_all out
  !
 !
!
```

Ingress Border Router R1 Configuration

Configure the SRGB:

```
RP/0/RP0/CPU0:R1(config)# segment-routing
RP/0/RP0/CPU0:R1(config-sr)# global-block 16000 23999
RP/0/RP0/CPU0:R1(config-sr)# exit
RP/0/RP0/CPU0:R1(config)#
```

Configure the Loopback addresses. Lo0 is advertised in IS-IS and used a BGP next-hop. Lo100 is advertised in BGP as an overlay prefix:

```
RP/0/RP0/CPU0:R1(config)# interface Loopback0
RP/0/RP0/CPU0:R1(config-if)# ipv4 address 1.1.1.1 255.255.255.255
RP/0/RP0/CPU0:R1(config-if)# exit

RP/0/RP0/CPU0:R1(config)# interface Loopback100
RP/0/RP0/CPU0:R1(config-if)# ipv4 address 151.1.1.1 255.255.255.255
RP/0/RP0/CPU0:R1(config-if)# exit
```

Enable SR MPLS under IS-IS:

```
RP/0/RP0/CPU0:R1(config)# router isis 1
RP/0/RP0/CPU0:R1(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-isis-af)# metric-style wide
RP/0/RP0/CPU0:R1(config-isis-af)# segment-routing mpls
RP/0/RP0/CPU0:R1(config-isis-af)# exit
```

Configure prefix segment identifier (SID) value on the IS-IS enabled Loopback interface:

```
RP/0/RP0/CPU0:R1(config-isis)# interface Loopback0 address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-isis-if-af)# prefix-sid absolute 16001
RP/0/RP0/CPU0:R1(config-isis-if-af)# exit
RP/0/RP0/CPU0:R1(config-isis-if)# exit
```

Enable IS-IS in core-facing interface:

```
RP/0/RP0/CPU0:R1(config-isis)# interface HundredGigE0/0/0/0
RP/0/RP0/CPU0:R1(config-isis-if)# point-to-point
RP/0/RP0/CPU0:R1(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-isis-if-af)# exit
RP/0/RP0/CPU0:R1(config-isis-if)# exit
RP/0/RP0/CPU0:R1(config-isis)# exit
```

Configure an RPL policy to prevent allocation of local label to overlay prefixes; such as Lo100 151.1.1.1/32:

```
RP/0/RP0/CPU0:R1(config)# prefix-set unlabelled_prefixes
RP/0/RP0/CPU0:R1(config-pfx)# 151.1.1.1/32
RP/0/RP0/CPU0:R1(config-pfx)# end-set
RP/0/RP0/CPU0:R1(config)# route-policy rpl_allocate_label
RP/0/RP0/CPU0:R1(config-rpl)# if destination in unlabelled_prefixes then
RP/0/RP0/CPU0:R1(config-rpl-if)# drop
RP/0/RP0/CPU0:R1(config-rpl-if)# else
RP/0/RP0/CPU0:R1(config-rpl-else)# pass
RP/0/RP0/CPU0:R1(config-rpl-else)# endif
RP/0/RP0/CPU0:R1(config-rpl)# end-policy
RP/0/RP0/CPU0:R1(config)#
```

Configure an RPL policy to influence the best-path selection by assigning a higher BGP local preference to the desired path. In this example, the desired egress exit path for prefix 161.1.1.0/28 is via R4 and then AS 50, and for prefix 161.1.1.1/32 is via R4 and then AS 40. Otherwise, the uninfluenced exit path for these prefixes is via R3:

```
RP/0/RP0/CPU0:R1(config)# route-policy rpl_epe
RP/0/RP0/CPU0:R1(config-rpl)# if destination in (161.1.1.0/28) and next-hop in (10.4.50.50)
 then
RP/0/RP0/CPU0:R1(config-rpl-if)# set local-preference 1000
RP/0/RP0/CPU0:R1(config-rpl-if)# elseif destination in (161.1.1.1/32) and next-hop in
(10.4.40.40) then
RP/0/RP0/CPU0:R1(config-rpl-elseif)# set local-preference 1000
RP/0/RP0/CPU0:R1(config-rpl-elseif)# endif
RP/0/RP0/CPU0:R1(config-rpl)# pass
RP/0/RP0/CPU0:R1(config-rpl)# end-policy
RP/0/RP0/CPU0:R1(config)#
```

Configure an RPL policy to advertise all candidate paths:

```
RP/0/RP0/CPU0:R1(config)# route-policy rpl_advertise_all_paths
RP/0/RP0/CPU0:R1(config-rpl)# set path-selection all advertise
RP/0/RP0/CPU0:R1(config-rpl)# end-policy


RP/0/RP0/CPU0:R1(config)# router bgp 100
RP/0/RP0/CPU0:R1(config-bgp)# bgp router-id 1.1.1.1
RP/0/RP0/CPU0:R1(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-bgp-af)# additional-paths receive
RP/0/RP0/CPU0:R1(config-bgp-af)# additional-paths send
RP/0/RP0/CPU0:R1(config-bgp-af)# additional-paths selection route-policy
rpl_advertise_all_paths
```

```
RP/0/RP0/CPU0:R1(config-bgp-af)# exit


RP/0/RP0/CPU0:R1(config-bgp)# neighbor 1.1.1.3
RP/0/RP0/CPU0:R1(config-bgp-nbr)# remote-as 100
RP/0/RP0/CPU0:R1(config-bgp-nbr)# update-source Loopback0
RP/0/RP0/CPU0:R1(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# advertise local-labeled-route disable
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R1(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R1(config-bgp-nbr)# exit

RP/0/RP0/CPU0:R1(config-bgp)# neighbor 1.1.1.4
RP/0/RP0/CPU0:R1(config-bgp-nbr)# remote-as 100
RP/0/RP0/CPU0:R1(config-bgp-nbr)# update-source Loopback0
RP/0/RP0/CPU0:R1(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# advertise local-labeled-route disable
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R1(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R1(config-bgp-nbr)# exit
RP/0/RP0/CPU0:R1(config-bgp)# exit
RP/0/RP0/CPU0:R1(config)# commit
```

Ingress Border Router R1 Running Configuration

```
segment-routing
 global-block 16000 23999

interface Loopback0
 ipv4 address 1.1.1.1 255.255.255.255
!
interface Loopback100
 ipv4 address 151.1.1.1 255.255.255.255

router isis 1
 is-type level-2-only
 net 47.0000.0000.0001.00
 address-family ipv4 unicast
  metric-style wide
  segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16001
  !
 !
 interface HundredGigE0/0/0/0
  point-to-point
  address-family ipv4 unicast
  !
 !
!

prefix-set unlabelled_prefixes
  151.1.1.1/32
end-set
!

route-policy rpl_allocate_label
  if destination in unlabelled_prefixes then
    drop
```

```
  else
    pass
  endif
end-policy
!

route-policy rpl_epe
  if destination in (161.1.1.0/28) and next-hop in (10.4.50.50) then
    set local-preference 1000
  elseif destination in (161.1.1.1/32) and next-hop in (10.4.40.40) then
    set local-preference 1000
  endif
  pass
end-policy
!

route-policy rpl_advertise_all_paths
  set path-selection all advertise
end-policy
!

router bgp 100
 bgp router-id 1.1.1.1
 ibgp policy out enforce-modifications
 address-family ipv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy rpl_advertise_all_paths
  network 151.1.1.1/32
  allocate-label route-policy rpl_allocate_label
 !
 neighbor 1.1.1.3
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
   advertise local-labeled-route disable
  !
  address-family ipv4 labeled-unicast
  !
 !
 neighbor 1.1.1.4
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
   advertise local-labeled-route disable
  !
  address-family ipv4 labeled-unicast
  !
 !
!
```

The following sections depict the **show** command outputs associated with the Egress Border routers (R3, R4) and Ingress PE router (R1):

Egress Border Router R3 Output

The following commands show the BGP EPE labels allocated for eBGP neighbors 10.3.40.40 and 10.3.50.50 alongside their corresponding entries in the FIB:

```
RP/0/RP0/CPU0:R3# show bgp egress-engineering

 Egress Engineering Object: 10.3.40.40/32 (0x7fc163c62e80)
       EPE Type: Peer
```

```
       Nexthop: 10.3.40.40
       Version: 2, rn_version: 2
         Flags: 0x00000006
    Local ASN: 100
   Remote ASN: 40
    Local RID: 1.1.1.3
   Remote RID: 1.1.1.40
 Local Address: 10.3.40.3
    First Hop: 10.3.40.40
         NHID: 0
          IFH: 0x198
        Label: 24004, Refcount: 4
      rpc_set: 0x7fc14410ff18, ID: 1

Egress Engineering Object: 10.3.50.50/32 (0x7fc163c62d88)
     EPE Type: Peer
       Nexthop: 10.3.50.50
       Version: 3, rn_version: 3
         Flags: 0x00000006
    Local ASN: 100
   Remote ASN: 50
    Local RID: 1.1.1.3
   Remote RID: 1.1.1.50
 Local Address: 10.3.50.3
    First Hop: 10.3.50.50
         NHID: 0
          IFH: 0x1a0
        Label: 24005, Refcount: 4
      rpc_set: 0x7fc144110088, ID: 2
```

```
RP/0/RP0/CPU0:R3# show mpls forwarding labels 24004
Thu Feb  3 22:11:18.459 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24004  Pop         No ID              Hu0/0/0/1    10.3.40.40      0

RP/0/RP0/CPU0:R3# show mpls forwarding labels 24005
Thu Feb  3 22:11:35.399 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24005  Pop         No ID              Hu0/0/0/2    10.3.50.50      0
```

The following output displays the BGP-LU prefixes used to advertise the EPE-enabled eBGP neighbors 10.3.40.40 and 10.3.50.50:

```
RP/0/RP0/CPU0:R3# show bgp ipv4 labeled-unicast
Thu Feb  3 22:11:57.865 UTC
BGP router identifier 1.1.1.3, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network            Next Hop            Metric LocPrf Weight Path
*> 10.3.40.40/32      0.0.0.0                              0 i
*> 10.3.50.50/32      0.0.0.0                              0 i

Processed 2 prefixes, 2 paths
```

The details of the BGP-LU prefixes can be found below. Note that the EPE label is advertised in BGP-LU.

```
RP/0/RP0/CPU0:R3# show bgp ipv4 labeled-unicast 10.3.40.40/32
Thu Feb  3 22:12:18.210 UTC
BGP routing table entry for 10.3.40.40/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 3           3
    Local Label: 24004
Last Modified: Feb  3 19:13:07.039 for 02:59:11
Paths: (1 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Local
    0.0.0.0 from 0.0.0.0 (1.1.1.3)
      Origin IGP, localpref 100, valid, extranet, best, group-best
      Received Path ID 0, Local Path ID 1, version 3
      Origin-AS validity: not-found


RP/0/RP0/CPU0:R3# show bgp ipv4 labeled-unicast 10.3.50.50/32
Thu Feb  3 22:12:27.282 UTC
BGP routing table entry for 10.3.50.50/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 4           4
    Local Label: 24005
Last Modified: Feb  3 19:13:07.039 for 02:59:20
Paths: (1 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Local
    0.0.0.0 from 0.0.0.0 (1.1.1.3)
      Origin IGP, localpref 100, valid, extranet, best, group-best
      Received Path ID 0, Local Path ID 1, version 4
      Origin-AS validity: not-found
```

The output below depicts the BGP route and CEF details for an overlay prefix (161.1.1.0/28) learned via the EPE-enabled BGP neighbors:

```
RP/0/RP0/CPU0:R3# show bgp ipv4 unicast
Thu Feb  3 22:58:01.736 UTC
BGP router identifier 1.1.1.3, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 14
BGP main routing table version 14
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs
```

```
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 10.3.40.40/32      0.0.0.0                                0 i
*> 10.3.50.50/32      0.0.0.0                                0 i
*>i151.1.1.1/32       1.1.1.1                 0    100      0 i
*> 161.1.1.0/28       10.3.40.40                             0 40 60 i
*                     10.3.50.50                             0 50 60 i

Processed 4 prefixes, 5 paths


RP/0/RP0/CPU0:R3# show bgp ipv4 unicast 161.1.1.0/28
Thu Feb  3 22:31:52.893 UTC
BGP routing table entry for 161.1.1.0/28
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 6           6
Last Modified: Feb  3 22:28:56.039 for 00:02:56
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  40 60
    10.3.40.40 from 10.3.40.40 (1.1.1.40)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 5
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  50 60
    10.3.50.50 from 10.3.50.50 (1.1.1.50)
      Origin IGP, localpref 100, valid, external, group-best, backup, add-path
      Received Path ID 0, Local Path ID 2, version 6
      Origin-AS validity: (disabled)


RP/0/RP0/CPU0:R3# show cef ipv4 161.1.1.0/28
Thu Feb 10 20:17:29.240 UTC
161.1.1.0/28, version 24, internal 0x5000001 0x40 (ptr 0x90684920) [1], 0x0 (0x0), 0x0 (0x0)

 Updated Feb 10 17:37:16.609
 Prefix Len 28, traffic index 0, precedence n/a, priority 4
   via 10.3.40.40/32, 5 dependencies, recursive, bgp-ext [flags 0x6020]
    path-idx 0 NHID 0x0 [0x90684c08 0x0], Internal 0x90211730
    next hop 10.3.40.40/32 via 10.3.40.40/32
   via 10.3.50.50/32, 4 dependencies, recursive, bgp-ext, backup [flags 0x6120]
    path-idx 1 NHID 0x0 [0x90685040 0x0]
    next hop 10.3.50.50/32 via 10.3.50.50/32
```

Egress Border Router R4 Output

The following outputs correspond to egress border router R4. They follow the same sequence shown for router R3.

```
RP/0/RP0/CPU0:R4# show bgp egress-engineering

Egress Engineering Object: 10.4.40.40/32 (0x7f84d2a4ae80)
       EPE Type: Peer
        Nexthop: 10.4.40.40
        Version: 2, rn_version: 2
          Flags: 0x00000006
      Local ASN: 100
     Remote ASN: 40
      Local RID: 1.1.1.4
     Remote RID: 1.1.1.40
  Local Address: 10.4.40.4
      First Hop: 10.4.40.40
           NHID: 0
            IFH: 0x198
          Label: 24004, Refcount: 4
        rpc_set: 0x7f84b010fdb8, ID: 1

 Egress Engineering Object: 10.4.50.50/32 (0x7f84d2a4ad88)
       EPE Type: Peer
        Nexthop: 10.4.50.50
        Version: 3, rn_version: 3
          Flags: 0x00000006
      Local ASN: 100
     Remote ASN: 50
      Local RID: 1.1.1.4
     Remote RID: 1.1.1.50
  Local Address: 10.4.50.4
      First Hop: 10.4.50.50
           NHID: 0
            IFH: 0x1a0
          Label: 24005, Refcount: 4
        rpc_set: 0x7f84b010ff28, ID: 2


RP/0/RP0/CPU0:R4# show mpls forwarding labels 24004
Thu Feb  3 22:34:55.059 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24004  Pop         No ID              Hu0/0/0/1    10.4.40.40      0


RP/0/RP0/CPU0:R4# show mpls forwarding labels 24005
Thu Feb  3 22:35:07.252 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24005  Pop         No ID              Hu0/0/0/2    10.4.50.50      0


RP/0/RP0/CPU0:R4# show bgp ipv4 labeled-unicast
Thu Feb  3 22:59:37.978 UTC
BGP router identifier 1.1.1.4, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 14
BGP main routing table version 14
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 10.4.40.40/32      0.0.0.0                                0 i
*> 10.4.50.50/32      0.0.0.0                                0 i

Processed 2 prefixes, 2 paths


RP/0/RP0/CPU0:R4# show bgp ipv4 labeled-unicast 10.4.40.40/32
Thu Feb  3 22:35:41.275 UTC
BGP routing table entry for 10.4.40.40/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 3           3
    Local Label: 24004
Last Modified: Feb  3 19:13:08.143 for 03:22:33
Paths: (1 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Labeled-unicast paths to peers (in unique update groups):
    1.1.1.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Labeled-unicast paths to peers (in unique update groups):
    1.1.1.1
  Local
    0.0.0.0 from 0.0.0.0 (1.1.1.4)
      Origin IGP, localpref 100, valid, extranet, best, group-best
      Received Path ID 0, Local Path ID 1, version 3
      Origin-AS validity: not-found


RP/0/RP0/CPU0:R4# show bgp ipv4 labeled-unicast 10.4.50.50/32
Thu Feb  3 22:35:53.259 UTC
BGP routing table entry for 10.4.50.50/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 4           4
    Local Label: 24005
Last Modified: Feb  3 19:13:08.143 for 03:22:45
Paths: (1 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Labeled-unicast paths to peers (in unique update groups):
    1.1.1.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Labeled-unicast paths to peers (in unique update groups):
    1.1.1.1
  Local
    0.0.0.0 from 0.0.0.0 (1.1.1.4)
      Origin IGP, localpref 100, valid, extranet, best, group-best
      Received Path ID 0, Local Path ID 1, version 4
      Origin-AS validity: not-found


RP/0/RP0/CPU0:R4# show bgp ipv4 unicast
Thu Feb  3 23:00:32.470 UTC
BGP router identifier 1.1.1.4, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 14
```

```
BGP main routing table version 14
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*> 10.4.40.40/32      0.0.0.0                                0 i
*> 10.4.50.50/32      0.0.0.0                                0 i
*>i151.1.1.1/32       1.1.1.1                  0    100      0 i
*> 161.1.1.0/28       10.4.40.40                             0 40 60 i
*                     10.4.50.50                             0 50 60 i

Processed 4 prefixes, 5 paths


RP/0/RP0/CPU0:R4# show bgp ipv4 unicast 161.1.1.0/28
Thu Feb  3 22:36:09.266 UTC
BGP routing table entry for 161.1.1.0/28
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 6           6
Last Modified: Feb  3 22:28:56.143 for 00:07:13
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  40 60
    10.4.40.40 from 10.4.40.40 (1.1.1.40)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 5
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  50 60
    10.4.50.50 from 10.4.50.50 (1.1.1.50)
      Origin IGP, localpref 100, valid, external, group-best, backup, add-path
      Received Path ID 0, Local Path ID 2, version 6
      Origin-AS validity: (disabled)
```

Ingress Border Router R1 Output

This section includes the outputs corresponding to ingress border router R1.

R1 learns the eBGP neighbor IP addresses via BGP-LU. In the details for each neighbor prefix, observe that the advertised BGP-LU label corresponds to the EPE label at the egress border router (R3 or R4).

```
RP/0/RP0/CPU0:R1# show bgp ipv4 labeled-unicast
Thu Feb 10 20:18:59.645 UTC
BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 8
BGP main routing table version 8
```

```
BGP NSR Initial initsync version 5 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*>i10.3.40.40/32      1.1.1.3                  100      0 i
*>i10.3.50.50/32      1.1.1.3                  100      0 i
*>i10.4.40.40/32      1.1.1.4                  100      0 i
*>i10.4.50.50/32      1.1.1.4                  100      0 i

Processed 4 prefixes, 4 paths


RP/0/RP0/CPU0:R1# show bgp ipv4 labeled-unicast 10.3.40.40/32
Thu Feb  3 23:01:57.912 UTC
BGP routing table entry for 10.3.40.40/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                15          15
    Local Label: 24004
Last Modified: Feb  3 22:47:43.539 for 00:14:14
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    1.1.1.3 (metric 30) from 1.1.1.3 (1.1.1.3)
      Received Label 24004
      Origin IGP, localpref 100, valid, internal, best, group-best, labeled-unicast
      Received Path ID 1, Local Path ID 1, version 15


RP/0/RP0/CPU0:R1# show bgp ipv4 labeled-unicast 10.3.50.50/32
Thu Feb  3 23:02:09.173 UTC
BGP routing table entry for 10.3.50.50/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                16          16
    Local Label: 24005
Last Modified: Feb  3 22:47:43.539 for 00:14:25
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    1.1.1.3 (metric 30) from 1.1.1.3 (1.1.1.3)
      Received Label 24005
      Origin IGP, localpref 100, valid, internal, best, group-best, labeled-unicast
      Received Path ID 1, Local Path ID 1, version 16


RP/0/RP0/CPU0:R1# show bgp ipv4 labeled-unicast 10.4.40.40/32
Thu Feb  3 23:02:18.843 UTC
BGP routing table entry for 10.4.40.40/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                17          17
    Local Label: 24006
Last Modified: Feb  3 22:47:43.539 for 00:14:35
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
```

```
  Not advertised to any peer
  Local
    1.1.1.4 (metric 30) from 1.1.1.4 (1.1.1.4)
      Received Label 24004
      Origin IGP, localpref 100, valid, internal, best, group-best, labeled-unicast
      Received Path ID 1, Local Path ID 1, version 17


RP/0/RP0/CPU0:R1# show bgp ipv4 labeled-unicast 10.4.50.50/32
Thu Feb  3 23:02:27.622 UTC
BGP routing table entry for 10.4.50.50/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 18          18
    Local Label: 24007
Last Modified: Feb  3 22:47:43.539 for 00:14:44
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    1.1.1.4 (metric 30) from 1.1.1.4 (1.1.1.4)
      Received Label 24005
      Origin IGP, localpref 100, valid, internal, best, group-best, labeled-unicast
      Received Path ID 1, Local Path ID 1, version 18


RP/0/RP0/CPU0:R1# show isis segment-routing label table

IS-IS 1 IS Label Table
Label         Prefix                 Interface
----------    ----------------       ---------
16001         1.1.1.1/32             Loopback0
16002         1.1.1.2/32
16003         1.1.1.3/32
16004         1.1.1.4/32
```

The following show commands depict the RIB and CEF outputs for the loopbacks of R3 and R4 learned vis ISIS-SR:

```
RP/0/RP0/CPU0:R1# show route 1.1.1.3/32

Routing entry for 1.1.1.3/32
  Known via "isis 1", distance 115, metric 30, labeled SR, type level-2
  Installed Feb 10 17:36:12.497 for 02:43:40
  Routing Descriptor Blocks
    10.1.2.2, from 1.1.1.3, via HundredGigE0/0/0/0
      Route metric is 30
  No advertising protos.


RP/0/RP0/CPU0:R1# show route 1.1.1.4/32

Routing entry for 1.1.1.4/32
  Known via "isis 1", distance 115, metric 30, labeled SR, type level-2
  Installed Feb 10 17:37:02.171 for 02:42:59
  Routing Descriptor Blocks
    10.1.2.2, from 1.1.1.4, via HundredGigE0/0/0/0
      Route metric is 30
  No advertising protos.


RP/0/RP0/CPU0:R1# show cef 1.1.1.3/32
```

```
1.1.1.3/32, version 18, labeled SR, internal 0x1000001 0x8110 (ptr 0x90cd33a0) [1], 0x0
(0x90c3eb10), 0xa28 (0x91a18378)
 Updated Feb 10 17:36:12.506
 local adjacency to HundredGigE0/0/0/0

 Prefix Len 32, traffic index 0, precedence n/a, priority 1
   via 10.1.2.2/32, HundredGigE0/0/0/0, 7 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x91de84d8 0x0]
    next hop 10.1.2.2/32
    local adjacency
     local label 16003      labels imposed {16003}


RP/0/RP0/CPU0:R1# show cef 1.1.1.4/32

1.1.1.4/32, version 20, labeled SR, internal 0x1000001 0x8110 (ptr 0x90cd32c8) [1], 0x0
(0x90c3eb58), 0xa28 (0x91a18408)
 Updated Feb 10 17:37:02.176
 local adjacency to HundredGigE0/0/0/0

 Prefix Len 32, traffic index 0, precedence n/a, priority 1
   via 10.1.2.2/32, HundredGigE0/0/0/0, 7 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x91de84d8 0x0]
    next hop 10.1.2.2/32
    local adjacency
     local label 16004      labels imposed {16004}
```

Next, we observe the BGP table for overlay prefixes at R1. In this usecase, we use prefix 161.1.1.0/28 as an overlay prefix learned from AS 40 and AS 50. Note that all BGP paths are present at R1 with a BGP next-hop unchanged. By default and without any BGP policy applied, the BGP best-path is the path from NH 10.3.40.40 (AS 40 via R3).

```
RP/0/RP0/CPU0:R1# show bgp

BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 21
BGP main routing table version 21
BGP NSR Initial initsync version 7 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*>i10.3.40.40/32      1.1.1.3                  100      0 i
*>i10.3.50.50/32      1.1.1.3                  100      0 i
*>i10.4.40.40/32      1.1.1.4                  100      0 i
*>i10.4.50.50/32      1.1.1.4                  100      0 i
*> 151.1.1.1/32       0.0.0.0                  0         32768 i
*>i161.1.1.0/28       10.3.40.40               100      0 40 60 i
* i                   10.3.50.50               100      0 50 60 i
* i                   10.4.40.40               100      0 40 60 i
* i                   10.4.50.50               100      0 50 60 i

Processed 6 prefixes, 9 paths
```

```
RP/0/RP0/CPU0:R1# show bgp ipv4 unicast 161.1.1.0/28

BGP routing table entry for 161.1.1.0/28
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 8           8
Last Modified: Feb 10 17:38:09.280 for 02:42:57
Paths: (4 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  40 60
    10.3.40.40 (metric 30) from 1.1.1.3 (1.1.1.3)
      Origin IGP, localpref 100, valid, internal, best, group-best
      Received Path ID 1, Local Path ID 1, version 8
  Path #2: Received by speaker 0
  Not advertised to any peer
  50 60
    10.3.50.50 (metric 30) from 1.1.1.3 (1.1.1.3)
      Origin IGP, localpref 100, valid, internal, group-best, add-path
      Received Path ID 2, Local Path ID 4, version 8
  Path #3: Received by speaker 0
  Not advertised to any peer
  40 60
    10.4.40.40 (metric 30) from 1.1.1.4 (1.1.1.4)
      Origin IGP, localpref 100, valid, internal, add-path
      Received Path ID 1, Local Path ID 2, version 8
  Path #4: Received by speaker 0
  Not advertised to any peer
  50 60
    10.4.50.50 (metric 30) from 1.1.1.4 (1.1.1.4)
      Origin IGP, localpref 100, valid, internal, add-path
      Received Path ID 2, Local Path ID 3, version 8
```

A ping and traceroute to the overlay prefix confirms that the traffic is directed to R3 (prefix SID 16003) and then to AS 40 (EPE label 24004 for the eBGP neighbor to AS 40 at R3).

```
RP/0/RP0/CPU0:R1# ping 161.1.1.1 source 151.1.1.1 count 10
Thu Feb  3 23:20:48.911 UTC
Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 161.1.1.1, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 30/36/54 ms


RP/0/RP0/CPU0:R1# traceroute 161.1.1.1 source 151.1.1.1
Thu Feb  3 23:20:53.630 UTC

Type escape sequence to abort.
Tracing the route to 161.1.1.1

 1  10.1.2.2 [MPLS: Labels 16003/24004 Exp 0] 49 msec  45 msec  42 msec
 2  10.2.3.3 [MPLS: Label 24004 Exp 0] 42 msec  37 msec  37 msec
 3  10.3.40.40 44 msec  37 msec  41 msec
 4  10.40.60.60 47 msec *  55 msec
```

Now, we proceed to apply a BGP route-policy that would modify BGP best-path selection and choose instead the path from NH 10.4.50.50 (AS 50 via R4).

```
RP/0/RP0/CPU0:R1(config)# route-policy rpl_epe
RP/0/RP0/CPU0:R1(config-rpl)# if destination in (161.1.1.0/28) and next-hop in (10.4.50.50)
 then
RP/0/RP0/CPU0:R1(config-rpl-if)# set local-preference 1000
```

```
RP/0/RP0/CPU0:R1(config-rpl-if)# elseif destination in (161.1.1.1/32) and next-hop in
(10.4.40.40) then
RP/0/RP0/CPU0:R1(config-rpl-elseif)# set local-preference 1000
RP/0/RP0/CPU0:R1(config-rpl-elseif)# endif
RP/0/RP0/CPU0:R1(config-rpl)# pass
RP/0/RP0/CPU0:R1(config-rpl)# end-policy
RP/0/RP0/CPU0:R1(config)#


RP/0/RP0/CPU0:R1(config)# router bgp 100
RP/0/RP0/CPU0:R1(config-bgp)# neighbor 1.1.1.3
RP/0/RP0/CPU0:R1(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# route-policy rpl_epe in
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:R1(config-bgp-nbr)# exit
RP/0/RP0/CPU0:R1(config-bgp)# neighbor 1.1.1.4
RP/0/RP0/CPU0:R1(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)# route-policy rpl_epe in
RP/0/RP0/CPU0:R1(config-bgp-nbr-af)#
```

Observe the new BGP best-path selected for the overlay prefix via NH 10.4.50.50:

```
RP/0/RP0/CPU0:R1# show bgp

BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 20
BGP main routing table version 20
BGP NSR Initial initsync version 7 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*>i10.3.40.40/32      1.1.1.3                  100      0 i
*>i10.3.50.50/32      1.1.1.3                  100      0 i
*>i10.4.40.40/32      1.1.1.4                  100      0 i
*>i10.4.50.50/32      1.1.1.4                  100      0 i
*> 151.1.1.1/32       0.0.0.0                  0         32768 i
* i161.1.1.0/28       10.3.40.40               100      0 40 60 i
* i                   10.3.50.50               100      0 50 60 i
* i                   10.4.40.40               100      0 40 60 i
*>i                   10.4.50.50              1000      0 50 60 i

Processed 6 prefixes, 9 paths


RP/0/RP0/CPU0:R1# show bgp ipv4 unicast 161.1.1.0/28

BGP routing table entry for 161.1.1.0/28
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                20          20
Last Modified: Feb  3 23:13:30.539 for 00:01:33
Paths: (4 available, best #4)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  40 60
    10.3.40.40 (metric 30) from 1.1.1.3 (1.1.1.3)
```

```
    Origin IGP, localpref 100, valid, internal, group-best, add-path
    Received Path ID 1, Local Path ID 4, version 20
Path #2: Received by speaker 0
Not advertised to any peer
50 60
  10.3.50.50 (metric 30) from 1.1.1.3 (1.1.1.3)
    Origin IGP, localpref 100, valid, internal, add-path
    Received Path ID 2, Local Path ID 3, version 8
Path #3: Received by speaker 0
Not advertised to any peer
40 60
  10.4.40.40 (metric 30) from 1.1.1.4 (1.1.1.4)
    Origin IGP, localpref 100, valid, internal, add-path
    Received Path ID 1, Local Path ID 2, version 8
Path #4: Received by speaker 0
Not advertised to any peer
50 60
  10.4.50.50 (metric 30) from 1.1.1.4 (1.1.1.4)
    Origin IGP, localpref 1000, valid, internal, best, group-best
    Received Path ID 2, Local Path ID 1, version 20
```

A ping and traceroute to the overlay prefix confirms that after the RPL policy is applied, the traffic is directed instead to R4 (prefix SID 16004) and then to AS 50 (EPE label 24005 for the eBGP neighbor to AS 50 at R4).

```
RP/0/RP0/CPU0:R1# ping 161.1.1.1 source 151.1.1.1 count 10
Thu Feb  3 23:17:43.812 UTC
Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 161.1.1.1, timeout is 2 seconds:
!!!!!!!!!!
Success rate is 100 percent (10/10), round-trip min/avg/max = 30/35/50 ms


RP/0/RP0/CPU0:R1# traceroute 161.1.1.1 source 151.1.1.1
Thu Feb  3 23:18:01.656 UTC

Type escape sequence to abort.
Tracing the route to 161.1.1.1

 1  10.1.2.2 [MPLS: Labels 16004/24005 Exp 0] 50 msec  42 msec  45 msec
 2  10.2.4.4 [MPLS: Label 24005 Exp 0] 50 msec  42 msec  42 msec
 3  10.4.50.50 46 msec  44 msec  44 msec
 4  10.50.60.60 51 msec  *  54 msec
```

IP Lookup Fallback for BGP Peering (EPE) Segments

Table 35: Feature History Table

Feature Name	Release	Description
IP Lookup Fallback for BGP Peering (EPE) Segments	Release 7.3.3	BGP peering segments/SIDs are part of the Segment Routing Centralized BGP Egress Peer Engineering solution (BGP-EPE). A BGP-EPE-enabled border router allocates and programs BGP peering SIDs (EPE labels) to steer traffic over a specific external interface/BGP neighbor. This feature allows a BGP-EPE-enabled border router to pop the EPE label and forward traffic based on an IP-based lookup when a BGP neighbor fails. Traffic arriving with the EPE label assigned to a failed neighbor is forwarded based on a destination IP address lookup to allow traffic to be forwarded over a different directly connected external peer.

BGP peering segments/SIDs are part of the Segment Routing Centralized BGP Egress Peer Engineering solution (BGP-EPE), as described in IETF RFC 9087. A BGP-EPE-enabled border router allocates and programs BGP peering SIDs (EPE labels) to steer traffic over a specific external interface/BGP neighbor.

This feature allows a BGP-EPE-enabled border router to pop the EPE label and forward traffic based on an IP-based lookup when a BGP neighbor fails. Traffic arriving with the EPE label assigned to a failed neighbor is forwarded based on a destination IP address lookup to allow traffic to be forwarded over a different directly connected external peer.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply for this feature:

- IP Lookup Fallback for BGP peering SIDs (EPE Peer-Node SIDs and Peer-Adjacencies SIDs) allocated dynamically or configured manually is supported.
- BGPv4 and BGPv6 EPE-enabled neighbors are supported
- Sub-second convergence is supported upon failure of EPE-enabled BGP neighbor with interface peering.
- Sub-second convergence is supported upon failure of EPE-enabled BGP neighbor with loopback peering over a single interface.
- Sub-second convergence is not supported upon failure of EPE-enabled BGP neighbor with loopback peering over more than one interface.
- IP Lookup Fallback for BGP Peer-Set SIDs is not supported

- MPLS egress path counters for BGP peering SIDs are not supported when IP Lookup Fallback is enabled

Enabling IP Lookup Fallback for BGP Peering (EPE) Segments

To guaranteed convergence, configure a route policy on the ingress border router to advertise all BGP paths. For example:

```
RP/0/RP0/CPU0:R1(config)# route-policy INSTALL_BACKUP
RP/0/RP0/CPU0:R1(config-rpl)# set path-selection all advertise
RP/0/RP0/CPU0:R1(config-rpl)# set path-selection backup 1 install multipath-protect
RP/0/RP0/CPU0:R1(config-rpl)# end-policy

RP/0/RP0/CPU0:R1(config)# router bgp 100
RP/0/RP0/CPU0:R1(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:R1(config-bgp-af)# additional-paths selection route-policy INSTALL_BACKUP
RP/0/RP0/CPU0:R1(config-bgp-af)# exit
RP/0/RP0/CPU0:R1(config-bgp)# exit
RP/0/RP0/CPU0:R1(config)#
```

To enable IP lookup fallback for EPE segments, use the **epe backup enable** command in router BGP address family configuration mode.

To retain the local label of the primary path after reconvergence for the specified amount of time, use the **retain local-label** *minutes* command in router BGP address family configuration mode. The range of *minutes* is from 3 to 60.

The following example shows how to enable IP lookup fallback for EPE segments associated with BGPv4 EPE-enabled neighbors:

```
RP/0/RP0/CPU0:R3(config)# router bgp 100
RP/0/RP0/CPU0:R3(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:R3(config-bgp-af)# epe backup enable
RP/0/RP0/CPU0:R3(config-bgp-af)# retain local-label 6
```

Running Config

```
router bgp 100
  address-family ipv4 unicast
   epe backup enable
   retain local-label 6
```

Verification

The following outputs display the forwarding entries for the EPE MPLS labels (24004 and 24005) at an egress border router **before** the IP Lookup Fallback for EPE feature is enabled. Observe that no backup is programmed.

```
RP/0/RP0/CPU0:R3# show mpls forwarding labels 24004 24005

Local  Outgoing    Prefix            Outgoing     Next Hop        Bytes
Label  Label       or ID             Interface                    Switched
------ ----------- ----------------- ------------ --------------- ------------
24004  Pop         No ID             Hu0/0/0/1    10.3.40.40      0
24005  Pop         No ID             Hu0/0/0/2    10.3.50.50      0

RP/0/RP0/CPU0:R3# show mpls forwarding labels 24004 24005 detail
```

```
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24004  Pop         No ID              Hu0/0/0/1    10.3.40.40      0
     Updated: Feb 10 18:38:16.116
     Path Flags: 0x6000 [  ]
     Version: 16, Priority: 3
     Label Stack (Top -> Bottom): { Imp-Null }
     NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 0, Weight: 0
     MAC/Encaps: 0/0, MTU: 1500
     Outgoing Interface: HundredGigE0/0/0/1 (ifhandle 0x00000198)
     Packets Switched: 0

24005  Pop         No ID              Hu0/0/0/2    10.3.50.50      0
     Updated: Feb 10 18:38:16.116
     Path Flags: 0x6000 [  ]
     Version: 17, Priority: 3
     Label Stack (Top -> Bottom): { Imp-Null }
     NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 0, Weight: 0
     MAC/Encaps: 0/0, MTU: 1500
     Outgoing Interface: HundredGigE0/0/0/2 (ifhandle 0x000001a0)
     Packets Switched: 0
```

The following output depicts the BGP table for an overlay prefix including its primary and backup path.

```
RP/0/RP0/CPU0:R3# show bgp ipv4 unicast 161.1.1.0/28

BGP routing table entry for 161.1.1.0/28
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 6           6
Last Modified: Feb  3 22:28:56.039 for 00:02:56
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  40 60
    10.3.40.40 from 10.3.40.40 (1.1.1.40)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 5
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  50 60
    10.3.50.50 from 10.3.50.50 (1.1.1.50)
      Origin IGP, localpref 100, valid, external, group-best, backup, add-path
      Received Path ID 0, Local Path ID 2, version 6
      Origin-AS validity: (disabled)


RP/0/RP0/CPU0:R3# show cef 161.1.1.0/28 detail

161.1.1.0/28, version 26, internal 0x5000001 0x40 (ptr 0x90cd2920) [1], 0x0 (0x0), 0x0 (0x0)

 Updated Feb 17 20:35:32.438
 Prefix Len 28, traffic index 0, precedence n/a, priority 4
  gateway array (0x90aa9a58) reference count 1, flags 0x102010, source rib (7), 0 backups
```

```
              [1 type 3 flags 0x48441 (0x90b5a148) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Feb 17 20:35:32.438
 LDI Update time Feb 17 20:35:32.438

  Level 1 - Load distribution: 0
  [0] via 10.3.40.40/32, recursive

   via 10.3.40.40/32, 3 dependencies, recursive, bgp-ext [flags 0x6020]
    path-idx 0 NHID 0x0 [0x90cd3250 0x0], Internal 0x9081e550
    next hop 10.3.40.40/32 via 10.3.40.40/32

    Load distribution: 0 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   HundredGigE0/0/0/1        10.3.40.40

   via 10.3.50.50/32, 2 dependencies, recursive, bgp-ext, backup [flags 0x6120]
    path-idx 1 NHID 0x0 [0x90cd3178 0x0]
    next hop 10.3.50.50/32 via 10.3.50.50/32
```

The following outputs display the forwarding entries for the EPE MPLS labels at an egress border router **after** the IP Lookup Fallback for EPE feature is enabled. Observe that a backup path is now programmed for an EPE local label.

```
RP/0/RP0/CPU0:R3# show mpls forwarding labels 24004 24005
Thu Feb 10 18:40:34.655 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24004  Unlabelled  No ID              Hu0/0/0/1    10.3.40.40      0
       Aggregate   No ID              default                      0            (!)
24005  Unlabelled  No ID              Hu0/0/0/2    10.3.50.50      0
       Aggregate   No ID              default                      0            (!)


RP/0/RP0/CPU0:R3# show mpls forwarding labels 24004 24005 detail

Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24004  Unlabelled  No ID              Hu0/0/0/1    10.3.40.40      0
     Updated: Feb 10 18:40:25.476
     Path Flags: 0x6000 [  ]
     Version: 18, Priority: 3
     Label Stack (Top -> Bottom): { Unlabelled }
     NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 1, Weight: 0
     MAC/Encaps: 14/14, MTU: 1500
     Outgoing Interface: HundredGigE0/0/0/1 (ifhandle 0x00000198)
     Packets Switched: 0

       Aggregate   No ID              default                      0            (!)
     Updated: Feb 10 18:40:25.476
     Path Flags: 0x100 [  BKUP, NoFwd ]
     Label Stack (Top -> Bottom): { }
     MAC/Encaps: 0/0, MTU: 0
     Packets Switched: 0
24005  Unlabelled  No ID              Hu0/0/0/2    10.3.50.50      0
     Updated: Feb 10 18:40:25.482
     Path Flags: 0x6000 [  ]
     Version: 19, Priority: 3
     Label Stack (Top -> Bottom): { Unlabelled }
```

```
     NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 1, Weight: 0
     MAC/Encaps: 14/14, MTU: 1500
     Outgoing Interface: HundredGigE0/0/0/2 (ifhandle 0x000001a0)
     Packets Switched: 0

       Aggregate   No ID              default                       0            (!)
     Updated: Feb 10 18:40:25.482
     Path Flags: 0x100 [  BKUP, NoFwd ]
     Label Stack (Top -> Bottom): { }
     MAC/Encaps: 0/0, MTU: 0
     Packets Switched: 0


RP/0/RP0/CPU0:R3# show bgp ipv4 unicast 161.1.1.0/28
Thu Feb  3 22:31:52.893 UTC
BGP routing table entry for 161.1.1.0/28
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 6           6
Last Modified: Feb  3 22:28:56.039 for 00:02:56
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.4
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  40 60
    10.3.40.40 from 10.3.40.40 (1.1.1.40)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 5
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    1.1.1.1
  50 60
    10.3.50.50 from 10.3.50.50 (1.1.1.50)
      Origin IGP, localpref 100, valid, external, group-best, backup, add-path
      Received Path ID 0, Local Path ID 2, version 6
      Origin-AS validity: (disabled)


RP/0/RP0/CPU0:R3# show cef 161.1.1.0/28 detail

161.1.1.0/28, version 24, internal 0x5000001 0x40 (ptr 0x90684920) [1], 0x0 (0x0), 0x0 (0x0)

 Updated Feb 10 17:37:16.610
 Prefix Len 28, traffic index 0, precedence n/a, priority 4
  gateway array (0x9045ba58) reference count 1, flags 0x102010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x9050c148) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Feb 10 17:37:16.610
 LDI Update time Feb 10 17:37:16.623

  Level 1 - Load distribution: 0
  [0] via 10.3.40.40/32, recursive

   via 10.3.40.40/32, 5 dependencies, recursive, bgp-ext [flags 0x6020]
    path-idx 0 NHID 0x0 [0x90684c08 0x0], Internal 0x90211730
    next hop 10.3.40.40/32 via 10.3.40.40/32

    Load distribution: 0 (refcount 1)
```

```
 Hash  OK  Interface                 Address
 0     Y   HundredGigE0/0/0/1        10.3.40.40

via 10.3.50.50/32, 4 dependencies, recursive, bgp-ext, backup [flags 0x6120]
 path-idx 1 NHID 0x0 [0x90685040 0x0]
 next hop 10.3.50.50/32 via 10.3.50.50/32
```

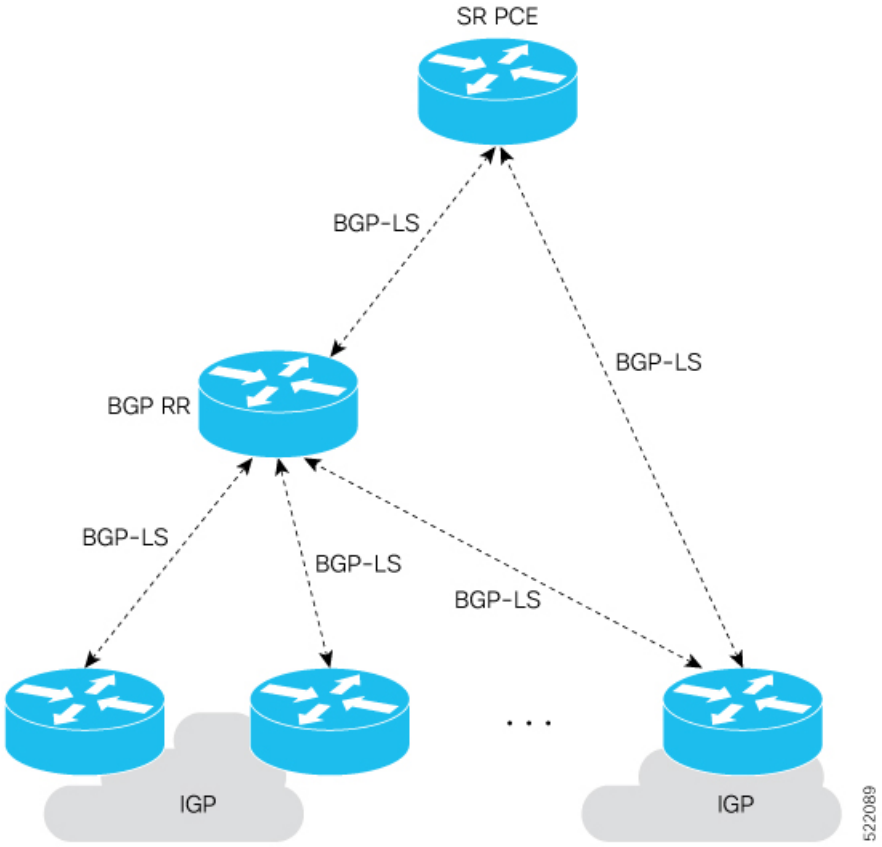
Configure BGP Link-State

BGP Link-State (LS) is an Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) originally defined to carry interior gateway protocol (IGP) link-state information through BGP. The BGP Network Layer Reachability Information (NLRI) encoding format for BGP-LS and a new BGP Path Attribute called the BGP-LS attribute are defined in RFC7752. The identifying key of each Link-State object, namely a node, link, or prefix, is encoded in the NLRI and the properties of the object are encoded in the BGP-LS attribute.

The BGP-LS Extensions for Segment Routing are documented in RFC9085.

BGP-LS applications like an SR Path Computation Engine (SR-PCE) can learn the SR capabilities of the nodes in the topology and the mapping of SR segments to those nodes. This can enable the SR-PCE to perform path computations based on SR-TE and to steer traffic on paths different from the underlying IGP-based distributed best-path computation.

The following figure shows a typical deployment scenario. In each IGP area, one or more nodes (BGP speakers) are configured with BGP-LS. These BGP speakers form an iBGP mesh by connecting to one or more route-reflectors. This way, all BGP speakers (specifically the route-reflectors) obtain Link-State information from all IGP areas (and from other ASes from eBGP peers).



Usage Guidelines and Limitations

- BGP-LS supports IS-IS and OSPFv2.
- The identifier field of BGP-LS (referred to as the Instance-ID) identifies the IGP routing domain where the NLRI belongs. The NLRIs representing link-state objects (nodes, links, or prefixes) from the same IGP routing instance must use the same Instance-ID value.
- When there is only a single protocol instance in the network where BGP-LS is operational, we recommend configuring the Instance-ID value to **0**.
- Assign consistent BGP-LS Instance-ID values on all BGP-LS Producers within a given IGP domain.
- NLRIs with different Instance-ID values are considered to be from different IGP routing instances.
- Unique Instance-ID values must be assigned to routing protocol instances operating in different IGP domains. This allows the BGP-LS Consumer (for example, SR-PCE) to build an accurate segregated multi-domain topology based on the Instance-ID values, even when the topology is advertised via BGP-LS by multiple BGP-LS Producers in the network.
- If the BGP-LS Instance-ID configuration guidelines are not followed, a BGP-LS Consumer may see duplicate link-state objects for the same node, link, or prefix when there are multiple BGP-LS Producers deployed. This may also result in the BGP-LS Consumers getting an inaccurate network-wide topology.

- The following table defines the supported extensions to the BGP-LS address family for carrying IGP topology information (including SR information) via BGP. For more information on the BGP-LS TLVs, refer to Border Gateway Protocol - Link State (BGP-LS) Parameters.

Table 36: IOS XR Supported BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
256	Local Node Descriptors	X	X	—
257	Remote Node Descriptors	X	X	—
258	Link Local/Remote Identifiers	X	X	—
259	IPv4 interface address	X	X	—
260	IPv4 neighbor address	X		
261	IPv6 interface address	X	—	—
262	IPv6 neighbor address	X	—	—
263	Multi-Topology ID	X	—	—
264	OSPF Route Type	—	X	—
265	IP Reachability Information	X	X	—
266	Node MSD TLV	X	X	—
267	Link MSD TLV	X	X	—
512	Autonomous System	—	—	X
513	BGP-LS Identifier	—	—	X
514	OSPF Area-ID	—	X	—
515	IGP Router-ID	X	X	—
516	BGP Router-ID TLV	—	—	X
517	BGP Confederation Member TLV	—	—	X
1024	Node Flag Bits	X	X	—
1026	Node Name	X	X	—
1027	IS-IS Area Identifier	X	—	—
1028	IPv4 Router-ID of Local Node	X	X	—
1029	IPv6 Router-ID of Local Node	X	—	—
1030	IPv4 Router-ID of Remote Node	X	X	—
1031	IPv6 Router-ID of Remote Node	X	—	—
1034	SR Capabilities TLV	X	X	—
1035	SR Algorithm TLV	X	X	—
1036	SR Local Block TLV	X	X	—

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
1039	Flex Algo Definition (FAD) TLV	X	X	—
1044	Flex Algorithm Prefix Metric (FAPM) TLV	X	X	—
1088	Administrative group (color)	X	X	—
1089	Maximum link bandwidth	X	X	—
1090	Max. reservable link bandwidth	X	X	—
1091	Unreserved bandwidth	X	X	—
1092	TE Default Metric	X	X	—
1093	Link Protection Type	X	X	—
1094	MPLS Protocol Mask	X	X	—
1095	IGP Metric	X	X	—
1096	Shared Risk Link Group	X	X	—
1099	Adjacency SID TLV	X	X	—
1100	LAN Adjacency SID TLV	X	X	—
1101	PeerNode SID TLV	—	—	X
1102	PeerAdj SID TLV	—	—	X
1103	PeerSet SID TLV	—	—	X
1114	Unidirectional Link Delay TLV	X	X	—
1115	Min/Max Unidirectional Link Delay TLV	X	X	—
1116	Unidirectional Delay Variation TLV	X	X	—
1117	Unidirectional Link Loss	X	X	—
1118	Unidirectional Residual Bandwidth	X	X	—
1119	Unidirectional Available Bandwidth	X	X	—
1120	Unidirectional Utilized Bandwidth	X	X	—
1122	Application-Specific Link Attribute TLV	X	X	—
1152	IGP Flags	X	X	—
1153	IGP Route Tag	X	X	—
1154	IGP Extended Route Tag	X	—	—
1155	Prefix Metric	X	X	—
1156	OSPF Forwarding Address	—	X	—
1158	Prefix-SID	X	X	—
1159	Range	X	X	—

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
1161	SID/Label TLV	X	X	—
1170	Prefix Attribute Flags	X	X	—
1171	Source Router Identifier	X	—	—
1172	L2 Bundle Member Attributes TLV	X	—	—
1173	Extended Administrative Group	X	X	—

Exchange Link State Information with BGP Neighbor

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

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# neighbor 10.0.0.2
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family link-state link-state
Router(config-bgp-nbr-af)# exit
```

IGP Link-State Database Distribution

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

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

```
Router# configure
Router(config)# router isis isp
Router(config-isis)# distribute link-state instance-id 32
```

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

```
Router# configure
Router(config)# router ospf 100
Router(config-ospf)# distribute link-state instance-id 32
```

Configurable Filters for IS-IS advertisements to BGP-Link State

Table 37: Feature History Table

Feature Name	Release Information	Feature Description
Configurable Filters for IS-IS Advertisements to BGP-Link State	Release 7.10.1	This feature allows you to configure a route map to filter IS-IS route advertisements to BGP-Link State (LS). It also provides a per-area configuration knob to disable IS-IS advertisements for external and propagated prefixes. This configuration of filters hence reduces the amount of redundant data for external and interarea prefixes sent to the BGP - LS clients. The feature introduces **exclude-external**, **exclude-interarea**, and **route-policy** *name* optional keywords in the **distribute link-state** command.

In a large IS-IS network, there are multiple routers in different areas distributing their link-state databases through BGP-LS. In addition, other protocols, such as OSPF do their own BGP-LS reporting and have routes that are redistributed into IS-IS. This can result in substantial amounts of redundant data for external and interarea prefixes which are sent to the BGP-LS clients only to be discarded.

Rather than sending redundant information, this feature provides the option of limiting the prefixes for which IS-IS TLV information is sent to BGP-LS.

There are three options to filter prefix Type-Length-Values (TLVs) that are reported in BGP-LS and the operators can specify these options on a per-level basis:

- **exclude-external**: Omits information for external prefixes that are redistributed from a different protocol or instance. These are identified by the “X” bit set in its Extended Reachability Attribute Flags or the ‘X’ bit of TLVs 236 and 237.
- **exclude-interarea**: Omits information for interarea prefixes and summaries. These are identified by the ‘R’ bit set in their Extended Reachability Attribute Flags or the ‘up or down’ bit set in TLVs 135, 235, 236, and 237.
- **route-policy***name*: Allows specification of a route-policy to provide filtering based on a set of destination prefixes.

The filtering is implemented at the point where the individual prefix TLVs are read from a label-switched path to generate updates to BGP-LS. It does not affect the advertisement of a node or the link information.

Configure Filters for IS-IS Advertisements to BGP-LS

Configuration Example

You can configure any of these filters for IS-IS advertisements to BGP-LS:

```
Router#config
Router(config)#router isis 1
```

```
Router(config-isis)#distribute link-state exclude-external
Router(config-isis)#commit


Router#config
Router(config)#router isis 1
Router(config-isis)#ddistribute link-state exclude-interarea
Router(config-isis)#commit


Router# config
Router(config)# router isis 1
Router(config-isis)#distribute link-state route-policy isis-rp-1
Router(config-isis)#commit
```

Note This feature does not introduce any new failure modes to IS-IS.

Running Configuration

To check the filter for IS-IS advertisements to BGP-LS, you can run the following command:

```
Router# show running-config
router isis 1
 distribute link-state exclude-external
 commit
 !
!


router isis 1
 distribute link-state exclude-interarea
 commit
 !
!


router isis 1
 distribute link-state route-policy isis-rp-1
 commit
 !
!
```

Configure BGP Proxy Prefix SID

Table 38: Feature History Table

Feature Name	Release	Description
BGP Proxy Prefix SID	Release 7.3.2	This feature is a BGP extension to signal BGP prefix-SIDs. This feature allows you to attach BGP prefix SID attributes for remote prefixes learned over BGP labeled unicast (LU) sessions and propagate them as SR prefixes using BGP LU. This allows an LSP towards non-SR endpoints to use segment routing global block in the SR domain.

To support segment routing, Border Gateway Protocol (BGP) requires the ability to advertise a segment identifier (SID) for a BGP prefix. A BGP-Prefix-SID is the segment identifier of the BGP prefix segment in a segment routing network. BGP prefix SID attribute is a BGP extension to signal BGP prefix-SIDs. However, there may be routers which do not support BGP extension for segment routing. Hence, those routers also do not support BGP prefix SID attribute and an alternate approach is required.

BGP proxy prefix SID feature allows you to attach BGP prefix SID attributes for remote prefixes learnt from BGP labeled unicast (LU) neighbours which are not SR-capable and propagate them as SR prefixes. This allows an LSP towards non SR endpoints to use segment routing global block in a SR domain. Since BGP proxy prefix SID uses global label values it minimizes the use of limited resources such as ECMP-FEC and provides more scalability for the networks.

BGP proxy prefix SID feature is implemented using the segment routing mapping server (SRMS). SRMS allows the user to configure SID mapping entries to specify the prefix-SIDs for the prefixes. The mapping server advertises the local SID-mapping policy to the mapping clients. BGP acts as a client of the SRMS and uses the mapping policy to calculate the prefix-SIDs.

Configuration Example:

This example shows how to configure the BGP proxy prefix SID feature for the segment routing mapping server.

```
RP/0/RSP0/CPU0:router(config)# segment-routing
RP/0/RSP0/CPU0:router(config-sr)# mapping-server
RP/0/RSP0/CPU0:router(config-sr-ms)# prefix-sid-map
RP/0/RSP0/CPU0:router(config-sr-ms-map)# address-family ipv4
RP/0/RSP0/CPU0:router(config-sr-ms-map-af)# 1.1.1.1/32 10 range 200
RP/0/RSP0/CPU0:router(config-sr-ms-map-af)# 192.168.64.1/32 400 range 300
```

This example shows how to configure the BGP proxy prefix SID feature for the segment-routing mapping client.

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# address-family ip4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# segment-routing prefix-sid-map
```

Verification

These examples show how to verify the BGP proxy prefix SID feature.

```
RP/0/RSP0/CPU0:router# show segment-routing mapping-server prefix-sid-map ipv4
 detail
Prefix
1.1.1.1/32
    SID Index:      10
    Range:          200
    Last Prefix:    1.1.1.200/32
    Last SID Index: 209
    Flags:
Number of mapping entries: 1


RP/0/RSP0/CPU0:router# show bgp ipv4 labeled-unicast 192.168.64.1/32

BGP routing table entry for 192.168.64.1/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               117         117
  Local Label: 16400
Last Modified: Oct 25 01:02:28.562 for 00:11:45Paths: (2 available, best #1)
 Advertised to peers (in unique update groups):
   201.1.1.1
 Path #1: Received by speaker 0  Advertised to peers (in unique update groups):
    201.1.1.1
  Local
   20.0.101.1 from 20.0.101.1 (20.0.101.1)      Received Label 61
   Origin IGP, localpref 100, valid, internal, best, group-best, multipath, labeled-unicast

   Received Path ID 0, Local Path ID 0, version 117
  Prefix SID Attribute Size: 7
  Label Index: 1

 RP/0/RSP0/CPU0:router# show route ipv4 unicast 192.68.64.1/32 detail

Routing entry for 192.168.64.1/32
  Known via "bgp 65000", distance 200, metric 0, [ei]-bgp, labeled SR, type internal
  Installed Oct 25 01:02:28.583 for 00:20:09
  Routing Descriptor Blocks
    20.0.101.1, from 20.0.101.1, BGP multi path
      Route metric is 0
      Label: 0x3d (61)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
   Route version is 0x6 (6)
  Local Label: 0x3e81 (16400)
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 4, Download Version 242
  No advertising protos.


RP/0/RSP0/CPU0:router# show cef ipv4 192.168.64.1/32 detail
192.168.64.1/32, version 476, labeled SR, drop adjacency, internal 0x5000001 0x80 (ptr
0x71c42b40) [1], 0x0 (0x71c11590), 0x808 (0x722b91e0)
 Updated Oct 31 23:23:48.733
```

```
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
 Extensions: context-label:16400
  gateway array (0x71ae7e78) reference count 3, flags 0x7a, source rib (7), 0 backups
                [2 type 5 flags 0x88401 (0x722eb450) ext 0x0 (0x0)]
  LW-LDI[type=5, refc=3, ptr=0x71c11590, sh-ldi=0x722eb450]
  gateway array update type-time 3 Oct 31 23:49:11.720
 LDI Update time Oct 31 23:23:48.733
 LW-LDI-TS Oct 31 23:23:48.733
   via 20.0.101.1/32, 0 dependencies, recursive, bgp-ext [flags 0x6020]
    path-idx 0 NHID 0x0 [0x7129a294 0x0]
    recursion-via-/32
    unresolved
     local label 16400
     labels imposed {ExpNullv6}
```

```
RP/0/RSP0/CPU0:router# show bgp labels
BGP router identifier 2.1.1.1, local AS number 65000
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 245
BGP main routing table version 245
BGP NSR Initial initsync version 16 (Reached)
BGP NSR/ISSU Sync-Group versions 245/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop        Rcvd Label      Local Label
*>i1.1.1.1/32         1.1.1.1         3               16010
*> 2.1.1.1/32         0.0.0.0         nolabel         3
*> 192.68.64.1/32      20.0.101.1      2               16400
*> 192.68.64.2/32      20.0.101.1      2               16401
```

BGP Best Path Computation using SR Policy Paths

Table 39: Feature History Table

Feature Name	Release Information	Feature Description
BGP Best Path Computation using SR Policy Paths	Release 7.5.2 Release 7.3.4	BGP best-path selection is modified for a prefix when at least one of its paths resolves over the next hop using SR policies (SR policy in "up" state). Under this condition, paths not steered over an SR policy (those using native next-hop resolution) are considered ineligible during best-path selection. You can thus control the best path selection in order to steer traffic, preferably or exclusively, over SR policies with the desired SLA. This feature introduces the **bgp bestpath sr-policy {force | prefer}** command.

BGP selects the best path from the available pool of paths such as iBGP, eBGP, color, or noncolor paths with native next hop and SR policy next hop. BGP uses either native next hop or an SR policy next hop for best path computation. However, BGP might not consider SR policy next hop for best path computation due to other factors in best path selection. By default, BGP considers a native next hop for the best path computation during the failure.

For more information, see Best path calculation algorithm.

When multiple advertisements of the same BGP prefix are received where some have extended community color, SRTE headend with BGP multi-path enabled installs multiple routes with or without extended community color. It may be required to exclude the path resolving over native next hop SR policy paths from BGP best path selection when a prefix has multiple paths in the presence of one BGP path with the extended community color that is resolved over the SR policy.

You may want to use the egress PE to exit a domain using local preference or other attributes before the next hop metric selection. In such scenarios, when SR policy of the primary path fails, the best path is resolved over a regular IGP next hop that is the default mode of operation. Traffic doesn't select the backup path with SR policy, instead traffic moves to native LSP on the primary path.

The BGP Best Path Computation using SR Policy Paths feature allows the BGP to use the path with SR policy as the best-path, backup, and multipath.

When this feature is enabled, some paths are marked as an ineligible path for BGP best path selection. Existing BGP best path selection order is applied to the eligible paths.

Use either of the following modes for the BGP to select the SR policy path as the best path for the backup path:

- Force mode: When force mode is enabled, only SR policy paths are considered for best path calculation. Use the **bgp bestpath sr-policy force** command to enable this mode.

  In a network, when at least one path has an active SR policy, the following paths are marked as ineligible for best path selection:

  - iBGP paths with noncolor or color paths with SR policy that isn't active.
  - eBGP with color and SR policy isn't active.
  - eBGP noncolor paths

Note Local and redistributed BGP paths are always eligible for best path selection.

- Prefer mode: When prefer mode is enabled, SR policy paths and eBGP noncolor paths are eligible for best path calculation.

  Use the **bgp bestpath sr-policy prefer** command to enable this mode.

  In a network, when at least one path has an active SR policy, the following paths are marked as ineligible for best path selection:

  - iBGP paths with noncolor or color paths with SR policy that isn't active.
  - eBGP with color and SR policy isn't active.

Note Local and redistributed BGP paths are always eligible for best path selection.

Configure BGP Best Path Computation using SR Policy Paths

To enable the feature, perform the following tasks on the ingress PE router that is the head-end of SR policy:

- Configure route policy.
- Configure SR policy.
- Configure BGP with either prefer or force mode.

Configuration Example

Configure route policies on the egress PE router:

```
Router(config)#extcommunity-set opaque color9001
Router(config-ext)#9001 co-flag 01
Router(config-ext)#end-set
Router(config)#extcommunity-set opaque color9002
Router(config-ext)#9002 co-flag 01
Router(config-ext)#end-set
Router(config)#commitC

Router(config)#route-policy for9001
Router(config-rpl)#set extcommunity color color9001
```

```
Router(config-rpl)# pass
Router(config-rpl)#end-policy

Router(config)#route-policy for9002
Router(config-rpl)#set extcommunity color color9002
Router(config-rpl)#pass
Router(config-rpl)#end-policy
Router(config)#commit


Router#configure
Router(config)#route-policy add_path
Router(config-rpl)#set path-selection backup 1 install multipath-protect advertise
multipath-protect-advertise
Router(config-rpl)#end-policy

Router(config)#route-policy pass-all
Router(config-rpl)#pass
Router(config-rpl)#end-policy
Router(config)#commit
```

Configure SR policy on the egress PE router:

```
Router#configure
Router(config)#segment-routing
Router(config-sr)#traffic-eng
Router(config-sr-te)#segment-list SL201
Router(config-sr-te-sl)#index 1 mpls label 25000
Router(config-sr-te-sl)#policy POLICY_9001
Router(config-sr-te-policy)#binding-sid mpls 47700
Router(config-sr-te-policy)#color 9001 end-point ipv6 ::
Router(config-sr-te-policy)#candidate-paths
Router(config-sr-te-policy-path)#preference 10
Router(config-sr-te-policy-path-pref)#explicit segment-list SL201
Router(config-sr-te-sl)#policy POLICY_9002
Router(config-sr-te-policy)#binding-sid mpls 47701
Router(config-sr-te-policy)#color 9002 end-point ipv6 ::
Router(config-sr-te-policy)#candidate-paths
Router(config-sr-te-policy-path)#preference 10
Router(config-sr-te-policy-path-pref)#explicit segment-list SL201
Router(config-sr-te-policy-path-pref)#commit
```

Configure BGP on the Egress PE router:

```
Router(config)#router bgp 100
Router(config-bgp)#nsr
Router(config-bgp)#bgp router-id 10.1.1.2
Router(config-bgp)#bgp best-path sr-policy force
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#maximum-paths eibgp 25
Router(config-bgp-af)#additional-paths receive
Router(config-bgp-af)#additional-paths send
Router(config-bgp-af)#additional-paths selection route-policy add_path
Router(config-bgp-af)#redistribute connected
Router(config-bgp-af)#redistribute static
Router(config-bgp-af)#allocate-label all
Router(config-bgp-af)#commit
Router(config-bgp-af)#exit
Router(config-bgp)#neighbor 31::2
Router(config-bgp-nbr)#remote-as 2
Router(config-bgp-nbr)#address-family ipv6 unicast
Router(config-bgp-nbr-af)#route-policy for9001 in
Router(config-bgp-nbr-af)#route-policy pass-all out
```

```
Router(config-bgp-nbr-af)#commit
Router(config-bgp-nbr-af)#exit
Router(config-bgp)#neighbor 32::2
Router(config-bgp-nbr)#remote-as 2
Router(config-bgp-nbr)#address-family ipv6 unicast
Router(config-bgp-nbr-af)#route-policy for9002 in
Router(config-bgp-nbr-af)#route-policy pass-all out
Router(config-bgp-nbr-af)#commit
```

Verification

The following show output shows that when the **force** option is enabled, the configured SR policy path is selected as the best path instead of the default best path.

```
Router#show bgp ipv6 unicast 2001:DB8::1 brief
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
* 2001:DB8::1         10:1:1::55                   100      0 2 i
* i                   10:1:1::55                   100      0 2 i

*                     30::2                                 0 2 I
*>                    31::2 C:9001                          0 2 I
*                     32::2 C:9002                          0 2 I
Router#
```

Use the following command to compare the best paths:

```
Router#show bgp ipv6 unicast 2001:DB8::1 bestpath-compare
BGP routing table entry for 2001:DB8::1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker              7641        7641
    Flags: 0x240232b2+0x20050000; multipath; backup available;
Last Modified: Dec  7 03:43:57.200 for 00:34:48
Paths: (24 available, best #4)
  Advertised IPv6 Unicast paths to update-groups (with more than one peer):
    0.3 0.4
  Advertised IPv6 Unicast paths to peers (in unique update groups):
    10.1.1.55
  Path #1: Received by speaker 0
  Flags: 0x2000000000020005, import: 0x20
  Flags2: 0x00
  Not advertised to any peer
  2
    10:1:1::55 (metric 30) from 10.1.1.55 (10.1.1.55), if-handle 0x00000000
      Origin IGP, localpref 100, valid, internal
      Received Path ID 1, Local Path ID 0, version 0
      Extended community: Color[CO-Flag]:8001[01]
      Non SR-policy path is ignored due to config knob
  Path #2: Received by speaker 0
  Flags: 0x2000000000020005, import: 0x20
  Flags2: 0x00
  Not advertised to any peer
  2
    10:1:1::55 (metric 30) from 10.1.1.55 (10.1.1.55), if-handle 0x00000000
      Origin IGP, localpref 100, valid, internal
      Received Path ID 3, Local Path ID 0, version 0
      Extended community: Color[CO-Flag]:8002[01]
      Non SR-policy path is ignored due to config knob
  Path #3: Received by speaker 0
  Flags: 0x3000000000060001, import: 0x20
```

```
Flags2: 0x00
Advertised IPv6 Unicast paths to update-groups (with more than one peer):
  0.4
Advertised IPv6 Unicast paths to peers (in unique update groups):
  10.1.1.55
2
  30::2 from 30::2 (198.51.100.1), if-handle 0x00000000
    Origin IGP, localpref 100, weight 65534, valid, external, backup, add-path
    Received Path ID 0, Local Path ID 2, version 7641
    Origin-AS validity: (disabled)
    Non SR-policy path is ignored due to config knob
Path #4: Received by speaker 0
Flags: 0xb000000001070001, import: 0x20
Flags2: 0x00
Advertised IPv6 Unicast paths to update-groups (with more than one peer):
  0.3 0.4
Advertised IPv6 Unicast paths to peers (in unique update groups):
  10.1.1.55
2
  31::2 C:9001 (bsid:48900) from 31::2 (198.51.100.2), if-handle 0x00000000
    Origin IGP, localpref 100, valid, external, best, group-best, multipath
    Received Path ID 0, Local Path ID 1, version 7641
    Extended community: Color[CO-Flag]:9001[01]
    Origin-AS validity: (disabled)
    SR policy color 9001, ipv6 null endpoint, up, not-registered, bsid 48900

    best of AS 2, Overall best
Path #5: Received by speaker 0
Flags: 0xb000000000030001, import: 0x20
Flags2: 0x00
Not advertised to any peer
2
  32::2 C:9002 (bsid:48901) from 32::2 (198.51.100.3), if-handle 0x00000000
    Origin IGP, localpref 100, valid, external, multipath
    Received Path ID 0, Local Path ID 0, version 0
    Extended community: Color[CO-Flag]:9002[01]
    Origin-AS validity: (disabled)
    SR policy color 9002, up, not-registered, bsid 48901
    Higher router ID than best path (path #4)
```

Use the **show bgp process** command to verify which mode is enabled.

In the following example, you see that the **force** mode is enabled.

```
Router#show bgp process
BGP Process Information:
BGP is operating in STANDALONE mode
Autonomous System number format: ASPLAIN
Autonomous System: 100
Router ID: 10.1.1.2 (manually configured)
Default Cluster ID: 10.1.1.2
Active Cluster IDs:  10.1.1.2
Fast external fallover enabled
Platform Loadbalance paths max: 64
Platform RLIMIT max: 8589934592 bytes
Maximum limit for BMP buffer size: 1638 MB
Default value for BMP buffer size: 1228 MB
Current limit for BMP buffer size: 1228 MB
Current utilization of BMP buffer limit: 0 B
Neighbor logging is enabled
Enforce first AS enabled
Use SR-Policy admin/metric of color-extcomm Nexthop during path comparison: disabled
SR policy path force is enabled
Default local preference: 100
```

```
Default keepalive: 60
Non-stop routing is enabled
Slow peer detection enabled
ExtComm Color Nexthop validation: RIB

Update delay: 120
Generic scan interval: 60
Configured Segment-routing Local Block: [0, 0]
In use Segment-routing Local Block: [15000, 15999]
Platform support mix of sr-policy and native nexthop: Yes

Address family: IPv4 Unicast
Dampening is not enabled
Client reflection is enabled in global config
Dynamic MED is Disabled
Dynamic MED interval : 10 minutes
Dynamic MED Timer : Not Running
Dynamic MED Periodic Timer : Not Running
Scan interval: 60
Total prefixes scanned: 33
Prefixes scanned per segment: 100000
Number of scan segments: 1
Nexthop resolution minimum prefix-length: 0 (not configured)
IPv6 Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 12642
Table version synced to RIB: 12642
Table version acked by RIB: 12642
IGP notification: IGPs notified
RIB has converged: version 2
RIB table prefix-limit reached ?  [No], version 0
Permanent Network Unconfigured

Node                Process     Nbrs Estb Rst Upd-Rcvd Upd-Sent Nfn-Rcv Nfn-Snt
node0_RSP1_CPU0     Speaker       53    3   2      316      823       0      53
```

SRv6 Double Recursion for Multi-Layer BGP Underlay

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Double Recursion for Multi-Layer BGP Underlay	Release 24.4.1	

Feature Name	Release Information	Feature Description
		Introduced in this release on: Fixed Systems (8100, 8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The feature introduces support to SRv6 double recursion where a network service such as BGP VPN (Layer 2/Layer 3) requires multiple layers of resolution, specifically where one routing layer resolves over another before reaching its final destination. You can achieve double recursion by collapsing the underlay, which typically involves protocols like IGP or BGP in the packet forwarding chain, you can achieve three level load balancing, allowing an even distribution of traffic across multiple layers of the network stack. The feature is supported on the ingress Provider Edge (PE) router. Previously, SRv6 supported only two levels of load balancing, which works for traditional service provider setups. The feature introduces these changes: **CLI:** • **tag-map tag** <*value*> **map forwarding-hierarchy level-2-used-as-nexthop** • The **show cef ipv6** *ipv6-prefixes* and **show cef ipv4** *ipv4-prefixes* commands are enhanced to include the Layer 2 prefix information, which resolve as nexthop Layer 3 prefixes. **Yang Data Models:** • `Cisco-IOS-XR-um-router-rib-cfg:router` (see GitHub, Yang Data Models Navigator) data

Feature Name	Release Information	Feature Description
		model.

SRv6 Double Recursion and Load Balancing

SRv6 double recursion refers to scenarios where a network service such as BGP VPN (Layer 2/Layer 3) requires multiple layers of resolution, specifically where one routing layer resolves over another before reaching its final destination.

The SRv6 double recursion feature is designed to support three level load balancing by collapsing the underlay, which typically involves protocols like IGP or BGP in the forwarding chain. This involves specific configurations in RIB, BGP, and static routes to indicate IPv4 and IPv6 unicast routes with Layer 2 prefixes, which resolve as nexthop Layer 3 prefixes.

Without this feature, the traffic drops the Layer 3 prefixes leading to packet loss.

Key Benefits of SRv6 Double Recursion

- Support for complex network scenarios: In traditional service provider (SP) networks, BGP VPN services typically resolve directly over IGP reachability, meaning there is a straightforward, single-level resolution path from the service to the IGP underlay. However, with the evolution of networking, especially in data centers, new architectures have emerged where BGP underlay networks are used. In these scenarios, BGP VPN services resolve over a BGP underlay, which in turn resolves over IGP or directly connected routes.
- Support for enhanced load balancing: SRv6 double recursion allows the routing platform to balance traffic more efficiently across all available paths, enhancing overall network performance and reducing congestion. Proper load balancing at each level of recursion helps in distributing traffic evenly, avoiding bottlenecks, and ensuring that packets take optimal paths even in complex topologies.
- Improved network flexibility: Double recursion allows advanced routing solutions in SP and data center networks, accommodating complex use cases beyond single-layer limitations.

Single and Double Recursion in BGP VPN Services

To optimize the network routing efficiency, it is important to understand the resolution processes of BGP VPN services.

- Single Recursion: BGP VPN service → IGP reachability. In single recursion, the BGP VPN service directly resolves to IGP (Interior Gateway Protocol) reachability. Single recursion is required when a BGP VPN service can directly resolve to an IGP route without any intermediary steps.
- Double Recursion: BGP VPN service → BGP underlay reachability → IGP reachability. In double recursion, the BGP VPN service first resolves over a BGP underlay, which then leads to IGP reachability. Double recursion is necessary in cases where a BGP VPN service cannot resolve directly to an IGP route but instead needs to traverse a BGP underlay first. This creates a layered resolution process that standard single-level load balancing cannot handle effectively.

Usage Guidelines and Limitations for SRv6 Double Recursion

The following usage guidelines and limitations apply:

- Set the locator prefix for IPv6 prefixes that are Layer 2 prefixes.
- If the collapsed BGP paths is a combination of IGP IPv6 and SRv6, the router filters out only the IPv6 paths.
- You cannot use a combination of BGP-SU (BGP Service Unicast) and BGP-IP paths at Layer 2.
- For a collapsed chain, there must be an encapsulation ID on BGP if IGP is SRv6.

Configure SRv6 Double Recursion for Multi-Layer BGP Underlay

To configure SRv6 double recursion for multi-layer BGP underlay:

- Enable the hardware support for BGP-LU.
- Configure the BGP-SR or BGP-IP IPv4 and IPv6 unicast routes in the RIB, to ensure that these Layer 2 prefixes could be used as nexthop for Layer 3 prefixes.
- Assign or map specific tags to Layer 2 routes in the RIB for IPv4 and IPv6 unicast routes in the VRF.

Before you begin

- For Layer 2 IPv6 prefixes, set the locator prefix using the **prefix-set** command in RIB.
- Configure the set of IPv4 and IPv6 Layer 2 prefixes, which resolve as nexthop Layer 3 prefixes using the **prefix-set** command.

Procedure

Step 1 Enable the hardware support for BGP-LU to allow the ingress PE router to advertise and forward the MPLS-labelled unicast routes.

Example:

```
Router#config
Router(config)#hw-module profile cef bgplu enable
```

Step 2 Configure the IPv4 and IPv6 unicast routes in the RIB, ensuring that these Layer 2 prefixes could be used as nexthop for Layer 3 prefixes.

Example:

```
Router#config
Router(config)#router bgp 100
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#table-policy level2-ipv4-policy
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#table-policy level2-ipv6-policy
Router(config-bgp-af)#commit
```

Step 3 Assign the required tag to the IPv4 and IPv6 unicast routes.

For these IPv4 and IPv6 routes, the RIB assigns the tags and forwards the route information to the Forwarding Information Base (FIB), indicating that these Layer 2 prefixes resolve as nexthop for Layer 3 prefixes.

Example:

IPv4 Layer 2 prefixes:

```
Router(config)#route-policy level2-ipv4-policy
Router(config-rpl)#if destination in level2_prefixes-ipv4 then
Router(config-rpl-if)#set tag 100
Router(config-rpl-if)#else
Router(config-rpl-else)#pass
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
Router(config)#commit
```

Example:

IPv6 Layer 2 prefixes:

```
Router(config)#route-policy level2-ipv6-policy
Router(config-rpl)#if destination in level2_prefixes-ipv6 then
Router(config-rpl-if)#set tag 100
Router(config-rpl-if)#endif
Router(config-rpl)#if destination in level2_locators then
Router(config-rpl-if)#set locator-prefix
Router(config-rpl-if)#else
Router(config-rpl-else)#pass
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
```

Step 4 Map the tags in the RIB for the IPv4 and IPv6 unicast routes.

In the following example, the RIB maps all the IPv4 and IPv6 routes tagged with the value 100, which indicates that these routes resolve as nexthop for Layer 3 prefixes. The RIB adds the *FIB_UPDATE_ROUTE_FLAG_EXTN_LVL2_HAS_DEPENDENT* flag when it sends the route update to the FIB.

Example:

```
Router#config
Router(config)#router rib
Router(config-rib)#tag-map tag 100 map forwarding-hierarchy level-2-used-as-nexthop
Router(config-rib)#commit
```

Step 5 Verify the running configuration using the **show running-config** command.

Example:

```
hw-module profile cef bgplu enable
!
router bgp 100
 address-family ipv4 unicast
  table-policy level2-ipv4-policy
 !
 address-family ipv6 unicast
  table-policy level2-ipv6-policy
 !
!
router rib
 tag-map tag 100 map forwarding-hierarchy level-2-used-as-nexthop
 !
!
```

Step 6 Use the **show cef ipv4** and **show cef ipv6** commands to verify the Layer 2 collapsed prefixes.

In the following example, the IPv6 Layer 2 prefixes, which resolve as nexthop for Layer 3 prefixes are collapsed. This is indicated by the **collapsed** keyword in the output. The SRv6 SID lists indicate the different encapsulation layers or hierarchy.

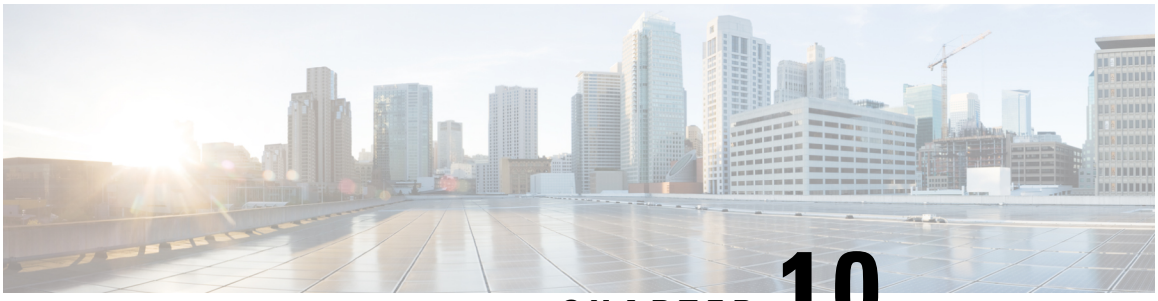
Example:

```
Router#show cef ipv6 2001:DB8:A:B::1/64
Thu Jun  6 12:48:52.399 EDT
2001:DB8:A:B::1/64, version 8, SRv6 Headend, internal 0x1000001 0x0 (ptr 0x63851c98) [1], 0x1400
(0x63851da0), 0x0 (0x638b2128)
 Updated Jun  6 12:41:10.589
 Prefix Len 64, traffic index 0, precedence n/a, priority 0, encap-id 0x11deadbeef
  gateway array (0x61e1a798) reference count 1, flags 0x10, source rib (7), 0 backups
                [2 type 3 flags 0x40008501 (0x63853e38) ext 0x0 (0x0) (collapsed)]
  LW-LDI[type=3, refc=1, ptr=0x63851da0, sh-ldi=0x63853e38]
  gateway array update type-time 1 Jun  6 12:41:10.589
 LDI Update time Jun  6 12:41:10.629
 LW-LDI-TS Jun  6 12:41:10.629
 Accounting: Disabled
   via 2001:DB8::1/128, 1 dependency, recursive [flags 0x3000000]
    path-idx 0 NHID 0x0 [0x63a2c098 0x0]
    next hop 2001:DB8::1/128 via  2001:DB8::1
    SRv6 H.Encaps.Red SID-list {2001:DB8:1:e002::}
     SRv6 H.Insert.Red SID-list {}
      SRv6 H.Insert.Red SID-list {2001:DB8:3:: 2001:DB8:4::}
   via 2001:DB8::1/128, 1 dependency, recursive [flags 0x3000000]
    path-idx 1 NHID 0x0 [0x63a2c270 0x0]
    next hop 2001:DB8::1/128 via 2001:DB8::1
    SRv6 H.Encaps.Red SID-list {2001:DB8:1:e002::}
       SRv6 H.Insert.Red SID-list {}

    Load distribution: 0 1 2 2 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   UNKNOWN intf 0x00000013   10::2
    1     Y   UNKNOWN intf 0x00000014   20::2
    2     Y   UNKNOWN intf 0x00000013   10::2
    3     Y   UNKNOWN intf 0x00000013   10::2
```

Step 7 For IPv6 or IPv4 static routes, use the **router static** command to configure and map the tags in RIB:

Example:

```
Router#config
Router(config)#router static
Router(config-static)#address-family ipv6 unicast
Router(config-static-afi)#2001:DB8:8::/48 4::4 tag 100
Router(config-static-afi)#commit
```



CHAPTER 10

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.

Table 41: Feature History Table

Feature Name	Release Information	Feature Description
Local ECMP Support on SRTE	Release 7.10.1	This feature supports the top security identifier (SID) resolved route interior gateway protocol (IGP) path's weight to be used in relative weight computation of outgoing paths of the SRTE policy. Now ECMP supports SRTE and computes per-egress-path relative weight using underlay (IGP) weight from Forwarding Information Base (FIB) / Routing Information Base (RIB) in addition to Segment List (SL) weight and number of primary egress paths for SL. Thus, IGP load balancing required at individual paths is also maintained. This feature is enabled by default and no configuration changes are required.
Segment Routing Traffic Engineering (SR-TE)	Release 7.5.2	You create a policy to steer traffic between a source-and-destination pair using the concept of source routing, where the source calculates the path and encodes it in the packet header as a segment. This functionality uses a single intelligent source and does not rely on the remaining nodes to compute a path through the network. This feature utilizes network bandwidth more effectively than traditional MPLS-TE networks by using ECMP at every segment level. Cisco 8000 series routers support the imposition of up to 8 MPLS transport labels, including TI-LFA backup labels for the protection of the top SID of an SR policy.

- SR-TE Policy Overview, on page 350
- Usage Guidelines and Limitations, on page 350
- Instantiation of an SR Policy, on page 352
- SR-TE Policy Path Types, on page 367
- Protocols, on page 383
- Traffic Steering, on page 390

• Enabling SR-TE with Next-Hop Independent Scaling Optimization, on page 421
• Miscellaneous, on page 422

SR-TE Policy Overview

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 following 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.

An SR-TE policy is identified as an ordered list (head-end, color, end-point):

- Head-end – Where the SR-TE policy is instantiated
- Color – A numerical value that distinguishes between two or more policies to the same node pairs (Head-end – End point)
- End-point – The destination of the SR-TE policy

Every SR-TE policy has a color value. Every policy between the same node pairs requires a unique color value.

An SR-TE policy uses one or more candidate paths. A candidate path is a single segment list (SID-list) or a set of weighted SID-lists (for weighted equal cost multi-path [WECMP]). A candidate path is either dynamic or explicit. See *SR-TE Policy Path Types* section for more information.

Usage Guidelines and Limitations

Observe the following guidelines and limitations for the platform.

- Broadcast links are not supported, configure IGP's interface as P2P (point-to-point).
- Before configuring SR-TE policies, use the **distribute link-state** command under IS-IS or OSPF to distribute the link-state database to external services.
- GRE tunnel as primary interface for an SR policy is not supported.
- GRE tunnel as backup interface for an SR policy with TI-LFA protection is not supported.
- Head-end computed inter-domain SR policy with Flex Algo constraint and IGP redistribution is not supported.
- The number of segment-lists (SLs) per SR policy is limited to a maximum of seven. If you need a policy with more than seven segment-lists, perform the following.
  1. Delete the existing SR policies.
  2. Configure the following command:

     ```
     Router(config)#hw-module profile cef te-tunnel highscale-no-ldp-over-te
     ```
  3. Reload the router for the configuration to take effect.
  4. Configure SR policy with more than seven segment-lists.

Note If you configure this command, the Autoroute feature will not work.

- For SR over SR policy traffic, the hardware can manage traffic accounting either for the SR label or SR policy but not for both. By default, the accounting of SR over SRTE traffic happens on SR policy. Perform the following steps to manage traffic using SR label accounting instead of SR policy accounting:

  1. Configure the following command:

     ```
     Router(config)#hw-module profile cef label-over-te-counters
     ```

  2. Reload the router for the configuration to take effect.

The following features are supported for SR-TE:

- SR-TE On-Demand Next Hop/Automated Steering (ODN/AS) is supported for global IPv4 BGP and IPv6 BGP prefixes with colors
- SR-TE BSID-based AS
- SR-TE head-end path computation
- LFA at SR-TE head-end
- Per-SR policy BSID label counters
- Per-SR policy aggregate counters
- Per-SR policy, per-segment list aggregate counters
- Per-SR policy, per-segment list, per-protocol aggregate counters:
  - Unlabeled IP – Unlabeled IPv4 and IPv6 traffic steered over SR policy
  - Labeled MPLS – Labeled traffic with BSID as top of label stack steered over SR policy
- Supports PCEP at SR-TE head-end
- Supports TI-LFA at SR-TE head-end
- Supports a maximum SID Depth (MSD) of 8 MPLS labels. In case it exceeds 8 MPLS labels, backup path is not created.
- SR-TE policy with autoroute-include based steering.
- Support for a BGP service route with a combination of paths pointing to SR native LSP with protection and paths pointing to SR-TE/BSID.

The following features are not supported:

- SR-TE ODN/AS for 6PE, VPNv4, VPNv6 (6vPE), EVPN
- SR-TE per-flow policy (PFP)
- Static Route Traffic-Steering using SR-TE Policy
- LDP over SR-TE Policy

- Mix of BSID and native paths with protection
- IPv6 IGP Routes over SRTE Policy

Instantiation of an SR Policy

An SR policy is instantiated, or implemented, at the head-end router.

The following sections provide details on the SR policy instantiation methods:

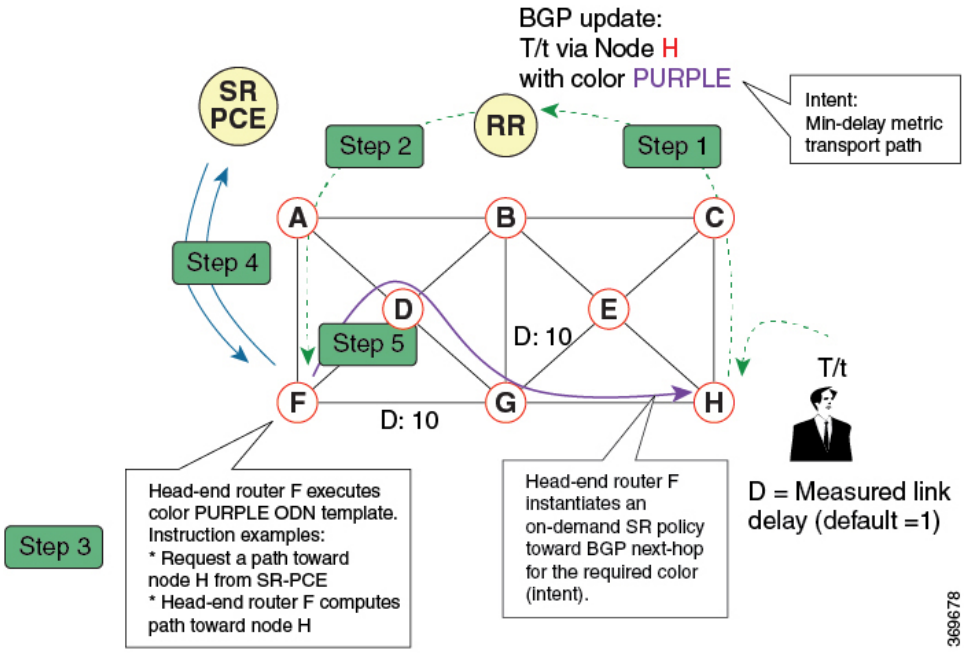
- On-Demand SR Policy – SR On-Demand Next-Hop , on page 352
- Manually Provisioned SR Policy, on page 364

On-Demand SR Policy – SR On-Demand Next-Hop

Segment Routing On-Demand Next Hop (SR-ODN) allows a service head-end router to automatically instantiate an SR policy to a BGP next-hop when required (on-demand). Its key benefits include:

- **SLA-aware BGP service** – Provides per-destination steering behaviors where a prefix, a set of prefixes, or all prefixes from a service can be associated with a desired underlay SLA. The functionality applies equally to single-domain and multi-domain networks.
- **Simplicity** – No prior SR Policy configuration needs to be configured and maintained. Instead, operator simply configures a small set of common intent-based optimization templates throughout the network.
- **Scalability** – Device resources at the head-end router are used only when required, based on service or SLA connectivity needs.

The following example shows how SR-ODN works:



1. An egress PE (node H) advertises a BGP route for prefix T/t. This advertisement includes an SLA intent encoded with a BGP color extended community. In this example, the operator assigns color purple (example value = 100) to prefixes that should traverse the network over the delay-optimized path.
2. The route reflector receives the advertised route and advertises it to other PE nodes.
3. Ingress PEs in the network (such as node F) are pre-configured with an ODN template for color purple that provides the node with the steps to follow in case a route with the intended color appears, for example:
   - Contact SR-PCE and request computation for a path toward node H that does not share any nodes with another LSP in the same disjointness group.
   - At the head-end router, compute a path towards node H that minimizes cumulative delay.
4. In this example, the head-end router contacts the SR-PCE and requests computation for a path toward node H that minimizes cumulative delay.
5. After SR-PCE provides the compute path, an intent-driven SR policy is instantiated at the head-end router. Other prefixes with the same intent (color) and destined to the same egress PE can share the same on-demand SR policy. When the last prefix associated with a given [intent, egress PE] pair is withdrawn, the on-demand SR policy is deleted, and resources are freed from the head-end router.

An on-demand SR policy is created dynamically for BGP global or VPN (service) routes. The following services are supported with SR-ODN:

- IPv4 BGP global routes
- IPv6 BGP global routes (6PE)
- VPNv4
- VPNv6 (6vPE)

SR-ODN Configuration Steps

Note In Cisco IOS XR release 7.5.2, SR-ODN configurations with Flexible Algorithm constraints can be configured using either of the following commands:

- **segment-routing traffic-eng on-demand color** *color* **dynamic sid-algorithm** *algorithm-number*
- **segment-routing traffic-eng on-demand color** *color* **constraints segments sid-algorithm** *algorithm-number*

Starting with Cisco IOS XR release 7.9.1, the **dynamic sid-algorithm** *algorithm-number* command has been deprecated. Only the **constraints segments sid-algorithm** *algorithm-number* command is supported. Configurations stored in NVRAM using the **dynamic sid-algorithm** *algorithm-number* command will be rejected at boot-up.

As a result, SR-ODN configurations with Flexible Algorithm constraints using the **dynamic sid-algorithm** *algorithm-number* CLI must be re-configured using the **constraints segments sid-algorithm** *algorithm-number* CLI.

To configure SR-ODN, complete the following configurations:

1. Define the SR-ODN template on the SR-TE head-end router.

   (Optional) If using Segment Routing Path Computation Element (SR-PCE) for path computation:

   a. Configure SR-PCE. For detailed SR-PCE configuration information, see Configure SR-PCE, on page 516.

   b. Configure the head-end router as Path Computation Element Protocol (PCEP) Path Computation Client (PCC). For detailed PCEP PCC configuration information, see Configure the Head-End Router as PCEP PCC, on page 384.

2. Define BGP color extended communities. Refer to the "Implementing BGP" chapter in the *BGP Configuration Guide for Cisco 8000 Series Routers*.

3. Define routing policies (using routing policy language [RPL]) to set BGP color extended communities. Refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

   The following RPL attach-points for setting/matching BGP color extended communities are supported:

Note The following table shows the supported RPL match operations; however, routing policies are required primarily to set BGP color extended community. Matching based on BGP color extended communities is performed automatically by ODN's on-demand color template.

Attach Point	Set	Match
VRF export	X	X
VRF import	–	X
EVI export	X	–
EVI import	X	X
Neighbor-in	X	X
Neighbor-out	X	X
Inter-AFI export	–	X
Inter-AFI import	–	X
Default-originate	X	–

4. Apply routing policies to a service. Refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Configure On-Demand Color Template

- Use the **on-demand color** *color* command to create an ODN template for the specified color value. The head-end router automatically follows the actions defined in the template upon arrival of BGP global or VPN routes with a BGP color extended community that matches the color value specified in the template.

The *color* range is from 1 to 4294967295.

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# on-demand color 10
```

Note Matching based on BGP color extended communities is performed automatically via ODN's on-demand color template. RPL routing policies are not required.

- Use the **on-demand color** *color* **dynamic** command to associate the template with on-demand SR policies with a locally computed dynamic path (by SR-TE head-end router utilizing its TE topology database) or centrally (by SR-PCE). The head-end router will first attempt to install the locally computed path; otherwise, it will use the path computed by the SR-PCE.

  ```
  Router(config)# segment-routing traffic-eng
  Router(config-sr-te)# on-demand color 10 dynamic
  ```

- Use the **on-demand color** *color* **dynamic pcep** command to indicate that only the path computed by SR-PCE should be associated with the on-demand SR policy. With this configuration, local path computation is not attempted; instead the head-end router will only instantiate the path computed by the SR-PCE.

  ```
  Router(config-sr-te)# on-demand color 10 dynamic pcep
  ```

Configure Dynamic Path Optimization Objectives

- Use the **metric type** {**igp** | **te** | **latency**} command to configure the metric for use in path computation.

  ```
  Router(config-sr-te-color-dyn)# metric type te
  ```

- Use the **metric margin** {**absolute** *value*| **relative** *percent*} command to configure the On-Demand dynamic path metric margin. The range for *value* and *percent* is from 0 to 2147483647.

  ```
  Router(config-sr-te-color-dyn)# metric margin absolute 5
  ```

Configure Dynamic Path Constraints

- Use the **disjoint-path group-id** *group-id* **type** {**link** | **node** | **srlg** | **srlg-node**} [**sub-id** *sub-id*] command to configure the disjoint-path constraints. The *group-id* and *sub-id* range is from 1 to 65535.

  ```
  Router(config-sr-te-color-dyn)# disjoint-path group-id 775 type link
  ```

- Use the **affinity** {**include-any** | **include-all** | **exclude-any**} {**name** *WORD*} command to configure the affinity constraints.

  ```
  Router(config-sr-te-color-dyn)# affinity exclude-any name CROSS
  ```

- Use the **constraints segments sid-algorithm** *algorithm-number* command to configure the SR Flexible Algorithm constraints. The *algorithm-number* range is from 128 to 255.

  ```
  Router(config-sr-te-color)# constraints segments sid-algorithm 128
  ```

Configuring SR-ODN: Examples

Configuring SR-ODN: Layer-3 Services Examples

Table 42: Feature History Table

Feature Name	Release Information	Feature Description
SR-ODN: Layer 3 IPv6 BGP Services	Release 7.9.1	Segment Routing On-Demand Next Hop (SR-ODN) allows a service head-end router to automatically instantiate an SR policy to a BGP next-hop when required (on-demand). This feature introduces support for Layer 3 IPv6 BGP services (VPNv4/VPNv6) over SR-ODN.

The following examples show end-to-end configurations used in implementing SR-ODN on the head-end router.

Configuring ODN Color Templates: Example

Configure ODN color templates on routers acting as SR-TE head-end nodes. The following example shows various ODN color templates:

- color 10: minimization objective = te-metric
- color 20: minimization objective = igp-metric
- color 21: minimization objective = igp-metric; constraints = affinity
- color 22: minimization objective = te-metric; path computation at SR-PCE; constraints = affinity
- color 30: minimization objective = delay-metric
- color 128: constraints = flex-algo

```
segment-routing
 traffic-eng
  on-demand color 10
   dynamic
    metric
     type te
    !
   !
  !
  on-demand color 20
   dynamic
    metric
     type igp
    !
   !
  !
  on-demand color 21
   dynamic
    metric
     type igp
```

```
    !
    affinity exclude-any
     name CROSS
    !
   !
  !
  on-demand color 22
   dynamic
    pcep
    !
    metric
     type te
    !
    affinity exclude-any
     name CROSS
    !
   !
  !
  on-demand color 30
   dynamic
    metric
     type latency
    !
   !
  !
  on-demand color 128
   dynamic
    sid-algorithm 128
   !
  !
!
end
```

Configuring BGP Color Extended Community Set: Example

The following example shows how to configure BGP color extended communities that are later applied to BGP service routes via route-policies.

Note In most common scenarios, egress PE routers that advertise BGP service routes apply (set) BGP color extended communities. However, color can also be set at the ingress PE router.

```
extcommunity-set opaque color10-te
  10
end-set
!
extcommunity-set opaque color20-igp
  20
end-set
!
extcommunity-set opaque color21-igp-excl-cross
  21
end-set
!
extcommunity-set opaque color30-delay
  30
end-set
!
extcommunity-set opaque color128-fa128
  128
```

```
end-set
!
```

Configuring RPL to Set BGP Color (Layer-3 Services): Examples

The following example shows various representative RPL definitions that set BGP color community.

The first four RPL examples include the set color action only. The last RPL example performs the set color action for selected destinations based on a prefix-set.

```
route-policy SET_COLOR_LOW_LATENCY_TE
  set extcommunity color color10-te
  pass
end-policy
!
route-policy SET_COLOR_HI_BW
  set extcommunity color color20-igp
  pass
end-policy
!
route-policy SET_COLOR_LOW_LATENCY
  set extcommunity color color30-delay
  pass
end-policy
!
route-policy SET_COLOR_FA_128
  set extcommunity color color128-fa128
  pass
end-policy
!

prefix-set sample-set
  192.68.0.0/24
end-set
!
route-policy SET_COLOR_GLOBAL
  if destination in sample-set then
    set extcommunity color color10-te
  else
    pass
  endif
end-policy
```

Applying RPL to BGP Services (Layer-3 Services): Example

The following example shows various RPLs that set BGP color community being applied to BGP Layer-3 VPN services (VPNv4/VPNv6) and BGP global.

- The L3VPN examples show the RPL applied at the VRF export attach-point.
- The BGP global example shows the RPL applied at the BGP neighbor-out attach-point.

```
vrf vrf_cust1
 address-family ipv4 unicast
  export route-policy SET_COLOR_LOW_LATENCY_TE
 !
 address-family ipv6 unicast
  export route-policy SET_COLOR_LOW_LATENCY_TE
 !
!
vrf vrf_cust2
 address-family ipv4 unicast
  export route-policy SET_COLOR_HI_BW
```

```
 !
 address-family ipv6 unicast
  export route-policy SET_COLOR_HI_BW
 !
!
vrf vrf_cust3
 address-family ipv4 unicast
  export route-policy SET_COLOR_LOW_LATENCY
 !
 address-family ipv6 unicast
  export route-policy SET_COLOR_LOW_LATENCY
 !
!

vrf vrf_cust4
 address-family ipv4 unicast
  export route-policy SET_COLOR_FA_128
 !
 address-family ipv6 unicast
  export route-policy SET_COLOR_FA_128
 !
!
router bgp 100
 neighbor-group BR-TO-RR
  address-family ipv4 unicast
   route-policy SET_COLOR_GLOBAL out
  !
 !
!
end
```

L3VPN IPv4 Services: Verifying BGP VRF Information

Use the **show bgp vrf** command to display BGP prefix information for VRF instances. The following output shows the BGP VRF table including a prefix (88.1.1.0/24) with color 10 advertised by router 1.1.1.8.

```
Router# show bgp vrf vrf_cust1

BGP VRF vrf_cust1, state: Active
BGP Route Distinguisher: 1.1.1.4:101
VRF ID: 0x60000007
BGP router identifier 1.1.1.4, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000007   RD version: 282
BGP main routing table version 287
BGP NSR Initial initsync version 31 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.4:101 (default for vrf vrf_cust1)
*> 44.1.1.0/24        40.4.101.11                            0 400 {1} i
*>i55.1.1.0/24        1.1.1.5                      100       0 500 {1} i
*>i88.1.1.0/24        1.1.1.8 C:10                 100       0 800 {1} i
*>i99.1.1.0/24        1.1.1.9                      100       0 800 {1} i

Processed 4 prefixes, 4 paths
```

The following output displays the details for prefix 88.1.1.0/24. Note the presence of BGP extended color community 10, and that the prefix is associated with an SR policy with color 10 and BSID value of 24036.

```
Router# show bgp vrf vrf_cust1 88.1.1.0/24

BGP routing table entry for 88.1.1.0/24, Route Distinguisher: 1.1.1.4:101
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               282         282
Last Modified: May 20 09:23:34.112 for 00:06:03
Paths: (1 available, best #1)
  Advertised to CE peers (in unique update groups):
    40.4.101.11
  Path #1: Received by speaker 0
  Advertised to CE peers (in unique update groups):
    40.4.101.11
  800 {1}
    1.1.1.8 C:10 (bsid:24036) (metric 20) from 1.1.1.55 (1.1.1.8)
      Received Label 24012
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported
      Received Path ID 0, Local Path ID 1, version 273
      Extended community: Color:10 RT:100:1
      Originator: 1.1.1.8, Cluster list: 1.1.1.55
      SR policy color 10, up, registered, bsid 24036, if-handle 0x08000024

     Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1.1.1.8:101
```

L3VPN IPv4 Services: Verifying Forwarding (CEF) Table

Use the **show cef vrf** command to display the contents of the CEF table for the VRF instance. Note that prefix 198.51.100.1/24 points to the BSID label corresponding to an SR policy. Other non-colored prefixes, such as 192.0.2.2/24, point to BGP next-hop.

```
Router# show cef vrf vrf_cust1

Prefix              Next Hop            Interface
------------------- ------------------- ------------------
0.0.0.0/0           drop                default handler
0.0.0.0/32          broadcast
10.0.0.1/8          attached            TenGigE0/0/0/0.101
172.16.0.1/12       broadcast           TenGigE0/0/0/0.101
172.16.0.2/12       receive             TenGigE0/0/0/0.101
172.16.0.3/12       172.16.0.3/12       TenGigE0/0/0/0.101
172.16.0.4/12       broadcast           TenGigE0/0/0/0.101
192.168.0.1/16      172.16.0.3/12       <recursive>
192.0.2.2/24        10.0.0.2/8          <recursive>
198.51.100.1/24     24036 (via-label)   <recursive>
```

The following output displays CEF details for prefix 198.51.100.1/24. Note that the prefix is associated with an SR policy with BSID value of 24036.

```
Router# show cef vrf vrf_cust1 198.51.100.1/24

198.51.100.1/24, version 51, internal 0x5000001 0x0 (ptr 0x98c60ddc) [1], 0x0 (0x0), 0x208
 (0x98425268)
 Updated May 20 09:23:34.216
 Prefix Len 24, traffic index 0, precedence n/a, priority 3
   via local-label 24036, 5 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x97091ec0 0x0]
    recursion-via-label
    next hop VRF - 'default', table - 0xe0000000
    next hop via 24036/0/21
```

```
     next hop srte_c_10_ep labels imposed {ImplNull 24012}
```

L3VPN IPv4 Services: Verifying SR Policy

Use the **show segment-routing traffic-eng policy** command to display SR policy information.

The following outputs show the details of an on-demand SR policy that was triggered by prefixes with color 10 advertised by node 10.0.0.8.

```
Router# show segment-routing traffic-eng policy color 10 tabular

 Color             Endpoint  Admin   Oper              Binding
                             State  State                 SID
------ -------------------- ------ ------ --------------------
    10              10.0.0.8     up     up                24036
```

The following outputs show the details of the on-demand SR policy for BSID 24036.

Note There are 2 candidate paths associated with this SR policy: the path that is computed by the head-end router (with preference 200), and the path that is computed by the SR-PCE (with preference 100). The candidate path with the highest preference is the active candidate path (highlighted below) and is installed in forwarding.

```
Router# show segment-routing traffic-eng policy binding-sid 24036

SR-TE policy database
---------------------

Color: 10, End-point: 10.0.0.8
  Name: srte_c_10_ep_10.0.0.8
  Status:
    Admin: up  Operational: up for 4d14h (since Jul  3 20:28:57.840)
  Candidate-paths:
    Preference: 200 (BGP ODN) (active)
      Requested BSID: dynamic
      PCC info:
        Symbolic name: bgp_c_10_ep_10.0.0.8_discr_200
        PLSP-ID: 12
      Dynamic (valid)
        Metric Type: TE,   Path Accumulated Metric: 30
            16009 [Prefix-SID, 10.0.0.9]
            16008 [Prefix-SID, 10.0.0.8]
    Preference: 100 (BGP ODN)
      Requested BSID: dynamic
      PCC info:
        Symbolic name: bgp_c_10_ep_10.0.0.8_discr_100
        PLSP-ID: 11
      Dynamic (pce 10.0.0.57) (valid)
        Metric Type: TE,   Path Accumulated Metric: 30
            16009 [Prefix-SID, 10.0.0.9]
            16008 [Prefix-SID, 10.0.0.8]
  Attributes:
    Binding SID: 24036
    Forward Class: 0
    Steering BGP disabled: no
    IPv6 caps enable: yes
```

L3VPN IPv4 Services: Verifying SR Policy Forwarding

Use the **show segment-routing traffic-eng forwarding policy** command to display the SR policy forwarding information.

The following outputs show the forwarding details for an on-demand SR policy that was triggered by prefixes with color 10 advertised by node 10.0.0.8.

```
Router# show segment-routing traffic-eng forwarding policy binding-sid 24036 tabular

Color Endpoint        Segment      Outgoing Outgoing     Next Hop     Bytes        Pure
                      List         Label    Interface                Switched     Backup
----- --------------- ------------ -------- ------------ ------------ ------------ ------
10    10.0.0.8        dynamic      16009    Gi0/0/0/4    10.4.5.5     0
                                   16001    Gi0/0/0/5    10.5.8.8     0            Yes

Router# show segment-routing traffic-eng forwarding policy binding-sid 24036 detail
Mon Jul  8 11:56:46.887 PST

SR-TE Policy Forwarding database
--------------------------------

Color: 10, End-point: 10.0.0.8
  Name: srte_c_10_ep_10.0.0.8
  Binding SID: 24036
  Segment Lists:
    SL[0]:
      Name: dynamic
      Paths:
        Path[0]:
          Outgoing Label: 16009
          Outgoing Interface: GigabitEthernet0/0/0/4
          Next Hop: 10.4.5.5
          Switched Packets/Bytes: 0/0
          FRR Pure Backup: No
          Label Stack (Top -> Bottom): { 16009, 16008 }
          Path-id: 1 (Protected), Backup-path-id: 2, Weight: 64
        Path[1]:
          Outgoing Label: 16001
          Outgoing Interface: GigabitEthernet0/0/0/5
          Next Hop: 10.5.8.8
          Switched Packets/Bytes: 0/0
          FRR Pure Backup: Yes
          Label Stack (Top -> Bottom): { 16001, 16009, 16008 }
          Path-id: 2 (Pure-Backup), Weight: 64
  Policy Packets/Bytes Switched: 0/0
  Local label: 80013
```

L3VPN IPv6 Services: Verifying BGP VRF Information

Use the **show bgp vrf** command to display BGP prefix information for VRF instances.

The following output displays the details for prefix 2020:0:0:1::. Note the presence of BGP extended color community 10, and that the prefix is associated with an SR policy with color 10 and BSID value of 51006.

```
Router# show bgp vrf vrf_cust1 ipv6 unicast 2020:0:0:1::

BGP routing table entry for 2020:0:0:1::/64, Route Distinguisher: 100:1001
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               66662        66662
Last Modified: Mar 16 09:18:40.678 for 00:01:36
Paths: (1 available, best #1)
```

```
  Advertised to CE peers (in unique update groups):
    2002::6701:b02
  Path #1: Received by speaker 0
  Advertised to CE peers (in unique update groups):
    2002::6701:b02
  2001
    100.2.1.1 C:10 (bsid:51006) (metric 40) from 100.2.1.1 (100.2.1.1)
      Received Label 51000
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported
      Received Path ID 1, Local Path ID 1, version 49971
      Extended community: Color[CO-Flag]:10[10] RT:100:1
      SR policy color 10, up, registered, bsid 51006, if-handle 0x0f000c9c

     Source AFI: VPNv6 Unicast, Source VRF: default, Source Route Distinguisher: 100:2001[0m
```

L3VPN IPv6 Services: Verifying Forwarding (CEF) Table

Use the **show cef vrf** command to display the contents of the CEF table for the VRF instance.

The following output displays CEF details for prefix 2020:0:0:1::/64. Note that the prefix is associated with an SR policy with BSID value of 51006.

```
Router# show cef vrf vrf_cust1 ipv6 2020:0:0:1:: detail

2020:0:0:1::/64, version 229, internal 0x5000001 0x30 (ptr 0xba4b9050) [1], 0x0 (0x0), 0x208
 (0x9a6e3858)
 Updated Mar 16 09:18:41.170
 Prefix Len 64, traffic index 0, precedence n/a, priority 3
  gateway array (0xbc2fe5d0) reference count 16, flags 0x2038, source rib (7), 0 backups
                [1 type 1 flags 0x48441 (0xbe849048) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Mar 16 09:18:41.170
 LDI Update time Mar 16 09:18:41.170
   via local-label 51006, 3 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x9a86d0e0 0x0]
    recursion-via-label
    next hop VRF - 'default', table - 0xe0000000
    next hop via 51006/0/21
     labels imposed {51000}

    Load distribution: 0 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   recursive                 51006/0        [0m
```

L3VPN IPv6 Services: Verifying SR Policy

Use the **show segment-routing traffic-eng policy** command to display SR policy information.

```
Router# show segment-routing traffic-eng policy endpoint ipv4 100.2.1.1 color 10

SR-TE policy database
---------------------

Color: 10, End-point: 100.2.1.1
  Name: srte_c_10_ep_100.2.1.1
  Status:
    Admin: up  Operational: up for 00:01:21 (since Mar 16 09:18:41.096)
  Candidate-paths:
    Preference: 200 (BGP ODN) (active)
```

```
    Requested BSID: dynamic
    Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 8
    Dynamic (valid)
      Metric Type: TE,   Path Accumulated Metric: 30
        SID[0]: 51002 [Adjacency-SID, 101.1.3.1 - 101.1.3.2]
        SID[1]: 41304 [Adjacency-SID, 101.3.4.1 - 101.3.4.2]
        SID[2]: 41400 [Adjacency-SID, 101.2.4.2 - 101.2.4.1]
  Preference: 100 (BGP ODN) (inactive)
    Requested BSID: dynamic
    PCC info:
      Symbolic name: bgp_c_10_ep_100.2.1.1_discr_100
      PLSP-ID: 3
    Constraints:
      Protection Type: protected-preferred
      Maximum SID Depth: 8
    Dynamic (pce) (inactive)
      Metric Type: NONE,   Path Accumulated Metric: 0
Attributes:
  Binding SID: 51006
  Forward Class: Not Configured
  Steering labeled-services disabled: no
  Steering BGP disabled: no
  IPv6 caps enable: yes
  Invalidation drop enabled: no
  Max Install Standby Candidate Paths: 0
```

Manually Provisioned SR Policy

Manually provisioned SR policies are configured on the head-end router. These policies can use dynamic paths or explicit paths. See the SR-TE Policy Path Types, on page 367 section for information on manually provisioning an SR policy using dynamic or explicit paths.

PCE-Initiated SR Policy

An SR-TE policy can be configured on the path computation element (PCE) to reduce link congestion or to minimize the number of network touch points.

The PCE collects network information, such as traffic demand and link utilization. When the PCE determines that a link is congested, it identifies one or more flows that are causing the congestion. The PCE finds a suitable path and deploys an SR-TE policy to divert those flows, without moving the congestion to another part of the network. When there is no more link congestion, the policy is removed.

To minimize the number of network touch points, an application, such as a Network Services Orchestrator (NSO), can request the PCE to create an SR-TE policy. PCE deploys the SR-TE policy using PCC-PCE communication protocol (PCEP).

For more information, see the PCE-initiated SR Policies for Traffic Management, on page 519 section in the *Configure Segment Routing Path Computation Element* chapter.

Cumulative Metric Bounds (Delay-Bound Use-Case)

SRTE can calculate a shortest path with cumulative metric bounds. For example, consider these metric bounds:

- IGP metric <= 10

- TE metric <= 60
- Hop count <= 4
- Latency <= 55

When an SR policy is configured on a head-end node with these metric bounds, a path is finalized towards the specified destination only if it meets each of these criteria.

You can set the maximum number of attempts for computing a shortest path that satisfies the cumulative metric bounds criteria, by using the **kshortest-paths** command in SR-TE configuration mode.

Restrictions

- PCE-based cumulative metric bounds computations are not supported. You must use non-PCE (SR-TE topology) based configuration for path calculation, for cumulative bounds.
- If you use PCE dynamic computation configuration with cumulative bounds, the PCE computes a path and validates against cumulative bounds. If it is valid, then the policy is created with this path on PCC. If the initial path doesn't respect the bounds, then the path is not considered, and no further K-shortest path algorithm is executed to find the path.

Configuring SRTE Shortest Path Calculation For Cumulative Metric Bounds

You can enable this feature for SR, and ODN SR policy configurations, as shown below.

SR Policy

SR Policy - A policy called **fromAtoB_XTC** is created towards destination IP address 192.168.0.2. Also, the candidate-paths preference, and other attributes are enabled.

```
Router# configure terminal
Router(config)# segment-routing traffic-eng policy fromAtoB_XTC
Router(config-sr-te-policy)# color 2 end-point ipv4 192.168.0.2
Router(config-sr-te-policy)# candidate-paths preference 100
Router(config-sr-te-policy-path-pref)# dynamic metric type te
```

Cumulative Metric bounds – IGP, TE, hop count, and latency metric bounds are set. SRTE calculates paths only when each criterion is satisfied.

```
Router(config-sr-te-policy-path-pref)# constraints bounds cumulative
Router(config-sr-te-pref-const-bounds-type)# type igp 10
Router(config-sr-te-pref-const-bounds-type)# type te 60
Router(config-sr-te-pref-const-bounds-type)# type hopcount 4
Router(config-sr-te-pref-const-bounds-type)# type latency 55
Router(config-sr-te-pref-const-bounds-type)# commit
```

ODN SR Policy

SR ODN Policy – An SR ODN policy with color 1000 is created. Also, the candidate-paths value is on-demand.

```
Router# configure terminal
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# on-demand color 1000 dynamic metric type te
Router(config-sr-te)# candidate-paths on-demand
Router(config-sr-te-candidate-path-type)# exit
Router(config-sr-te-candidate-path)# exit
```

Cumulative Metric bounds – IGP, TE, hop count, and latency metric bounds are set for the policy. SRTE calculates paths, only when each criterion is satisfied.

```
Router(config-sr-te)# on-demand color 1000 dynamic bounds cumulative
Router(config-sr-te-odc-bounds-type)# type igp 100
Router(config-sr-te-odc-bounds-type)# type te 60
Router(config-sr-te-odc-bounds-type)# type hopcount 6
Router(config-sr-te-odc-bounds-type)# type latency 1000
Router(config-sr-te-odc-bounds-type)# commit
```

To set the maximum number of attempts for computing paths that satisfy the cumulative metric bounds criteria, use the **kshortest-paths** command.

```
Router# configure terminal
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# kshortest-paths 120
Router(config-sr-te)# commit
```

Verification

Use this command to view SR policy configuration details. Pointers:

- The **Number of K-shortest-paths** field displays 4. It means that the K-shortest path algorithm took 4 computations to find the right path. The 4 shortest paths that are computed using K-shortest path algorithm did not respect the cumulative bounds. The fifth shortest path is valid against the bounds.
- The values for the metrics of the actual path (**TE**, **IGP**, **Cumulative Latency** and **Hop count** values in the **Dynamic** section) are within the configured cumulative metric bounds.

```
Router# show segment-routing traffic-eng policy color 2

Color: 2, End-point: 192.168.0.2
  Name: srte_c_2_ep_192.168.0.2
  Status:
    Admin: up  Operational: up for 3d02h (since Dec 15 12:13:21.993)

  Candidate-paths:

    Preference: 100 (configuration) (active)

      Name: fromAtoB_XTC
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Affinity:
          exclude-any:
          red
        Maximum SID Depth: 10
        IGP Metric Bound: 10
        TE Metric Bound: 60
        Latency Metric Bound: 55
        Hopcount Metric Bound: 4

      Dynamic (valid)

        Metric Type: TE,   Path Accumulated Metric: 52
        Number of K-shortest-paths: 4
        TE Cumulative Metric: 52
        IGP Cumulative Metric: 3
        Cumulative Latency: 52
        Hop count: 3
          16004 [Prefix-SID, 192.168.0.4]
          24003 [Adjacency-SID, 10.16.16.2 - 10.16.16.5]
```

```
        24001 [Adjacency-SID, 10.14.14.5 - 10.14.14.4]

Attributes:

  Binding SID: 24011
  Forward Class: Not Configured
  Steering labeled-services disabled: no
  Steering BGP disabled: no
  IPv6 caps enable: yes
  Invalidation drop enabled: no
```

SR-TE Policy Path Types

A **dynamic** path is based on an optimization objective and a set of constraints. The head-end computes a solution, resulting in a SID-list or a set of SID-lists. When the topology changes, a new path is computed. If the head-end does not have enough information about the topology, the head-end might delegate the computation to a Segment Routing Path Computation Element (SR-PCE).

An **explicit** path is a specified SID-list or set of SID-lists.

An SR-TE policy initiates a single (selected) path in RIB/FIB. This is the preferred valid candidate path. A path is selected when the path is valid and its preference is the best among all candidate paths for that policy.

Note The protocol of the source is not relevant in the path selection logic.

A candidate path has the following characteristics:

- It has a preference – If two policies have the same {color, endpoint} but different preferences, the policy with the highest preference is selected.
- It is associated with a single binding SID (BSID) – A BSID conflict occurs when there are different SR policies with the same BSID. In this case, the policy that is installed first gets the BSID and is selected.
- It is valid if it is usable.

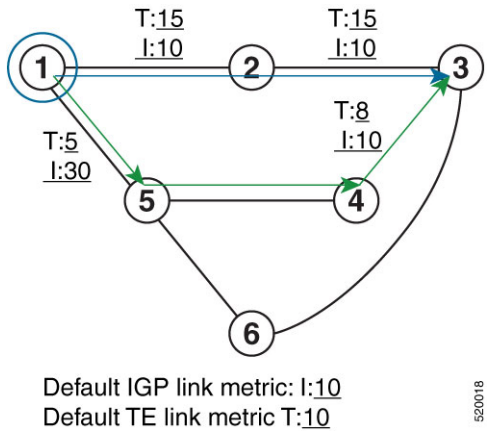
Dynamic Paths

Optimization Objectives

Optimization objectives allow the head-end router to compute a SID-list that expresses the shortest dynamic path according to the selected metric type:

- IGP metric — Refer to the "Implementing IS-IS" and "Implementing OSPF" chapters in the *Routing Configuration Guide for Cisco 8000 Series Routers*.
- TE metric — See the Configure Interface TE Metrics, on page 368 section for information about configuring TE metrics.
- Delay — See the Configure Performance Measurement chapter for information about measuring delay for links or SR policies.

This example shows a dynamic path from head-end router 1 to end-point router 3 that minimizes IGP or TE metric:



- The blue path uses the minimum IGP metric: Min-Metric (1 → 3, IGP) = SID-list <16003>; cumulative IGP metric: 20
- The green path uses the minimum TE metric: Min-Metric (1 → 3, TE) = SID-list <16005, 16004, 16003>; cumulative TE metric: 23

Configure Interface TE Metrics

Use the **metric** *value* command in SR-TE interface submode to configure the TE metric for interfaces. The *value* range is from 0 to 2147483647.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# interface type interface-path-id
Router(config-sr-te-if)# metric value
```

Configuring TE Metric: Example

The following configuration example shows how to set the TE metric for various interfaces:

```
segment-routing
 traffic-eng
  interface TenGigE0/0/0/0
   metric 100
  !
  interface TenGigE0/0/0/1
   metric 1000
  !
  interface TenGigE0/0/2/0
   metric 50
  !
 !
end
```

Constraints

Constraints allow the head-end router to compute a dynamic path according to the selected metric type:

- TE affinity — You can apply a color or name to links or interfaces by assigning affinity bit-maps to them. You can then specify an affinity (or relationship) between an SR policy path and link colors. SR-TE computes a path that includes or excludes links that have specific colors,or combinations of colors. See the Named Interface Link Admin Groups and SR-TE Affinity Maps, on page 369 section for information on named interface link admin groups and SR-TE Affinity Maps.
- Disjoint — SR-TE computes a path that is disjoint from another path in the same disjoint-group. Disjoint paths do not share network resources. Path disjointness may be required for paths between the same pair of nodes, between different pairs of nodes, or a combination (only same head-end or only same end-point).
- Flexible Algorithm — Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraint.

Named Interface Link Admin Groups and SR-TE Affinity Maps

Named Interface Link Admin Groups and SR-TE Affinity Maps provide a simplified and more flexible means of configuring link attributes and path affinities to compute paths for SR-TE policies.

In the traditional TE scheme, links are configured with attribute-flags that are flooded with TE link-state parameters using Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF).

Named Interface Link Admin Groups and SR-TE Affinity Maps let you assign, or map, up to 32256 color names for affinity and attribute-flag attributes instead of 32-bit hexadecimal numbers. After mappings are defined, the attributes can be referred to by the corresponding color name in the CLI. Furthermore, you can define constraints using *include-any*, *include-all*, and *exclude-any* arguments, where each statement can contain up to 10 colors.

Note You can configure affinity constraints using attribute flags or the Flexible Name Based Policy Constraints scheme; however, when configurations for both schemes exist, only the configuration pertaining to the new scheme is applied.

Configure Named Interface Link Admin Groups and SR-TE Affinity Maps

Use the **affinity name** *NAME* command in SR-TE interface submode to assign affinity to interfaces. Configure this on routers with interfaces that have an associated admin group attribute.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# interface TenGigE0/0/1/2
Router(config-sr-if)# affinity
Router(config-sr-if-affinity)# name RED
```

Use the **affinity-map name** *NAME* **bit-position** *bit-position* command in SR-TE sub-mode to define affinity maps. The *bit-position* range is from 0 to 255.

Configure affinity maps on the following routers:

- Routers with interfaces that have an associated admin group attribute.
- Routers that act as SR-TE head-ends for SR policies that include affinity constraints.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
```

```
Router(config-sr-te)# affinity-map
Router(config-sr-te-affinity-map)# name RED bit-position 23
```

Configuring Link Admin Group: Example

The following example shows how to assign affinity to interfaces and to define affinity maps. This configuration is applicable to any router (SR-TE head-end or transit node) with colored interfaces.

```
segment-routing
 traffic-eng
  interface TenGigE0/0/1/1
   affinity
    name CROSS
    name RED
   !
  !
  interface TenGigE0/0/1/2
   affinity
    name RED
   !
  !
  interface TenGigE0/0/2/0
   affinity
    name BLUE
   !
  !
  affinity-map
   name RED bit-position 23
   name BLUE bit-position 24
   name CROSS bit-position 25
  !
end
```

Configure SR Policy with Dynamic Path

To configure a SR-TE policy with a dynamic path, optimization objectives, and affinity constraints, complete the following configurations:

1. Define the optimization objectives. See the Optimization Objectives, on page 367 section.
2. Define the constraints. See the Constraints, on page 368 section.
3. Create the policy.

The following example shows a configuration of an SR policy at an SR-TE head-end router. The policy has a dynamic path with optimization objectives and affinity constraints computed by the head-end router.

```
segment-routing
 traffic-eng
  policy foo
   color 100 end-point ipv4 10.1.1.2
   candidate-paths
    preference 100
     dynamic
      metric
       type te
      !
     !
     constraints
      affinity
```

```
      exclude-any
       name RED
      !
     !
    !
   !
  !
 !
```

The following example shows a configuration of an SR policy at an SR-TE head-end router. The policy has a dynamic path with optimization objectives and affinity constraints computed by the SR-PCE.

```
segment-routing
 traffic-eng
  policy baa
   color 101 end-point ipv4 10.1.1.2
   candidate-paths
    preference 100
     dynamic
      pcep
      !
      metric
       type te
      !
     !
     constraints
      affinity
       exclude-any
        name BLUE
       !
      !
     !
    !
   !
  !
```

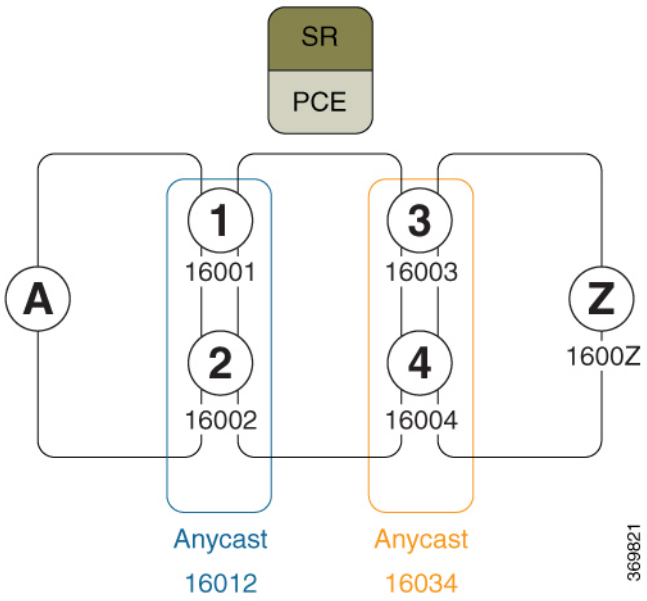
Anycast SID-Aware Path Computation

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

Note For information on configuring Anycast SID, see Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface, on page 252 and Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface, on page 285.

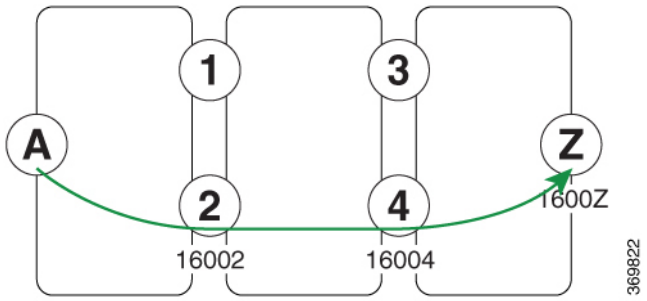
This example shows how Anycast SIDs are inserted into a computed SID list.

The following figure shows 3 isolated IGP domains without redistribution and without BGP 3107. Each Area Border Router (ABR) 1 through 4 is configured with a node SID. ABRs 1 and 2 share Anycast SID 16012 and ABRs 3 and 4 share Anycast SID 16034.



Consider the case where routers A and Z are provider edge (PE) routers in the same VPN. Router A receives a VPN route with BGP next-hop to router Z. Router A resolves the SR path to router Z using SR-ODN or SR-PCE.

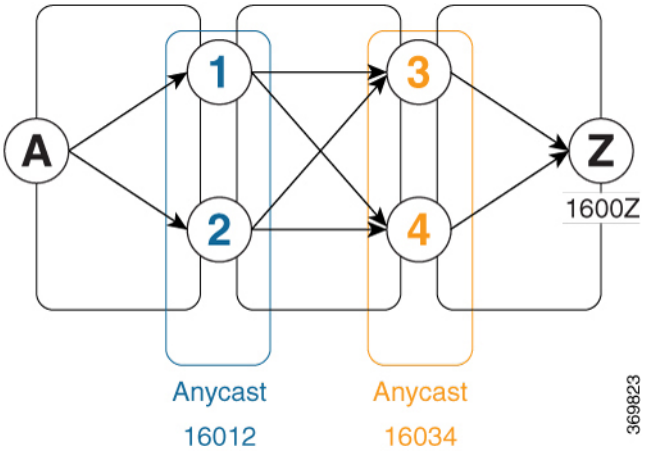
Before considering Anycast SIDs, the head-end router or SR-PCE computes the SID list.



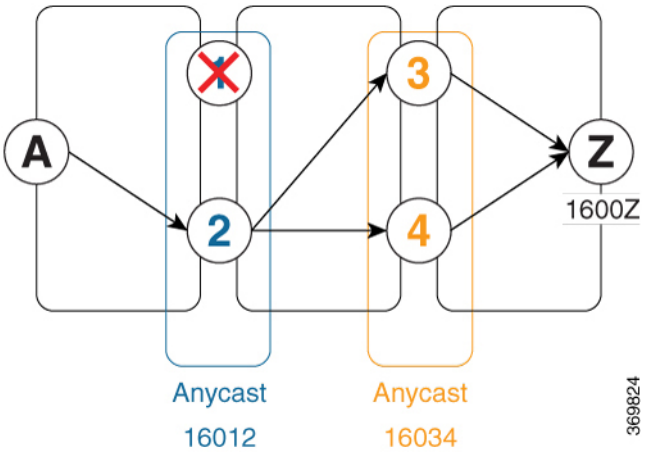
In this case, the optimized computed path from router A to router Z is 16002 > 16004 > 1600Z

The path computation process reiterates the original SID-list and replaces node SIDs with Anycast SIDs (when possible). SR-TE verifies that the Anycast-encoded SID list maintains an optimum path and does not violate any path constraints (link affinity, metric bounds). If the SID list is verified, then the Anycast-encoded SID list is signaled and instantiated in the forwarding.

Using the Anycast-encoded SID list, the optimized computed path from router A to router Z is 16012 > 16034 > 1600Z. The Anycast SID-aware path computation provides load-balancing.



The Anycast SID aware path computation also provides resiliency. For example, if one of the ABRs (in this case, ABR 1) becomes unavailable or unreachable, the path from router A to router Z (16012 > 16034 > 1600Z) will still be valid and usable.



Configuration Examples

1. Configure Prefix SIDs on the ABR nodes.
   a. Configure each node with a node SID.
   b. Configure each group of nodes with a shared Anycast SID.

   See Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface, on page 252 and Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface, on page 285.

2. Configure SR policies to include Anycast SIDs for path computation using the **anycast-sid-inclusion** command.

   This example shows how to configure a local SR policy to include Anycast SIDs for PCC-initiated path computation at the head-end router:

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# policy FOO
Router(config-sr-te-policy)# color 10 end-point ipv4 10.1.1.10
Router(config-sr-te-policy)# candidate-paths
```

```
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# dynamic
Router(config-sr-te-pp-info)# anycast-sid-inclusion
```

Running Configuration

Use the **anycast-sid-inclusion** command to include Anycast SIDs into the computed paths of the following policy types:

- Local SR policy with PCC-initiated path computation at the head-end router:

```
segment-routing
  traffic-eng
    policy FOO
      color 10 end-point ipv4 10.1.1.10
      candidate-paths
        preference 100
          dynamic
            anycast-sid-inclusion
```

- Local SR policy with PCC-initiated/PCE-delegated path computation at the SR-PCE:

```
segment-routing
 traffic-eng
  policy BAR
   color 20 end-point ipv4 10.1.1.20
    candidate-paths
     preference 100
      dynamic
       pcep
       anycast-sid-inclusion
```

- On-demand SR policies with a locally computed dynamic path at the head-end, or centrally computed dynamic path at the SR-PCE:

```
segment-routing
  traffic-eng
   on-demand color 10
    dynamic
     anycast-sid-inclusion
```

- On-demand SR policies with centrally computed dynamic path at the SR-PCE:

```
segment-routing
 traffic-eng
  on-demand color 20
   dynamic
    pcep
    anycast-sid-inclusion
```

Explicit Path with Affinity Constraint Validation for Anycast SIDs

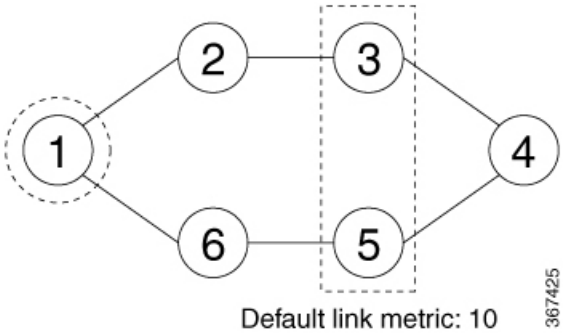
Note For information about configuring Anycast SIDs, see Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface, on page 252 or Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface, on page 285.

Routers that are configured with the same Anycast SID, on the same Loopback address and with the same SRGB, advertise the same prefix SID (Anycast).

The shortest path with the lowest IGP metric is then verified against the affinity constraints. If multiple nodes have the same shortest-path metric, all their paths are validated against the affinity constraints. A path that is not the shortest path is not validated against the affinity constraints.

Affinity Support for Anycast SIDs: Examples

In the following examples, nodes 3 and 5 advertise the same Anycast prefix (10.1.1.8) and assign the same prefix SID (16100).

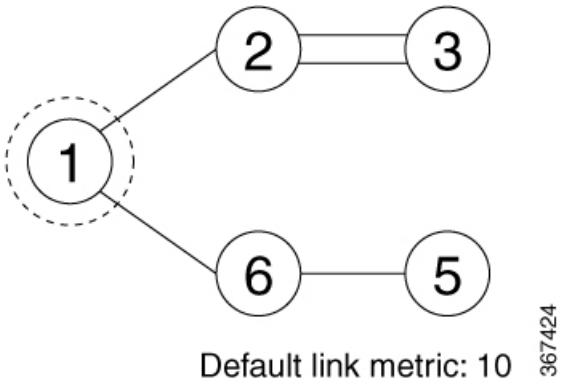


Node 1 uses the following SR-TE policy:

```
segment-routing
 traffic-eng
  policy POLICY1
   color 20 end-point ipv4 10.1.1.4
   binding-sid mpls 1000
   candidate-paths
    preference 100
     explicit segment-list SIDLIST1
     constraints
      affinity
       exclude-any
        red
  segment-list name SIDLIST1
   index 10 address ipv4 192.68.100.100
   index 20 address ipv4 10.4.4.4
```

Affinity Constraint Validation With ECMP Anycast SID: Example

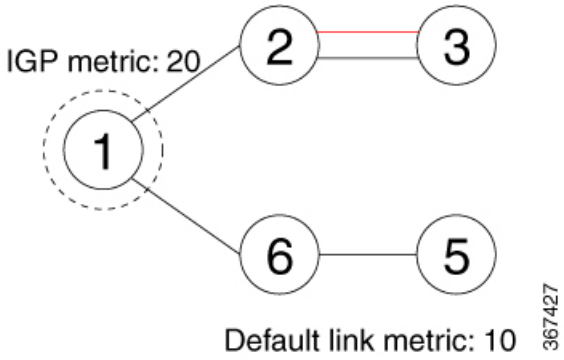
In this example, the shortest path to both node 3 and node 5 has an equal accumulative IGP metric of 20. Both paths are validated against affinity constraints.




```
Name: POLICY1 (Color: 2, End-point: 198.51.100.6)
  Status:
    Admin: up  Operational: up for 00:03:52 (since Jan 24 01:52:14.215)
  Candidate-paths:
    Preference 100:
      Constraints:
        Affinity:
          exclude-any: red
      Explicit: segment-list SIDLIST1 (active)
        Weight: 0, Metric Type: IGP
          16100 [Prefix-SID, 10.1.1.8]
          16004 [Prefix-SID, 10.4.4.4]
```

Affinity Constraint Validation With Non-ECMP Anycast SID: Example

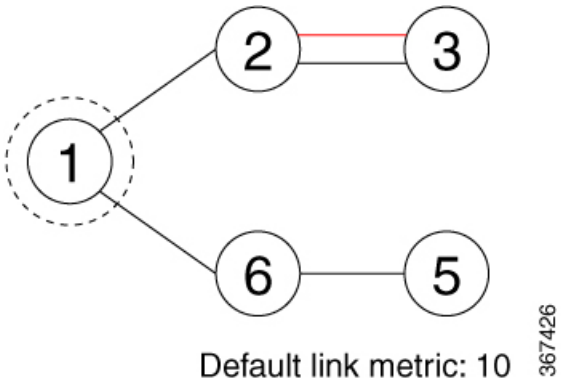
In this example, the shortest path to node 5 has an accumulative IGP metric of 20, and the shortest path to node 3 has an accumulative IGP metric of 30. Only the shortest path to node 5 is validated against affinity constraints.



Note Even though parallel link (23) is marked with red, it is still considered valid since anycast traffic flows only on the path to node 5.

Invalid Path Based on Affinity Constraint: Example

In this example, parallel link (23) is marked as red, so the path to anycast node 3 is invalidated.




```
SR-TE policy database
---------------------
Name: POLICY1 (Color: 2, End-point: 198.51.100.6)
  Status:
    Admin: up  Operational: up for 00:03:52 (since Jan 24 01:52:14.215)
  Candidate-paths:
    Preference 100:
      Constraints:
        Affinity:
          exclude-any: red
      Explicit: segment-list SIDLIST1 (inactive)
      Inactive Reason: Link [10.2.21.23,10.2.21.32] failed to satisfy affinity exclude-any
 constraint=0x00000008, link attributes=0x0000000A
```

Explicit Paths

SR-TE Policy with Explicit Path

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
SR-TE Explicit Segment Lists with Mix of IPv4 and IPv6 Segments	Release 7.9.1	This feature allows you to configure an explicit segment list with IPv4 addresses and include an IPv6 address as a non-first SID. This feature allows you to deploy a centralized BGP EPE solution for 6PE in an SR-MPLS network where the last segment is associated with a EPE-enabled BGPv6 neighbor.

An explicit segment list is defined as a sequence of one or more segments. A segment can be configured as an IP address or an MPLS label representing a node or a link.

An explicit segment list can be configured with the following:

- IP-defined segments

- MPLS label-defined segments
- A combination of IP-defined segments and MPLS label-defined segments

Behaviors and Limitations

- An IP-defined segment can be associated with an IPv4 or IPv6 address (for example, a link or a Loopback address).
- An IPv6 address cannot be the first segment of the segment list.
  - A segment defined with an IPv6 address (for example, IPv6 EPE SID) enables use-cases such as Use Case: Centralized BGP EPE for 6PE in an SR-MPLS Network where the last segment of the explicit segment list is associated with an EPE-enabled BGPv6 neighbor.
- The local MPLS label assigned to a learned BGP prefix SID can be configured as the first segment of a segment list.
- When a segment of the segment list is defined as an MPLS label, subsequent segments can only be configured as MPLS labels.

Configure Local SR-TE Policy Using Explicit Paths

To configure an SR-TE policy with an explicit path, complete the following configurations:

1. Create the segment list.
2. Create the SR-TE policy.

Create a segment list with IP addresses:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name SIDLIST1
Router(config-sr-te-sl)# index 10 mpls adjacency 10.1.1.2
Router(config-sr-te-sl)# index 20 mpls adjacency 10.1.1.3
Router(config-sr-te-sl)# index 30 mpls adjacency 10.1.1.4
Router(config-sr-te-sl)# exit
```

Create a segment list with MPLS labels:

```
Router(config-sr-te)# segment-list name SIDLIST2
Router(config-sr-te-sl)# index 10 mpls label 16002
Router(config-sr-te-sl)# index 20 mpls label 16003
Router(config-sr-te-sl)# index 30 mpls label 16004
Router(config-sr-te-sl)# exit
```

Create a segment list with IP addresses and MPLS labels:

```
Router(config-sr-te)# segment-list name SIDLIST3
Router(config-sr-te-sl)# index 10 mpls adjacency 10.1.1.2
Router(config-sr-te-sl)# index 20 mpls label 16003
Router(config-sr-te-sl)# index 30 mpls label 16004
Router(config-sr-te-sl)# exit
```

Create a segment list with IPv4 and IPv6 addresses:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name SIDLIST4
Router(config-sr-te-sl)# index 10 mpls adjacency 10.1.1.2
Router(config-sr-te-sl)# index 20 mpls adjacency 10.1.1.3
Router(config-sr-te-sl)# index 30 mpls adjacency 2001:db8:10:1:1::100
Router(config-sr-te-sl)# exit
```

Create the SR-TE policy:

```
Router(config-sr-te)# policy POLICY2
Router(config-sr-te-policy)# color 20 end-point ipv4 10.1.1.4
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST2
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# exit
Router(config-sr-te-policy-path)# preference 200
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST1
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST4
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# exit
```

Running Configuration

```
Router# show running-configuration
segment-routing
 traffic-eng
  segment-list SIDLIST1
   index 10 mpls adjacency 10.1.1.2
   index 20 mpls adjacency 10.1.1.3
   index 30 mpls adjacency 10.1.1.4
  !
  segment-list SIDLIST2
   index 10 mpls label 16002
   index 20 mpls label 16003
   index 30 mpls label 16004
  !
  segment-list SIDLIST3
   index 10 mpls adjacency 10.1.1.2
   index 20 mpls label 16003
   index 30 mpls label 16004
  !

  segment-list SIDLIST4
   index 10 mpls adjacency 10.1.1.2
   index 10 mpls adjacency 10.1.1.3
   index 10 mpls adjacency 2001:db8:10:1:1::100
  !

  policy POLICY2
   color 20 end-point ipv4 10.1.1.4
   candidate-paths
    preference 100
     explicit segment-list SIDLIST1
     !
    !
    preference 200
     explicit segment-list SIDLIST2
     !
     explicit segment-list SIDLIST4
```

```
     !
    !
   !
  !
 !
!
```

Verification

This feature provides for displaying detailed segment list information. This is in addition to the current behavior of displaying segment list information from active policies. For active candidate paths, the status of segment list will either be valid or invalid. If the segment list is invalid, the reason for its invalidity along with the entire label/IP stack of segment list is displayed. For inactive candidate paths, the status of segment list will always be inactive. Since the validity of segment list under inactive path is not checked, it is always displayed inactive.

Verify the SR-TE policy configuration using:

```
Router# show segment-routing traffic-eng policy name srte_c_20_ep_10.1.1.4

SR-TE policy database
---------------------

Color: 20, End-point: 10.1.1.4
  Name: srte_c_20_ep_10.1.1.4
  Status:
    Admin: up  Operational: up for 00:00:15 (since Jul 14 00:53:10.615)
  Candidate-paths:
    Preference: 200 (configuration) (active)
      Name: POLICY2
      Requested BSID: dynamic
        Protection Type: protected-preferred
        Maximum SID Depth: 8
      Explicit: segment-list SIDLIST2 (active)
        Weight: 1, Metric Type: TE
          16002
          16003
          16004

    Preference: 100 (configuration) (inactive)
      Name: POLICY2
      Requested BSID: dynamic
        Protection Type: protected-preferred
        Maximum SID Depth: 8
      Explicit: segment-list SIDLIST1 (inactive)
        Weight: 1, Metric Type: TE
          [Adjacency-SID, 10.1.1.2 - <None>]
          [Adjacency-SID, 10.1.1.3 - <None>]
          [Adjacency-SID, 10.1.1.4 - <None>]
    Attributes:
    Binding SID: 51301
   Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
```

Configuring Explicit Path with Affinity Constraint Validation

To fully configure SR-TE flexible name-based policy constraints, you must complete these high-level tasks in order:

1. Assign Color Names to Numeric Values
2. Associate Affinity-Names with SR-TE Links
3. Associate Affinity Constraints for SR-TE Policies

```
/* Enter the global configuration mode and assign color names to numeric values
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# affinity-map
Router(config-sr-te-affinity-map)# name blue bit-position 0
Router(config-sr-te-affinity-map)# name green bit-position 1
Router(config-sr-te-affinity-map)# name red bit-position 2
Router(config-sr-te-affinity-map)# exit


/* Associate affinity-names with SR-TE links
Router(config-sr-te)# interface Gi0/0/0/0
Router(config-sr-te-if)# affinity
Router(config-sr-te-if-affinity)# name blue
Router(config-sr-te-if-affinity)# exit
Router(config-sr-te-if)# exit
Router(config-sr-te)# interface Gi0/0/0/1
Router(config-sr-te-if)# affinity
Router(config-sr-te-if-affinity)# name blue
Router(config-sr-te-if-affinity)# name green
Router(config-sr-te-if-affinity)# exit
Router(config-sr-te-if)# exit
Router(config-sr-te)#


/* Associate affinity constraints for SR-TE policies
Router(config-sr-te)# segment-list name SIDLIST1
Router(config-sr-te-sl)# index 10 mpls adjacency 10.1.1.2
Router(config-sr-te-sl)# index 20 mpls adjacency 10.2.2.23
Router(config-sr-te-sl)# index 30 mpls adjacency 10.1.1.4
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list name SIDLIST2
Router(config-sr-te-sl)# index 10 mpls adjacency 10.1.1.2
Router(config-sr-te-sl)# index 30 mpls adjacency 10.1.1.4
Router(config-sr-te-sl)# exit
Router(config-sr-te)# segment-list name SIDLIST3
Router(config-sr-te-sl)# index 10 mpls adjacency 10.1.1.5
Router(config-sr-te-sl)# index 30 mpls adjacency 10.1.1.4
Router(config-sr-te-sl)# exit

Router(config-sr-te)# policy POLICY1
Router(config-sr-te-policy)# color 20 end-point ipv4 10.1.1.4
Router(config-sr-te-policy)# binding-sid mpls 1000
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 200
Router(config-sr-te-policy-path-pref)# constraints affinity exclude-any red
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST1
Router(config-sr-te-pp-info)# exit
```

```
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST2
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-pref)# exit
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list SIDLIST3
```

Running Configuration

```
Router# show running-configuration
segment-routing
 traffic-eng
  interface GigabitEthernet0/0/0/0
   affinity
    name blue
   !
  !
  interface GigabitEthernet0/0/0/1
   affinity
    name blue
    name green
   !
  !
  segment-list SIDLIST1
   index 10 mpls adjacency 10.1.1.2
   index 20 mpls adjacency 10.2.2.23
   index 30 mpls adjacency 10.1.1.4
  !
  segment-list SIDLIST2
   index 10 mpls adjacency 10.1.1.2
   index 30 mpls adjacency 10.1.1.4
  !
  segment-list SIDLIST3
   index 10 mpls adjacency 10.1.1.5
   index 30 mpls adjacency 10.1.1.4
  !
  policy POLICY1
   binding-sid mpls 1000
   color 20 end-point ipv4 10.1.1.4
   candidate-paths
    preference 100
     explicit segment-list SIDLIST3
     !
    !
    preference 200
     explicit segment-list SIDLIST1
     !
     explicit segment-list SIDLIST2
     !
     constraints
      affinity
       exclude-any
        name red
       !
      !
     !
    !
   !
  !
  affinity-map
   name red bit-position 2
   name blue bit-position 0
   name green bit-position 1
  !
```

```
 !
!
```

Verification

Note Use the auto-generated SR policy name assigned by the router. Auto-generated SR policy names use the following naming convention: **srte_c_***color-value***_ep_***endpoint-address*. For example, **srte_c_20_ep_10.1.1.4**.

```
RP/0/RP0/CPU0:ios# show segment-routing traffic-eng policy name srte_c_20_ep_10.1.1.4

SR-TE policy database
---------------------

Color: 20, End-point: 10.1.1.4
  Name: srte_c_20_ep_10.1.1.4
  Status:
    Admin: up  Operational: down for 00:07:22 (since Feb 19 17:14:55.564)
  Candidate-paths:
    Preference: 200 (configuration)
      Name: POLICY1
      Requested BSID: 1000
      Constraints:
        Protection Type: protected-preferred
        Affinity:
          exclude-any:
          red
        Maximum SID Depth: 8
      Explicit: segment-list SIDLIST1 (active)
        Weight: 1, Metric Type: TE
      Explicit: segment-list SIDLIST2 (active)
        Weight: 1, Metric Type: TE
    Preference: 100 (configuration)
      Name: POLICY1
      Requested BSID: 1000
      Explicit: segment-list SIDLIST3 (active)
        Weight: 1, Metric Type: TE
  Attributes:
    Forward Class: 0
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: no
    Invalidation drop enabled: no
```

Protocols

Path Computation Element Protocol

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

Configure the Head-End Router as PCEP PCC

Configure the head-end router as PCEP Path Computation Client (PCC) to establish a connection to the PCE. The PCC and PCE addresses must be routable so that TCP connection (to exchange PCEP messages) can be established between PCC and PCE.

Configure the PCC to Establish a Connection to the PCE

Use the **segment-routing traffic-eng pcc** command to configure the PCC source address, the SR-PCE address, and SR-PCE options.

A PCE can be given an optional precedence. If a PCC is connected to multiple PCEs, the PCC selects a PCE with the lowest precedence value. If there is a tie, a PCE with the highest IP address is chosen for computing path. The precedence *value* range is from 0 to 255.

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 local-source-address
Router(config-sr-te-pcc)# pce address ipv4 PCE-address[precedence value]
Router(config-sr-te-pcc)# pce address ipv4 PCE-address[password {clear | encrypted} LINE]
Router(config-sr-te-pcc)# pce address ipv4 PCE-address[keychain WORD]
```

Configure PCEP-Related Timers

Use the **timers keepalive** command to specify how often keepalive messages are sent from PCC to its peers. The range is from 0 to 255 seconds; the default value is 30.

```
Router(config-sr-te-pcc)# timers keepalive seconds
```

Use the **timers deadtimer** command to specify how long the remote peers wait before bringing down the PCEP session if no PCEP messages are received from this PCC. The range is from 1 to 255 seconds; the default value is 120.

```
Router(config-sr-te-pcc)# timers deadtimer seconds
```

Use the **timers delegation-timeout** command to specify how long a delegated SR policy can remain up without an active connection to a PCE. The range is from 0 to 3600 seconds; the default value is 60.

```
Router(config-sr-te-pcc)# timers delegation-timeout seconds
```

PCE-Initiated SR Policy Timers

Use the **timers initiated orphans** command to specify the amount of time that a PCE-initiated SR policy will remain delegated to a PCE peer that is no longer reachable by the PCC. The range is from 10 to 180 seconds; the default value is 180.

```
Router(config-sr-te-pcc)# timers initiated orphans seconds
```

Use the **timers initiated state** command to specify the amount of time that a PCE-initiated SR policy will remain programmed while not being delegated to any PCE. The range is from 15 to 14440 seconds (24 hours); the default value is 600.

```
Router(config-sr-te-pcc)# timers initiated state seconds
```

Note Multicast follows the same timer for Point-to-Multipoint (P2MP) policies:

- The default interval is 10 minutes.
- You can set the internal timer to '0' to ensure that forwarding states remain active indefinitely, even if the PCE is down for more than 5 minutes.
- If the PCE is down for more than 5 minutes, it is because the PCE is going down or the reachability to PCE going down.

To better understand how the PCE-initiated SR policy timers operate, consider the following example:

- PCE A instantiates SR policy P at head-end N.
- Head-end N delegates SR policy P to PCE A and programs it in forwarding.
- If head-end N detects that PCE A is no longer reachable, then head-end N starts the PCE-initiated **orphan** and **state** timers for SR policy P.
- If PCE A reconnects before the **orphan** timer expires, then SR policy P is automatically delegated back to its original PCE (PCE A).
- After the **orphan** timer expires, SR policy P will be eligible for delegation to any other surviving PCE(s).
- If SR policy P is not delegated to another PCE before the **state** timer expires, then head-end N will remove SR policy P from its forwarding.

Enable SR-TE SYSLOG Alarms

Use the **logging policy status** command to enable SR-TE related SYSLOG alarms.

```
Router(config-sr-te)# logging policy status
```

Enable PCEP Reports to SR-PCE

Use the **report-all** command to enable the PCC to report all SR policies in its database to the PCE.

```
Router(config-sr-te-pcc)# report-all
```

Customize MSD Value at PCC

Use the **maximum-sid-depth** *value* command to customize the Maximum SID Depth (MSD) signaled by PCC during PCEP session establishment.

The default MSD *value* is equal to the maximum MSD supported by the platform (5).

```
Router(config-sr-te)# maximum-sid-depth value
```

Note The platform's SR-TE label imposition capabilities are as follows:

- Up to 5 transport labels when no service labels are imposed
- Up to 3 transport labels when service labels are imposed

For cases with path computation at PCE, a PCC can signal its MSD to the PCE in the following ways:

- During PCEP session establishment – The signaled MSD is treated as a node-wide property.
  - MSD is configured under **segment-routing traffic-eng maximum-sid-depth** *value* command
- During PCEP LSP path request – The signaled MSD is treated as an LSP property.
  - On-demand (ODN) SR Policy: MSD is configured using the **segment-routing traffic-eng on-demand color** *color* **maximum-sid-depth** *value* command
  - Local SR Policy: MSD is configured using the **segment-routing traffic-eng policy** *WORD* **candidate-paths preference** *preference* **dynamic metric sid-limit** *value* command.

Note If the configured MSD values are different, the per-LSP MSD takes precedence over the per-node MSD.

After path computation, the resulting label stack size is verified against the MSD requirement.

- If the label stack size is larger than the MSD and path computation is performed by PCE, then the PCE returns a "no path" response to the PCC.
- If the label stack size is larger than the MSD and path computation is performed by PCC, then the PCC will not install the path.

Note A sub-optimal path (if one exists) that satisfies the MSD constraint could be computed in the following cases:

- For a dynamic path with TE metric, when the PCE is configured with the **pce segment-routing te-latency** command or the PCC is configured with the **segment-routing traffic-eng te-latency** command.
- For a dynamic path with LATENCY metric
- For a dynamic path with affinity constraints

For example, if the PCC MSD is 4 and the optimal path (with an accumulated metric of 100) requires 5 labels, but a sub-optimal path exists (with accumulated metric of 110) requiring 4 labels, then the sub-optimal path is installed.

Customize the SR-TE Path Calculation

Use the **te-latency** command to enable ECMP-aware path computation for TE metric.

```
Router(config-sr-te)# te-latency
```

Note ECMP-aware path computation is enabled by default for IGP and LATENCY metrics.

Configure PCEP Redundancy Type

Use the **redundancy pcc-centric** command to enable PCC-centric high-availability model, where the PCC allows only the PCE with the lowest precedence to initiate policies.

```
Router(config-sr-te-pcc)# redundancy pcc-centric
```

Configuring Head-End Router as PCEP PCC and Customizing SR-TE Related Options: Example

The following example shows how to configure an SR-TE head-end router with the following functionality:

- Enable the SR-TE head-end router as a PCEP client (PCC) with 3 PCEP servers (PCE) with different precedence values. The PCE with IP address 10.1.1.57 is selected as BEST.
- Enable SR-TE related syslogs.
- Set the Maximum SID Depth (MSD) signaled during PCEP session establishment to 5.
- Enable PCEP reporting for all policies in the node.

```
segment-routing
 traffic-eng
  pcc
   source-address ipv4 10.1.1.2
   pce address ipv4 10.1.1.57
    precedence 150
    password clear <password>
   !
   pce address ipv4 10.1.1.58
    precedence 200
    password clear <password>
   !
   pce address ipv4 10.1.1.59
    precedence 250
    password clear <password>
   !
  !
  logging
   policy status
  !
  maximum-sid-depth 5
  pcc
   report-all
  !
 !
!
end
```

Verification

```
RP/0/RSP0/CPU0:Router# show segment-routing traffic-eng pcc ipv4 peer

PCC's peer database:
--------------------

Peer address: 10.1.1.57, Precedence: 150, (best PCE)
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation

Peer address: 10.1.1.58, Precedence: 200
```

```
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation

Peer address: 10.1.1.59, Precedence: 250
  State up
  Capabilities: Stateful, Update, Segment-Routing, Instantiation
```

BGP SR-TE

BGP may be used to distribute SR Policy candidate paths to an SR-TE head-end. Dedicated BGP SAFI and NLRI have been defined to advertise a candidate path of an SR Policy. The advertisement of Segment Routing policies in BGP is documented in the IETF draft https://datatracker.ietf.org/doc/draft-ietf-idr-segment-routing-te-policy/

SR policies with IPv4 and IPv6 end-points can be advertised over BGPv4 or BGPv6 sessions between the SR-TE controller and the SR-TE headend.

The Cisco IOS-XR implementation supports the following combinations:

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

Configure BGP SR Policy Address Family at SR-TE Head-End

Perform this task to configure BGP SR policy address family at SR-TE head-end:

SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 65000`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp router-id** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.1.1.1`	Configures the local router with a specified router ID.
Step 4	**address-family** {**ipv4** | **ipv6**} **sr-policy** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 sr-policy`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
Step 5	**exit**	
Step 6	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.0.1`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 7	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 8	**address-family** {**ipv4** | **ipv6**} **sr-policy** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 sr-policy`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
Step 9	**route-policy** *route-policy-name* {**in** | **out**} **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass out`	Applies the specified policy to IPv4 or IPv6 unicast routes.

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

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

```
router bgp 65000
bgp router-id 10.1.1.1
 !
 address-family ipv4 sr-policy
 !
 address-family ipv6 sr-policy
 !
 neighbor 10.1.3.1
  remote-as 10
  description *** eBGP session to BGP SRTE controller ***
  address-family ipv4 sr-policy
   route-policy pass in
   route-policy pass out
  !
  address-family ipv6 sr-policy
   route-policy pass in
   route-policy pass out
  !
 !
!
```

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

The following configuration shows an SR-TE head-end with a BGPv6 session towards a BGP SR-TE controller. This BGP session is used to signal both IPv4 and IPv6 SR policies.

```
router bgp 65000
bgp router-id 10.1.1.1
 address-family ipv4 sr-policy
 !
 address-family ipv6 sr-policy
 !
 neighbor 3001::10:1:3:1
  remote-as 10
  description *** eBGP session to BGP SRTE controller ***
  address-family ipv4 sr-policy
   route-policy pass in
   route-policy pass out
  !
  address-family ipv6 sr-policy
   route-policy pass in
   route-policy pass out
  !
 !
!
```

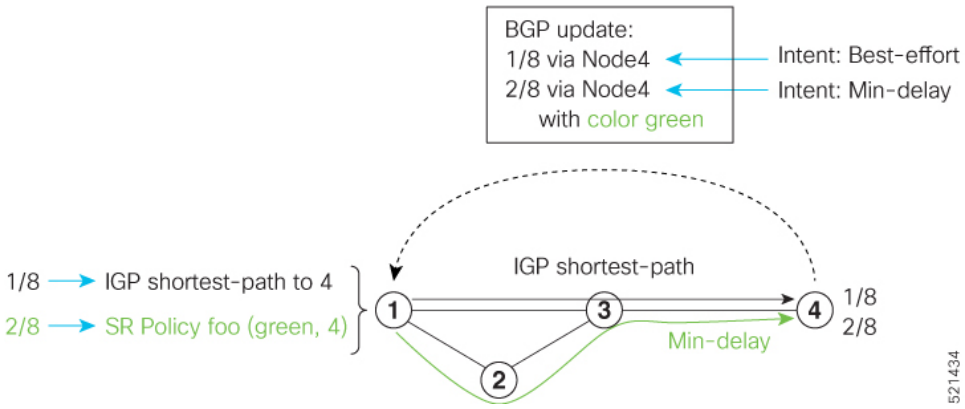
Traffic Steering

Automated Steering

Automated steering (AS) allows service traffic to be automatically steered onto the required transport SLA path programmed by an SR policy.

With AS, BGP automatically steers traffic onto an SR Policy based on the next-hop and color of a BGP service route. The color of a BGP service route is specified by a color extended community attribute. This color is used as a transport SLA indicator, such as min-delay or min-cost.

When the next-hop and color of a BGP service route matches the end-point and color of an SR Policy, BGP automatically installs the route resolving onto the BSID of the matching SR Policy. Recall that an SR Policy on a head-end is uniquely identified by an end-point and color.



When a BGP route has multiple extended-color communities, each with a valid SR Policy, the BGP process installs the route on the SR Policy giving preference to the color with the highest numerical value.

The granularity of AS behaviors can be applied at multiple levels, for example:

- At a service level—When traffic destined to all prefixes in a given service is associated to the same transport path type. All prefixes share the same color.
- At a destination/prefix level—When traffic destined to a prefix in a given service is associated to a specific transport path type. Each prefix could be assigned a different color.
- At a flow level—When flows destined to the same prefix are associated with different transport path types

AS behaviors apply regardless of the instantiation method of the SR policy, including:

- On-demand SR policy
- Manually provisioned SR policy
- PCE-initiated SR policy

Color-Only Automated Steering

Color-only steering is a traffic steering mechanism where a policy is created with given color, regardless of the endpoint.

You can create an SR-TE policy for a specific color that uses a NULL end-point (0.0.0.0 for IPv4 NULL, and ::0 for IPv6 NULL end-point). This means that you can have a single policy that can steer traffic that is based on that color and a NULL endpoint for routes with a particular color extended community, but different destinations (next-hop).

Note Every SR-TE policy with a NULL end-point must have an explicit path-option. The policy cannot have a dynamic path-option (where the path is computed by the head-end or PCE) since there is no destination for the policy.

You can also specify a color-only (CO) flag in the color extended community for overlay routes. The CO flag allows the selection of an SR-policy with a matching color, regardless of endpoint Sub-address Family Identifier (SAFI) (IPv4 or IPv6). See Setting the Color-Only Flag, on page 392.

Configure Color-Only Steering

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy P1
Router(config-sr-te-policy)# color 1 end-point ipv4 0.0.0.0

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy P2
Router(config-sr-te-policy)# color 2 end-point ipv6 ::0


Router# show running-configuration
segment-routing
 traffic-eng
  policy P1
   color 1 end-point ipv4 0.0.0.0
  !
  policy P2
   color 2 end-point ipv6 ::
  !
 !
!
end
```

Setting the Color-Only Flag

The BGP-based steering mechanism matches BGP color and next-hop with that of an SR-TE policy. If the policy does not exist, BGP requests SR-PCE to create an SR-TE policy with the associated color, end-point, and explicit paths. For color-only steering (NULL end-point), you can configure a color-only (CO) flag as part of the color extended community in BGP.

Note See Color-Only Automated Steering, on page 391 for information about color-only steering (NULL end-point).

The behavior of the steering mechanism is based on the following values of the CO flags:

co-flag 00	**1.** The BGP next-hop and color <N, C> is matched with an SR-TE policy of same <N, C>. **2.** If a policy does not exist, then IGP path for the next-hop N is chosen.

co-flag 01	**1.** The BGP next-hop and color <N, C> is matched with an SR-TE policy of same <N, C>. **2.** If a policy does not exist, then an SR-TE policy with NULL end-point with the same address-family as N and color C is chosen. **3.** If a policy with NULL end-point with same address-family as N does not exist, then an SR-TE policy with any NULL end-point and color C is chosen. **4.** If no match is found, then IGP path for the next-hop N is chosen.

Configuration Example

```
Router(config)# extcommunity-set opaque overlay-color
Router(config-ext)# 1 co-flag 01
Router(config-ext)# end-set
Router(config)#
Router(config)# route-policy color
Router(config-rpl)# if destination in (10.5.5.1/32) then
Router(config-rpl-if)# set extcommunity color overlay-color
Router(config-rpl-if)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
Router(config)#
```

Address-Family Agnostic Automated Steering

Address-family agnostic steering uses an SR-TE policy to steer both labeled and unlabeled IPv4 and IPv6 traffic. This feature requires support of IPv6 encapsulation (IPv6 caps) over IPV4 endpoint policy.

IPv6 caps for IPv4 NULL end-point is enabled automatically when the policy is created in Segment Routing Path Computation Element (SR-PCE). The binding SID (BSID) state notification for each policy contains an "ipv6_caps" flag that notifies SR-PCE clients (PCC) of the status of IPv6 caps (enabled or disabled).

An SR-TE policy with a given color and IPv4 NULL end-point could have more than one candidate path. If any of the candidate paths has IPv6 caps enabled, then all of the remaining candidate paths need IPv6 caps enabled. If IPv6 caps is not enabled on all candidate paths of same color and end-point, traffic drops can occur.

You can disable IPv6 caps for a particular color and IPv4 NULL end-point using the **ipv6 disable** command on the local policy. This command disables IPv6 caps on all candidate paths that share the same color and IPv4 NULL end-point.

Disable IPv6 Encapsulation

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy P1
Router(config-sr-te-policy)# color 1 end-point ipv4 0.0.0.0
Router(config-sr-te-policy)# ipv6 disable
```

Per-Flow Automated Steering

Table 44: Feature History Table

Feature Name	Release Information	Feature Description
SR-TE: Per Flow Automated Steering	Release 7.11.1	This feature extends the SR-TE automated steering capabilities with per-flow granularity. Before this feature, per-destination automated steering (AS) dynamically steered all traffic flows destined to a BGP service prefix over a single path of an SR policy. With this feature, Per-Flow AS dynamically steers traffic flows destined to a BGP service prefix over different paths across the network. Traffic flows are determined based on the attributes of incoming packets (for example, source/destination IP address or QoS markings). Per-flow AS is realized using a Per-Flow SR policy (PFP). PFPs offer the user the flexibility to deliver the desired transport paths for different traffic flows.

Per-destination automated steering (AS) dynamically steered all traffic flows destined to a BGP service prefix over a single path of an SR policy. This type of policy is called a Per-Destination Policy (PDP).

The steering of traffic through a Segment Routing (SR) policy is based on the candidate paths of that policy. For a given policy, a candidate path specifies the path to be used to steer traffic to the policy's destination. The policy determines which candidate path to use based on the candidate path's preference and state. The candidate path that is valid and has the highest preference is used to steer all traffic using the given policy. This type of policy is called a Per-Destination Policy (PDP).

Per-Flow Automated Traffic Steering using SR-TE Policies introduces a way to steer traffic on an SR policy based on the attributes of the incoming packets, called a Per-Flow Policy (PFP).

These policies can be used to control and manage the behavior of traffic flows within a network. Forward Class (FC) are groups of flows that share similar characteristics or requirements. These classes are defined based on criteria such as QoS requirements, application types, or other factors. Each flow is assigned to a specific FC based on its attributes, as determined by the Per-Flow Policy. Multiple flows can be part of the same FC. The FC is represented as a numeric value of 0 to 7, providing up to 8 options to the endpoint.

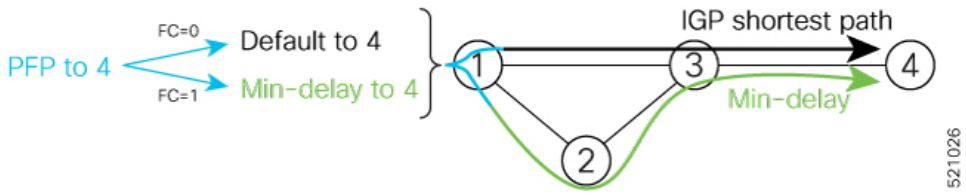
A PFP defines an array of FC-to-PDP mappings. A PFP can then be used to steer traffic into a given PDP based on the FC assigned to a packet.

As with PDPs, PFPs are identified by a {headend, color, endpoint} tuple. The color associated with a given FC corresponds to a valid PDP policy of that color and same endpoint as the parent PFP. So PFP policies contain mappings of different FCs to valid PDP policies of different colors.

Every PFP has an FC designated as its default FC. The default FC is associated to packets with an undefined FC under the PFP, or for packets with an FC that has no valid PDP policy.

The following example shows a per-flow policy from Node1 to Node4:

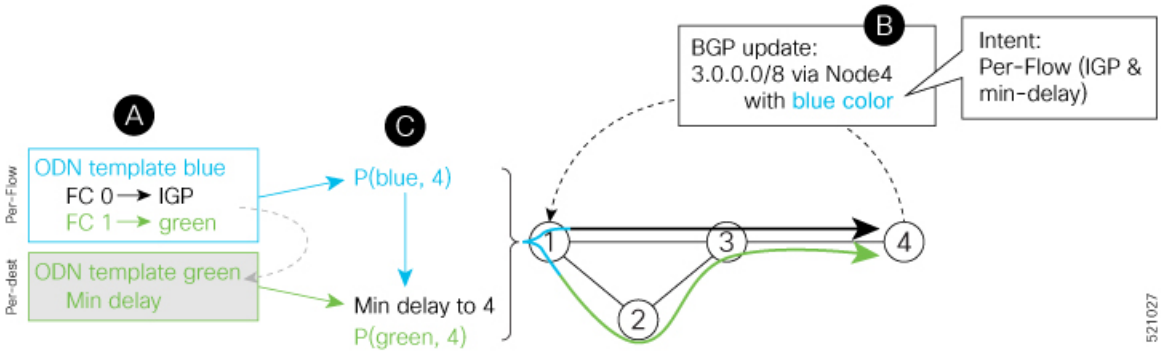
Figure 28: PFP Example



- FC=0 (default) -> shortest path to Node4
  - IGP shortest path = 16004
- FC=1 -> Min-delay path to Node4
  - SID list = {16002,16004}

The same on-demand instantiation behaviors of PDPs apply to PFPs. For example, an edge node automatically (on demand) instantiates Per-Flow SR Policy paths to an endpoint by service route signaling. Automated Steering steers the service route in the matching SR Policy.

Figure 29: PFP with ODN Example



Like PDPs, PFPs have a binding SID (BSID). Existing SR-TE automated steering (AS) mechanisms for labeled traffic (via BSID) and unlabeled traffic (via BGP) onto a PFP is similar to that of a PDP. The classification policy on the ingress interface marks the packet with an FC based on the configured class-map. The packet is then steered to the PDP that corresponds to that FC.

Usage Guidelines and Limitations

The following guidelines and limitations apply to the platform when acting as a head-end of a PFP policy:

- BGP IPv4 unicast over PFP (steered via ODN/AS) is supported
- BGP IPv6 unicast (with IPv4 next-hop [6PE]) over PFP (steered via ODN/AS) is supported

- BGP IPv6 unicast (with IPv6 next-hop) over PFP (steered via ODN/AS) is supported
- BGP VPNv4 over PFP (steered via ODN/AS) is supported
- BGP VPNv6 (6VPE) over PFP (steered via ODN/AS) is supported
- BGP EVPN over PFP is not supported
- Pseudowire and VPLS over PFP are not supported
- BGP multipath is supported
- BGP PIC is not supported
- Labeled traffic (Binding SID as top-most label in the stack) steered over PFP is supported
- When not explicitly configured, FC 0 is the default FC.
- A PFP is considered valid as long as its default FC has a valid PDP.
- An ingress QoS policy applied to an input interface is used to classify flows and set corresponding MPLS experimental values.
- The following counters are supported:
  - PFP's BSID counter (packet, bytes)
  - Per-FC counters (packet, byte)
    - Collected from the PDP's segment-list-per-path egress counters
    - If an SR policy is used for more than one purpose (as a regular policy as well as a PDP under one or more PFPs), then the collected counters will represent the aggregate of all contributions. To preserve independent counters, it is recommended that an SR policy be used only for one purpose.
- Based on the following fields, the inbound packet classification is supported:
  - IP precedence
  - IP DSCP
  - L3 ACL-based (L3 source/destination IP; L4 source/destination port)
  - MPLS EXP
- A color associated with a PFP SR policy cannot be used by a non-PFP SR policy. For example, if a per-flow ODN template for color 100 is configured, then the system will reject the configuration of any non-PFP SR policy using the same color. You must assign different color value ranges for PFP and non-PFP SR policies.

Configuring ODN Template for PFP Policies: Example

The following example depicts an ODN template for PFP policies that includes three FCs.

The example also includes the corresponding ODN templates for PDPs as follows:

- FC0 (default FC) mapped to color 10 = Min IGP path

- FC1 mapped to color 20 = Flex Algo 128 path
- FC2 mapped to color 30 = Flex Algo 129 path

```
segment-routing
 traffic-eng
  on-demand color 10
   dynamic
    metric
     type igp
    !
   !
  !
  on-demand color 20
   constraints
    segments
     sid-algorithm 128
    !
   !
  !
  on-demand color 30
   constraints
    segments
     sid-algorithm 129
    !
   !
  !
  on-demand color 1000
   per-flow
    forward-class 0 color 10
    forward-class 1 color 20
    forward-class 2 color 30
```

Manually Configuring a PFP and PDPs: Example

The following example depicts a manually defined PFP that includes three FCs and corresponding manually defined PDPs.

The example also includes the corresponding PDPs as follows:

- FC0 (default FC) mapped to color 10 = Min IGP path
- FC1 mapped to color 20 = Min TE path
- FC2 mapped to color 30 = Min delay path

```
segment-routing
 traffic-eng
  policy MyPerFlow
   color 1000 end-point ipv4 10.1.1.4
   candidate-paths
    preference 100
     per-flow
      forward-class 0 color 10
      forward-class 1 color 20
      forward-class 2 color 30
  !
  policy MyLowIGP
   color 10 end-point ipv4 10.1.1.4
   candidate-paths
    preference 100
```

```
    dynamic
     metric type igp
  !
  policy MyLowTE
   color 20 end-point ipv4 10.1.1.4
   candidate-paths
    preference 100
     dynamic
      metric type te
  !
  policy MyLowDelay
   color 30 end-point ipv4 10.1.1.4
   candidate-paths
    preference 100
     dynamic
      metric type delay
```

Configuring Ingress Classification: Example

An ingress classification policy is used to classify and mark traffic to a corresponding forwarding class.

The following shows an example of such ingress classification policy:

```
class-map match-any MinDelay
 match dscp 46
 end-class-map
!
class-map match-any PremiumHosts
 match access-group ipv4 PrioHosts
 end-class-map
 !
!
policy-map MyPerFlowClassificationPolicy
 class MinDelay
  set forward-class 2
 !
 class PremiumHosts
  set forward-class 1
 !
 class class-default
 !
 end-policy-map
!
interface GigabitEthernet0/0/0/0
 description PE_Ingress_Interface
 service-policy input MyPerFlowClassificationPolicy
!
```

Determining Per-Flow Policy State

A PFP is brought down for the following reasons:

- The PDP associated with the default FC is in a down state.
- All FCs are associated with PDPs that are in a down state.
- The FC assigned as the default FC is missing in the forward class mapping.

  Scenario 1—FC 0 (default FC) is not configured in the FC mappings below:

  ```
  policy foo
    color 1 end-point ipv4 10.1.1.1
    per-flow
  ```

```
   forward-class 1 color 10
   forward-class 2 color 20
```

Scenario 2—FC 1 is configured as the default FC, however it is not present in the FC mappings:

```
policy foo
  color 1 end-point ipv4 10.1.1.1
  per-flow
   forward-class 0 color 10
   forward-class 2 color 20
   forward-class default 1
```

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.

Explicit Binding SID

Use the **binding-sid mpls** *label* command in SR-TE policy configuration mode to specify the explicit BSID. Explicit BSIDs are allocated from the segment routing local block (SRLB) or the dynamic range of labels. A best-effort is made to request and obtain the BSID for the SR-TE policy. If requested BSID is not available (if it does not fall within the available SRLB or is already used by another application or SR-TE policy), the policy stays down.

Use the **binding-sid explicit** {**fallback-dynamic** | **enforce-srlb**} command to specify how the BSID allocation behaves if the BSID value is not available.

- Fallback to dynamic allocation – If the BSID is not available, the BSID is allocated dynamically and the policy comes up:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# binding-sid explicit fallback-dynamic
```

- Strict SRLB enforcement – If the BSID is not within the SRLB, the policy stays down:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# binding-sid explicit enforce-srlb
```

This example shows how to configure an SR policy to use an explicit BSID of 1000. If the BSID is not available, the BSID is allocated dynamically and the policy comes up.

```
segment-routing
 traffic-eng
  binding-sid explicit fallback-dynamic
  policy goo
   binding-sid mpls 1000
  !
 !
!
```

Stitching SR-TE Polices Using Binding SID: Example

In this example, three SR-TE policies are stitched together to form a seamless end-to-end path from node 1 to node 10. The path is a chain of SR-TE policies stitched together using the binding-SIDs of intermediate policies, providing a seamless end-to-end path.

Figure 30: Stitching SR-TE Polices Using Binding SID

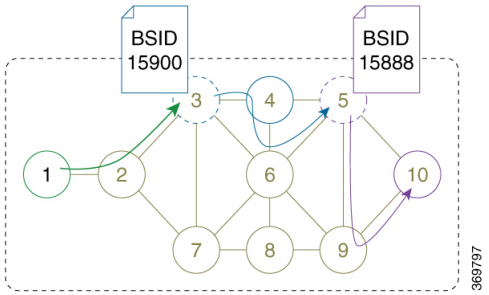


Table 45: Router IP Address

Router	Prefix Address	Prefix SID/Adj-SID
3	Loopback0 - 10.1.1.3	Prefix SID - 16003
4	Loopback0 - 10.1.1.4 Link node 4 to node 6 - 10.4.6.4	Prefix SID - 16004 Adjacency SID - dynamic
5	Loopback0 - 10.1.1.5	Prefix SID - 16005

Router	Prefix Address	Prefix SID/Adj-SID
6	Loopback0 - 10.1.1.6 Link node 4 to node 6 - 10.4.6.6	Prefix SID - 16006 Adjacency SID - dynamic
9	Loopback0 - 10.1.1.9	Prefix SID - 16009
10	Loopback0 - 10.1.1.10	Prefix SID - 16010

Procedure

Step 1 On node 5, do the following:

a) Define an SR-TE policy with an explicit path configured using the loopback interface IP addresses of node 9 and node 10.

b) Define an explicit binding-SID (**mpls label 15888**) allocated from SRLB for the SR-TE policy.

Example:

Node 5

```
segment-routing
 traffic-eng
  segment-list PATH-9_10
   index 10 address ipv4 10.1.1.9
   index 20 address ipv4 10.1.1.10
  !
  policy foo
   binding-sid mpls 15888
   color 777 end-point ipv4 10.1.1.10
   candidate-paths
    preference 100
     explicit segment-list PATH5-9_10
     !
    !
   !
  !
 !
!

RP/0/RSP0/CPU0:Node-5# show segment-routing traffic-eng policy color 777

SR-TE policy database
---------------------

Color: 777, End-point: 10.1.1.10
  Name: srte_c_777_ep_10.1.1.10
  Status:
    Admin: up  Operational: up for 00:00:52 (since Aug 19 07:40:12.662)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Name: foo
      Requested BSID: 15888
      PCC info:
        Symbolic name: cfg_foo_discr_100
        PLSP-ID: 70
      Explicit: segment-list PATH-9_10 (valid)
        Weight: 1, Metric Type: TE
          16009 [Prefix-SID, 10.1.1.9]
```

```
        16010 [Prefix-SID, 10.1.1.10]
  Attributes:
    Binding SID: 15888 (SRLB)
    Forward Class: 0
    Steering BGP disabled: no
    IPv6 caps enable: yes
```

Step 2 On node 3, do the following:

a) Define an SR-TE policy with an explicit path configured using the following:

- Loopback interface IP address of node 4
- Interface IP address of link between node 4 and node 6
- Loopback interface IP address of node 5
- Binding-SID of the SR-TE policy defined in Step 1 (**mpls label 15888**)

Note This last segment allows the stitching of these policies.

b) Define an explicit binding-SID (**mpls label 15900**) allocated from SRLB for the SR-TE policy.

Example:

Node 3

```
segment-routing
 traffic-eng
  segment-list PATH-4_4-6_5_BSID
   index 10 address ipv4 10.1.1.4
   index 20 address ipv4 10.4.6.6
   index 30 address ipv4 10.1.1.5
   index 40 mpls label 15888
  !
  policy baa
   binding-sid mpls 15900
   color 777 end-point ipv4 10.1.1.5
   candidate-paths
    preference 100
     explicit segment-list PATH-4_4-6_5_BSID
     !
    !
   !
  !
 !
!

RP/0/RSP0/CPU0:Node-3# show segment-routing traffic-eng policy color 777

SR-TE policy database
---------------------

Color: 777, End-point: 10.1.1.5
  Name: srte_c_777_ep_10.1.1.5
  Status:
    Admin: up  Operational: up for 00:00:32 (since Aug 19 07:40:32.662)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Name: baa
      Requested BSID: 15900
      PCC info:
        Symbolic name: cfg_baa_discr_100
        PLSP-ID: 70
```

```
        Explicit: segment-list PATH-4_4-6_5_BSID (valid)
          Weight: 1, Metric Type: TE
            16004 [Prefix-SID, 10.1.1.4]
            80005 [Adjacency-SID, 10.4.6.4 - 10.4.6.6]
            16005 [Prefix-SID, 10.1.1.5]
            15888
    Attributes:
      Binding SID: 15900 (SRLB)
      Forward Class: 0
      Steering BGP disabled: no
      IPv6 caps enable: yes
```

Step 3 On node 1, define an SR-TE policy with an explicit path configured using the loopback interface IP address of node 3 and the binding-SID of the SR-TE policy defined in step 2 (**mpls label 15900**). This last segment allows the stitching of these policies.

Example:

Node 1

```
segment-routing
 traffic-eng
  segment-list PATH-3_BSID
   index 10 address ipv4 10.1.1.3
   index 20 mpls label 15900
  !
  policy bar
   color 777 end-point ipv4 10.1.1.3
   candidate-paths
    preference 100
     explicit segment-list PATH-3_BSID
     !
    !
   !
  !
 !
!

RP/0/RSP0/CPU0:Node-1# show segment-routing traffic-eng policy color 777

SR-TE policy database
---------------------

Color: 777, End-point: 10.1.1.3
  Name: srte_c_777_ep_10.1.1.3
  Status:
    Admin: up  Operational: up for 00:00:12 (since Aug 19 07:40:52.662)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Name: bar
      Requested BSID: dynamic
      PCC info:
        Symbolic name: cfg_bar_discr_100
        PLSP-ID: 70
      Explicit: segment-list PATH-3_BSID (valid)
        Weight: 1, Metric Type: TE
          16003 [Prefix-SID, 10.1.1.3]
          15900
  Attributes:
    Binding SID: 80021
    Forward Class: 0
```

```
Steering BGP disabled: no
IPv6 caps enable: yes
```

Static Route Traffic-Steering using SR-TE Policy

Table 46: Feature History Table

Feature Name	Release Information	Feature Description
Static Route Traffic-Steering using SR-TE Policy	Release 7.10.1	IPv4 and IPv6 static routes now leverage the SR policies to aid Segment Routing Traffic Engineering (SR-TE). This facilitates traffic steering because you can now configure IP Static Route with SR static policy.

The Static Route Traffic-Steering using SR-TE Policy feature allows you to specify a Segment Routing (SR) policy when configuring static routes.

For information on configuring static routes, see the "Implementing Static Routes" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Configuration Example

```
Router(config)# router static
Router (config-static)# address-family ipv4 unicast

//configure administrative distance
Router (config-static-afi)# 192.2.0.0/24 sr-policy sample-policy 110

//Configure load metric
Router (config-static-afi)# 192.2.0.0/24 sr-policy sample-policy metric 50

//Install the route in RIB regardless of reachability
Router (config-static-afi)# 192.2.0.0/24 sr-policy sample-policy permanent
```

Running Configuration

```
Router# show running-configuration
 router static
  address-family ipv4 unicast
  192.2.0.1/24 sr-policy sample-policy 110
  192.2.0.1/24 sr-policy sample-policy metric 50
  192.2.0.1/24 sr-policy sample-policy permanent
  !
 !
!
```

Verification Steps

```
Router# show route 192.2.0.0/24

Routing entry for 192.2.0.0/24
```

```
Known via "static", distance 1, metric 0
Installed Jul 28 09:18:25.639 for 00:00:09
Routing Descriptor Blocks
directly connected, via srte_c_2_ep_10.3.3.10, permanent
Route distance is 110, metric is 0, Wt is 50
```

Label Distribution Protocol (LDP) over Segment Routing Traffic Engineering (SR-TE) Policy

Table 47: Feature History Table

Feature Name	Release Information	Feature Description
Label Distribution Protocol over Segment Routing Policy	Release 7.10.1	This feature extends the existing MPLS Label Distribution Protocol (LDP) address family neighbor configuration to specify an SR policy as the targeted end-point, thus integrating LDP over SR-TE in core networks and enabling interoperability between LDP and SR-capable devices. The feature introduces the **neighbor sr-policy** command.

LDP over SR policy is supported for locally configured SR policies with IPv4 end-points.

For more information about MPLS LDP, see the "Implementing MPLS Label Distribution Protocol" chapter in the *MPLS Configuration Guide*.

Note Before you configure an LDP targeted adjacency over SR policy name, you need to create the SR policy under Segment Routing configuration. The SR policy interface names are created internally based on the color and endpoint of the policy. LDP is non-operational if SR policy name is unknown.

The following functionality applies:

1. Configure the SR policy – LDP receives the associated end-point address from the interface manager (IM) and stores it in the LDP interface database (IDB) for the configured SR policy.
2. Configure the SR policy name under LDP – LDP retrieves the stored end-point address from the IDB and uses it. Use the auto-generated SR policy name assigned by the router when creating an LDP targeted adjacency over an SR policy.

Note You can use the **show segment-routing traffic-eng policy** command to display the auto generated SR policy name. Auto-generated SR policy name uses the following naming convention: **srte_c_***color_val***_ep_***endpoint-address*. For example, **srte_c_1000_ep_10.1.1.2**

Configuration Example

```
/* Enter the SR-TE configuration mode and create the SR policy. This example corresponds
to a local SR policy with an explicit path. */
```

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list sample-sid-list
Router(config-sr-te-sl)# index 10 address ipv4 10.1.1.7
Router(config-sr-te-sl)# index 20 address ipv4 10.1.1.2
Router(config-sr-te-sl)# exit
Router(config-sr-te)# policy sample_policy
Router(config-sr-te-policy)# color 1000 end-point ipv4 10.1.1.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list sample-sid-list
Router(config-sr-te-pp-info)# end

/* Configure LDP over an SR policy */
Router(config)# mpls ldp
Router(config-ldp)# address-family ipv4
Router(config-ldp-af)# neighbor sr-policy srte_c_1000_ep_10.1.1.2 targeted
Router(config-ldp-af)#
```

Note Do one of the following to configure LDP discovery for targeted hellos. Ensure you use the same method for the peer router, it must have similar configurations:

- Active targeted hellos (SR policy head end):

```
mpls ldp
 interface GigabitEthernet0/0/0/0
 !
!
```

- Passive targeted hellos (SR policy end-point):

```
mpls ldp
 address-family ipv4
  discovery targeted-hello accept
 !
!
```

Running Configuration

```
segment-routing
 traffic-eng
  segment-list sample-sid-list
   index 10 address ipv4 10.1.1.7
   index 20 address ipv4 10.1.1.2
  !
  policy sample_policy
   color 1000 end-point ipv4 10.1.1.2
   candidate-paths
    preference 100
     explicit segment-list sample-sid-list
     !
    !
   !
  !
 !
!

mpls ldp
 address-family ipv4
```

```
  neighbor sr-policy srte_c_1000_ep_10.1.1.2 targeted
  discovery targeted-hello accept
 !
!
```

Verification

```
Router# show mpls ldp interface brief
Interface       VRF Name            Config Enabled IGP-Auto-Cfg TE-Mesh-Grp cfg
--------------- ------------------- ------ ------- ------------ ---------------
Te0/3/0/0/3     default             Y      Y       0            N/A
Te0/3/0/0/6     default             Y      Y       0            N/A
Te0/3/0/0/7     default             Y      Y       0            N/A
Te0/3/0/0/8     default             N      N       0            N/A
Te0/3/0/0/9     default             N      N       0            N/A
srte_c_1000_    default             Y      Y       0            N/A


Router# show mpls ldp interface
Interface TenGigE0/3/0/0/3 (0xa000340)
   VRF: 'default' (0x60000000)
   Enabled via config: LDP interface
Interface TenGigE0/3/0/0/6 (0xa000400)
   VRF: 'default' (0x60000000)
   Enabled via config: LDP interface
Interface TenGigE0/3/0/0/7 (0xa000440)
   VRF: 'default' (0x60000000)
   Enabled via config: LDP interface
Interface TenGigE0/3/0/0/8 (0xa000480)
   VRF: 'default' (0x60000000)
   Disabled:
Interface TenGigE0/3/0/0/9 (0xa0004c0)
   VRF: 'default' (0x60000000)
   Disabled:
Interface srte_c_1000_ep_10.1.1.2 (0x520)
   VRF: 'default' (0x60000000)
   Enabled via config: LDP interface


Router# show segment-routing traffic-eng policy color 1000

SR-TE policy database
---------------------

Color: 1000, End-point: 1.1.1.2
  Name: srte_c_1000_ep_10.1.1.2
  Status:
    Admin: up  Operational: up for 00:02:00 (since Jul  2 22:39:06.663)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Name: sample_policy
      Requested BSID: dynamic
      PCC info:
        Symbolic name: cfg_sample_policy_discr_100
        PLSP-ID: 17
      Explicit: segment-list sample-sid-list (valid)
        Weight: 1, Metric Type: TE
          16007 [Prefix-SID, 10.1.1.7]
          16002 [Prefix-SID, 10.1.1.2]
  Attributes:
    Binding SID: 80011
    Forward Class: 0
    Steering BGP disabled: no
    IPv6 caps enable: yes
```

```
Router# show mpls ldp neighbor 10.1.1.2 detail

Peer LDP Identifier: 10.1.1.2:0
  TCP connection: 10.1.1.2:646 - 10.1.1.6:57473
  Graceful Restart: No
  Session Holdtime: 180 sec
  State: Oper; Msgs sent/rcvd: 421/423; Downstream-Unsolicited
  Up time: 05:22:02
  LDP Discovery Sources:
    IPv4: (1)
      Targeted Hello (10.1.1.6 -> 10.1.1.2, active/passive)
    IPv6: (0)
  Addresses bound to this peer:
    IPv4: (9)
      10.1.1.2       10.2.2.99       10.1.2.2       10.2.3.2
      10.2.4.2       10.2.22.2      10.2.222.2     10.30.110.132
      10.2.9.2
    IPv6: (0)
  Peer holdtime: 180 sec; KA interval: 60 sec; Peer state: Estab
  NSR: Disabled
  Clients: LDP over SR Policy
  Capabilities:
    Sent:
      0x508  (MP: Point-to-Multipoint (P2MP))
      0x509  (MP: Multipoint-to-Multipoint (MP2MP))
      0x50a  (MP: Make-Before-Break (MBB))
      0x50b  (Typed Wildcard FEC)
    Received:
      0x508  (MP: Point-to-Multipoint (P2MP))
      0x509  (MP: Multipoint-to-Multipoint (MP2MP))
      0x50a  (MP: Make-Before-Break (MBB))
      0x50b  (Typed Wildcard FEC)
```

Autoroute Include

Table 48: Feature History Table

Feature Name	Release Information	Feature Description
Support for mixed SR-TE Policy Autoroute paths, unprotected native paths, and protected (LFA/TI-LFA) native paths	Release 7.11.1	When an SR-TE policy is autoroute announced, an IGP route can use it as one of its nexthops, as well as other native paths for load balancing. If protection is configured under IGP, some of these native paths may have LFA or TI-LFA backup paths. In earlier releases, this mix of SR-TE policy paths and protected native paths wasn't supported. This feature adds support for protected (LFA/TI-LFA) native paths. This functionality is enabled by default.

Feature Name	Release Information	Feature Description
IPv6 'Autoroute Include' Support for an SR-TE Policy with an IPv4 Endpoint	Release 7.5.4	You can configure an SR-TE policy to automatically steer incoming unlabeled IPv6 traffic at the headend router into that SR-TE policy with an IPv4 endpoint. Compared to MPLS/RSVP-TE, SR-TE provides more granular and automated steering techniques. In earlier releases, you could automatically steer only incoming IPv4 traffic for specific or all prefixes. New command: **autoroute include ipv6 all**

You can configure SR-TE policies with Autoroute Include to steer all prefixes and specifically IGP (IS-IS, OSPF) prefixes over non-shortest paths and divert traffic for those prefixes onto the SR-TE policy.

The Autoroute SR-TE policy adds the prefixes into the IGP, which determines if the prefixes on the endpoint or downstream of the endpoint are eligible to use the SR-TE policy. If a prefix is eligible, then the IGP checks if the prefix is listed in the Autoroute Include configuration. If the prefix is included, then the IGP downloads the prefix route with the SR-TE policy as the outgoing path.

The **autoroute include ipv4** {**all** | *address*} option applies Autoroute Destination functionality for all eligible or specified IPv4 prefixes. The *address* option is supported for IS-IS only; it is not supported for OSPF.

The **autoroute include ipv6 all** option applies Autoroute Destination functionality for all eligible IPv6 prefixes.

Usage Guidelines and Limitations

- Autoroute Include for IPv6 is supported for unlabeled IGP prefixes and BGP penultimate next-hop (PNH).
- Autoroute Include supports three metric types:
  - Default (no metric): The path over the SR-TE policy inherits the shortest path metric.
  - Absolute (constant) metric: The shortest path metric to the policy endpoint is replaced with the configured absolute metric. The metric to any prefix that is Autoroute Included is modified to the absolute metric. Use the **autoroute metric constant** *constant-metric* command, where *constant-metric* is from 1 to 2147483647.
  - Relative metric: The shortest path metric to the policy endpoint is modified with the relative value configured (plus or minus). Use the **autoroute metric relative** *relative-metric* command, where *relative-metric* is from -10 to +10.

Note To prevent load-balancing over IGP paths, you can set a metric that is lower than the value considered by the IGP for autorouted destinations. For example, you can use a relative metric of **-1**. By doing this, you can influence the IGP to prefer other paths over the autorouted destinations, effectively preventing load-balancing over those paths.

- The ECMP path-set of an IGP route with a mix of SR-TE Policy paths (Autoroute) and unprotected native paths is supported.
- The ECMP path-set of an IGP route with a mix of SR-TE Policy paths (Autoroute) and protected (LFA/TI-LFA) native paths is supported.

Note Mixed paths are not supported when the following features are enabled:

- Increase in RSVP-TE Tunnel Scale for LDP over TE (using the **hw-module profile cef te-tunnel highscale-ldp-over-te-no-sr-over-srte -** command)
- LDP Over RSVP LSR (using the **hw-module profile cef te-tunnel highscale-no-ldp-over-te** command)
- SR-TE with Next-Hop Independent Scaling Optimization (using the **segment-routing traffic-eng separate-next-hop** command)

- LDP to SR-TE interworking is not supported.

Configuration Examples

The following example shows how to configure autoroute include for all IPv4 prefixes:

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)#policy P1
Router(config-sr-te-policy)# color 20 end-point ipv4 10.1.1.2
Router(config-sr-te-policy)# autoroute include ipv4 all
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-pp-index)# explicit segment-list Plist-1
```

The following example shows how to configure autoroute include for the specified IPv4 prefixes:

Note This option is supported for IS-IS only; it is not supported for OSPF.

```
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
```

```
Router(config-sr-te)#policy P1
Router(config-sr-te-policy)# color 20 end-point ipv4 10.1.1.2
Router(config-sr-te-policy)# autoroute include ipv4 10.1.1.21/32
Router(config-sr-te-policy)# autoroute include ipv4 10.1.1.23/32
Router(config-sr-te-policy)# autoroute metric constant 1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-pp-index)# explicit segment-list slist-1
```

The following example shows how to configure the IPv6 autoroute function for an SR-TE policy with an IPv4 endpoint:

```
Router# configure
Router(config)# segment-routing traffic-eng policy pol12
Router(config-sr-te-policy)# autoroute include ipv6 all
Router(config-sr-te-policy)# commit
```

The following example shows how to configure the IPv6 autoroute function for a PCE-instantiated SR-TE policy with an IPv4 endpoint:

```
Router# configure
Router(config)# segment-routing traffic-eng pcc profile 10
Router(config-pcc-prof)# autoroute include ipv6 all
Router(config-pcc-prof)# commit
```

Verification

```
Router# show segment-routing traffic-eng policy name srte_c_20_ep_10.1.1.2 private

SR-TE policy database
---------------------

Color: 20, End-point: 10.1.1.2 ID: 1
  Name: srte_c_20_ep_10.1.1.2
  Status:
    Admin: up  Operational: down for 00:05:57 (since Mar 13 18:08:26.690)
  Candidate-paths:
    Preference: 100 (configuration) (inactive)
      Originator: ASN 0 node-address <None> discriminator: 100
      Name: policy1
      Requested BSID: 15001
        Protection Type: protected-preferred
        Maximum SID Depth: 8
      ID: 1
      Source: <None>
      Stale: no
      Checkpoint flags: 0x00000000
      Autoroute:
        Force SR include: no
        Include IPv6 all: yes
        Prefix: 0.0.0.0/0
      Explicit: segment-list slist1 (inactive)
        Weight: 2, Metric Type: TE
        IGP area: 0

. . .

Router# show isis ipv6 route 2001:131:0:63::1/64 detail

L2 2001:131:0:63::1/64 (41/115) Label: None, low priority
   Installed Feb 22 23:03:14.620 for 00:00:02
     via ::, srte/c/1/ep/10.1.1.2, Label: Exp-Null-v6, SR-TB5-R2, SRGB Base: 21000, Weight:
 0
```

```
     src 0010.9400.0006.00-02, 2002::6702:102


Router# show route ipv6 2001:131:0:63::1/64 detail

Routing entry for 2001:131:0:63::/64
  Known via "isis core-sr", distance 115, metric 41, type level-2
  Installed Feb 22 23:03:14.624 for 00:04:20
  Routing Descriptor Blocks
    directly connected, via srte c_1_ep_10.1.1.2
      Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id:Oxe0000000
      Route metric 18 41
      Label: OX2 (2)
      Tunnel ID: None
      Binding Label: 0x272e (15001)
      Extended communities count: 0
      Path id:1
      Path ref count: 0
      NHID: 0x0 (Ref: 0)
      MPLS eid:Ox118aa00000002

. . .
```

The following output shows an ECMP path-set of an IGP route with a mix of SR-TE Policy paths (Autoroute Include) and protected (LFA/TI-LFA) native paths:

```
RP/0/RP0/CPU0:R1# show route ipv4 131.0.2.1

Routing entry for 131.0.2.1/32
  Known via "ospf core-sr", distance 110, metric 14, labeled SR(SRMS), type intra area
  Installed Jun  1 21:01:02.789 for 2d00h
  Routing Descriptor Blocks
    101.1.3.2, from 103.2.1.2, via Bundle-Ether1301, Protected
      Route metric is 14
    101.1.5.2, from 103.2.1.2, via Bundle-Ether1501, Backup (Local-LFA)
      Route metric is 15
    100.2.1.1, from 103.2.1.2, via srte_c_700_ep_100.2.1.1
      Route metric is 14
  No advertising protos.
```

SR-TE Automated Steering Without BGP Prefix Path Label

Table 49: Feature History Table

Feature Name	Release Information	Feature Description
SR-TE Automated Steering Without BGP Prefix Path Label	Release 7.9.1	This feature allows traffic to a BGP service route to be steered over an SR-TE policy using automated-steering principles without imposing the service route's prefix label. This feature allows you to deploy a centralized BGP EPE solution for 6PE in an SR-MPLS network. This feature introduces the **bgp prefix-path-label ignore** command.

This feature allows traffic to a BGP service route to be steered over an SR-TE policy using automated-steering principles without imposing the service route's prefix label (see Automated Steering, on page 390). BGP ignores the programming of the label associated with a prefix path (for example, 6PE/VPN label) when recursing onto the BSID of an SR-TE policy with this feature enabled.

This feature allows you to deploy a Use Case: Centralized BGP EPE for 6PE in an SR-MPLS Network.

Usage Guidelines and Limitations

This functionality applies to local/manually configured SR-TE candidate-paths.

This functionality does not apply to on-demand SR-TE candidate-paths triggered by ODN.

This functionality does not apply to SR-TE candidate-paths instantiated via PCEP (PCE-initiated) or BGP-TE.

Configuration

Use the **bgp prefix-path-label ignore** command in SR-TE policy steering config mode to indicate BGP ignores the programming of the label associated with a prefix path (for example, 6PE/VPN label) when recursing onto the BSID of an SR-TE policy with this feature enabled.

```
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy FOO
Router(config-sr-te-policy)# steering
Router(config-sr-te-policy-steering)# bgp prefix-path-label ignore
Router(config-sr-te-policy-steering)# exit
Router(config-sr-te-policy)# color 100 end-point ipv4 0.0.0.0
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-pref)# explicit segment-list sample-sl
```

Verification

The following output displays the SR-TE policy (SR policy color 100, IPv4 null end-point) details showing the ignore prefix label steering behavior:

```
Router# show segment-routing traffic-eng policy candidate-path name FOO private

SR-TE policy database
---------------------

Color: 100, End-point: 0.0.0.0 ID: 3
  Name: srte_c_100_ep_0.0.0.0
  Status:
    Admin: up  Operational: up for 00:10:07 (since Feb  2 12:58:43.554)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Originator: ASN 0 node-address <None> discriminator: 100
      Name: FOO
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      ID: 1
      Source: 20.1.0.100
      Stale: no
      Checkpoint flags: 0x00000000
      Steering:
        Client: BGP
          Disabled: no
          Ignore prefix label: yes
      Explicit: segment-list sample-sl (valid)
        Weight: 1, Metric Type: TE
        IGP area: 2
          SID[0]: 16102 [Prefix-SID: 20.1.0.102, Algorithm: 0]
          SID[1]: 16103 [Prefix-SID: 20.1.0.103, Algorithm: 0]
          SID[2]: 24008 [Adjacency-SID, 15:15:15::4 - 15:15:15::5]
  LSPs:

. . .

  Attributes:
    Binding SID: 24030
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
  Notification to clients:
    Binding SID: 24030
    Bandwidth : 0 Kbps (0 Kbps)
    State: UP
    Flags: [add] [ipv6_caps] [ignore_prefix_label]
    Metric Type: NONE
    Metric Value: 2147483647
    Admin Distance: 100
  ifhandle: 0x00000170
  Source: 20.1.0.100
  Transition count: 1
  LSPs created count: 1
  Reoptimizations completed count: 1
  Retry Action Flags: 0x00000000, ()
  Last Retry Timestamp: never (0 seconds ago)
  Policy reference: 0x1f81e50
```

The following output shows that BGP received the ignore prefix label steering behavior for an SR policy color 100 and IPv4 null end-point:

```
Router# show bgp nexthops 0.0.0.0 color 100 | include "BGP prefix label"

    BGP prefix label: [No]
```

The following output shows the details for a IPv6 BGP global route (151:1::/64) learned from an IPv4 next-hop (6PE) that is steered over an SR policy (BSID 24030). BGP programs the prefix path ignoring its label.

```
Router# show bgp ipv6 labeled-unicast 151:1::/64 detail

BGP routing table entry for 151:1::/64
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               2003         2003
    Local Label: 81718 (no rewrite);
    Flags: 0x003e1001+0x30010000;
Last Modified: Nov 23 16:59:17.891 for 00:00:03
Paths: (400 available, best #1)
  Advertised IPv6 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
    0.3
  Path #1: Received by speaker 0
  Flags: 0xa480000001060205+0x01, import: 0x020
  Advertised IPv6 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
    0.3
  300, (Received from a RR-client)
    5.5.3.1 C:100 (bsid:24030) (admin 100) (metric 2147483647) from 4.4.4.1 (5.5.5.5),
if-handle 0x00000170
      Prefix Label not imposed due to SR policy config
```

Use Case: Centralized BGP EPE for 6PE in an SR-MPLS Network

In this use case, an operator wants to control the egress peering router/egress transit autonomous system (AS) used by selected Internet IPv6 prefixes. To achieve this, SR policies with explicit paths are used to steer traffic to an intended egress peering router and intended egress transit AS. BGP-EPE SIDs are used in order to force traffic onto an intended egress transit AS. Traffic steering follows SR-TE automated-steering principles.

The following key features enable the use-case:

- SR-TE Policy with Explicit Path — Allows the last segment of an SR policy's SID list to be associated with an EPE-enabled BGPv6 neighbor.
- SR-TE Automated Steering Without BGP Prefix Path Label — Allows traffic to an Internet IPv6 prefix to be steered over an SR-TE policy without imposing the 6PE label learned from the original advertising router.

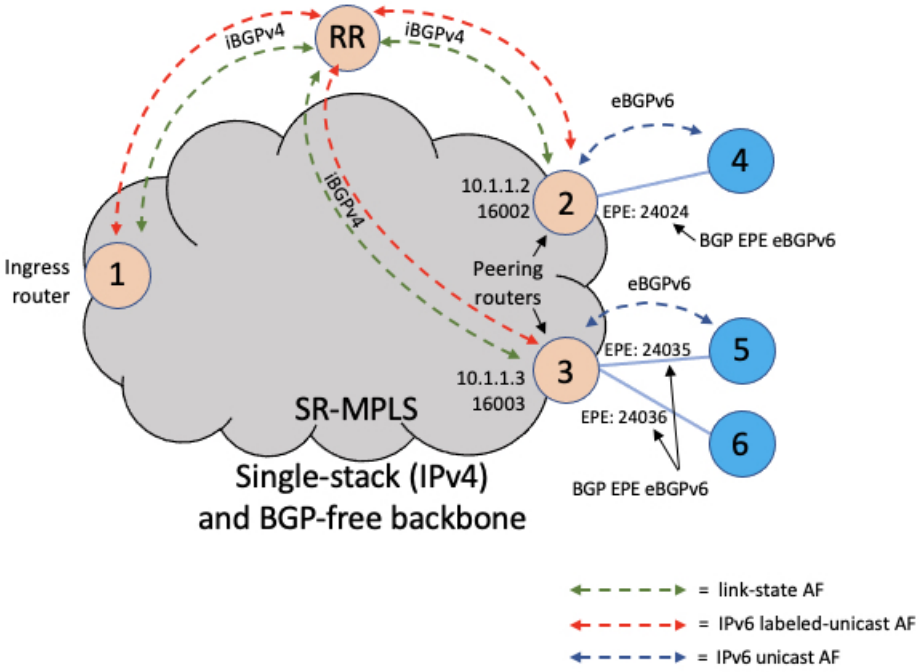
Topology

The below topology shows a single-stack IPv4 SR-MPLS and BGP-free network that delivers Internet IPv6 connectivity using 6PE.

Peering routers 2 and 3 learn IPv6 reachability through transit AS's (ASBR routers 4, 5, 6) via eBGPv6 neighbors.

BGP EPE SIDs are enabled on external BGPv6 neighbors at router 2 (for example, EPE label 24024) and router 3 (for example, EPE labels 24035 and 24036).

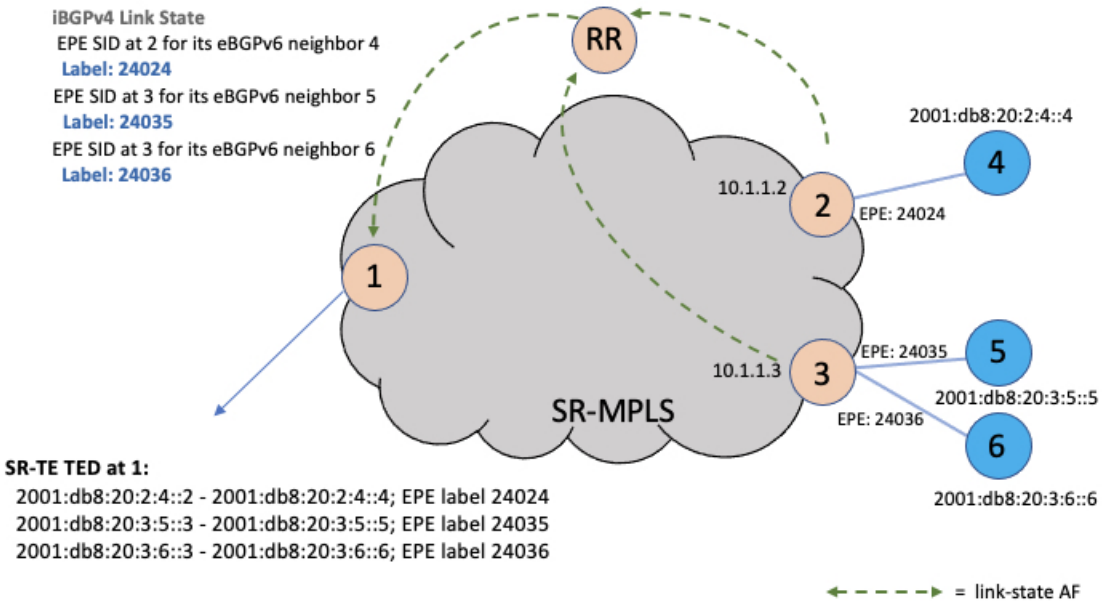
Figure 31: Network Setup



BGP EPE Propagation via BGP-LS

Peering routers 2 and 3 advertise their EPE-enabled neighbors via BGP-LS. As a result, ingress router node 1 learns those EPE-enabled neighbors via BGP-LS. This allows the SR-TE database at the ingress router to include the external links.

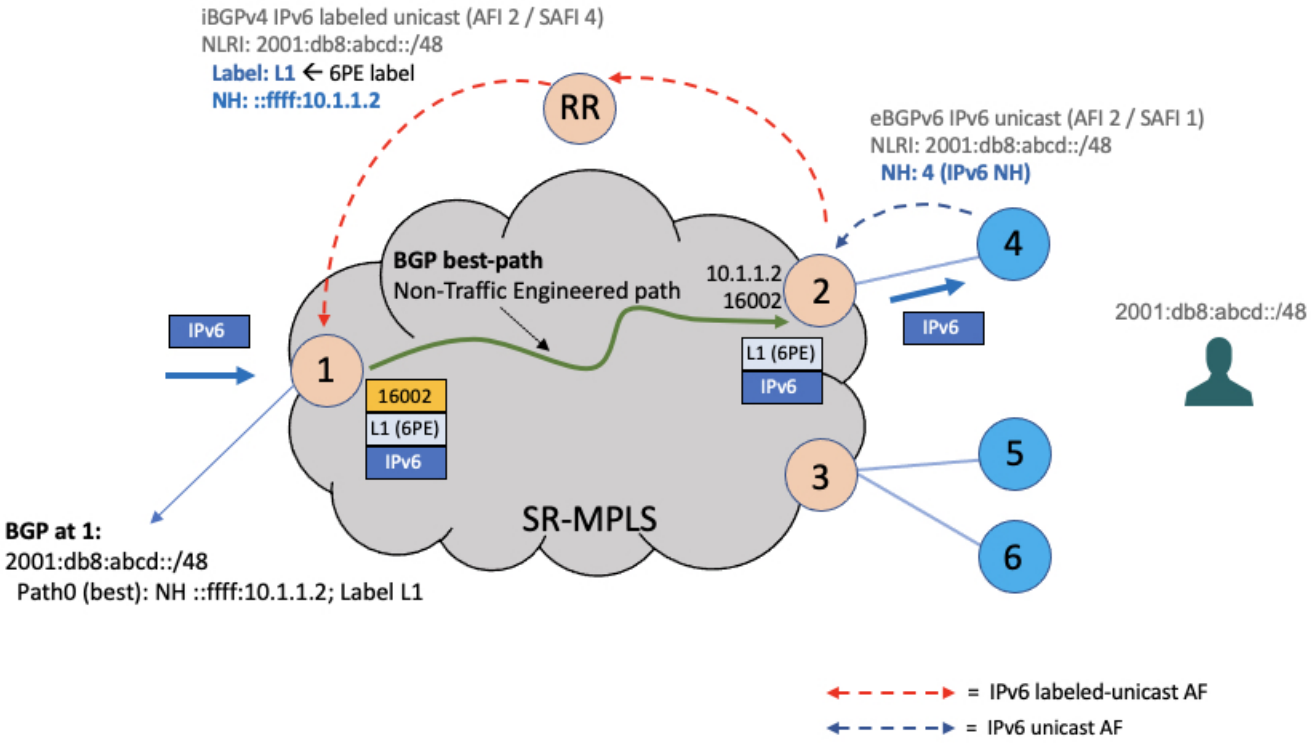
Figure 32: BGP-LS



Steady State (Non-Traffic-Engineered)

At steady state, router 1 selects (as BGP best-path) the path from router 2 for IPv6 prefix 2001:db8:abcd::/48. Traffic to this prefix is sent over the SR-native LSP associated with router 2 (prefix SID 16002) along side the advertised 6PE label.

Figure 33: Steady State



EPE Traffic-Engineered Path

To create a traffic-engineered path that steers traffic to an intended egress peering router/egress transit AS (for example, node 3/ASBR node 6), an SR policy can be configured at ingress border router 1 with the following:

- An IPv4 null (0.0.0.0) end-point, in order to perform color-only automated steering (see Color-Only Automated Steering, on page 391).
- An explicit segment list with SIDs corresponding to the intended egress node (for example, node 3) and the intended egress peering link (for example, ASBR node 6).
- The **bgp prefix-label ignore** steering command in order to indicate BGP to ignore the programming of the 6PE label associated with the prefix path.

```
segment-routing
 traffic-eng
  segment-list sl-to_3-epe_36
   index 10 mpls adjacency 10.1.1.3
   index 20 mpls adjacency 2001:db8:20:3:6::3
  !
  policy FOO
   steering
    bgp prefix-path-label ignore
   !
   color 10 end-point ipv4 0.0.0.0
   candidate-paths
    preference 100
```

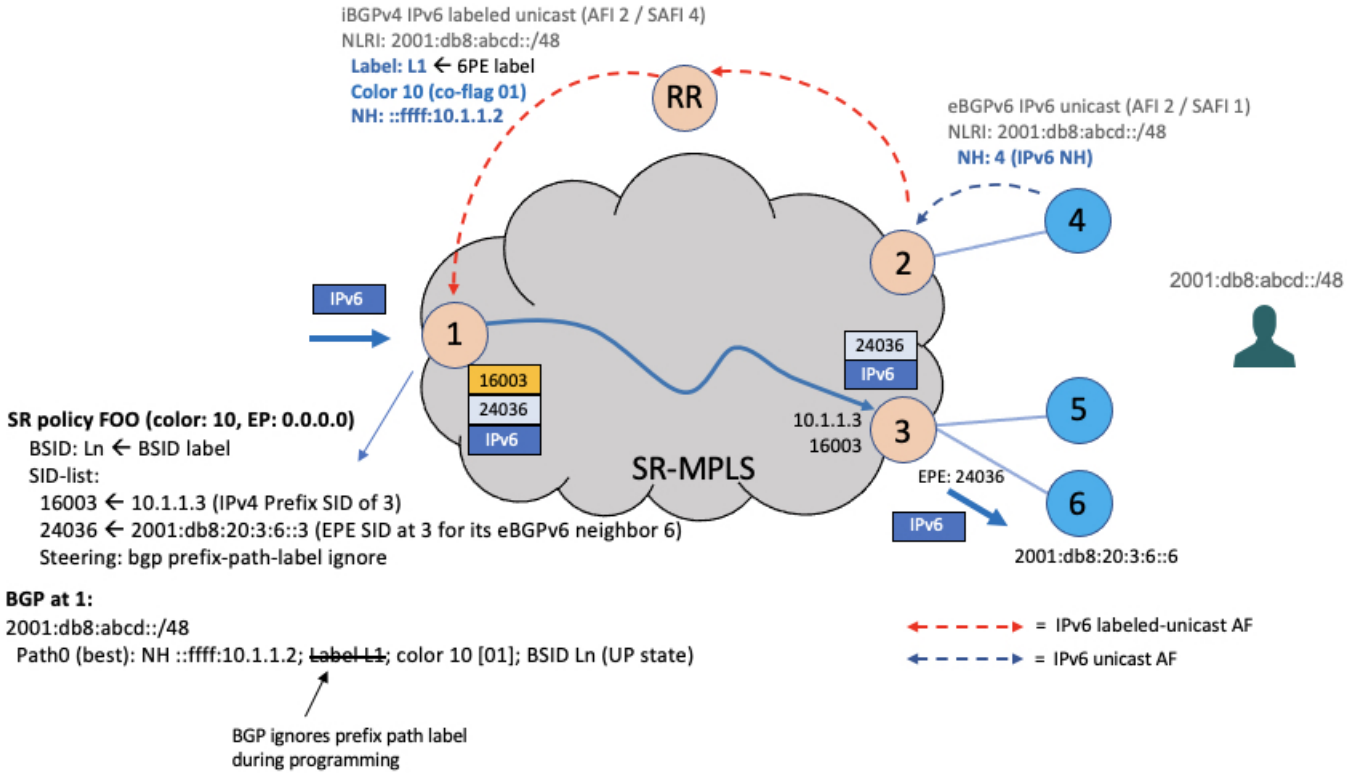
```
explicit segment-list sl-to_3-epe_36
```

When a given IPv6 Internet destination needs to be steered over an intended egress peering router/egress AS, the operator can perform one of the following:

- Advertise a new BGP prefix path from a Route Server that includes a color extended community value equal to the color of the SR-TE policy for the intended egress peering router/egress AS, or
- Apply a color extended community value equal to the color of the SR-TE policy for the intended egress peering router/egress AS at the peering router advertising the best path (for example, node 2), as shown below.

Note The BGP color includes the color-only flag value of 01 in order to allow for color-only automated steering.

Figure 34: EPE Traffic-Engineered Path



The following output depicts the details of the SR-TE policy programmed at the ingress border router node 1 used to send traffic to the egress peering router node 3 and egress AS behind ASBR node 6:

```
Router1# show segment-routing traffic-eng policy candidate-path name FOO private

SR-TE policy database
---------------------

Color: 10, End-point: 0.0.0.0 ID: 3
```

```
  Name: srte_c_10_ep_0.0.0.0
  Status:
    Admin: up  Operational: up for 00:10:07 (since Feb  2 12:58:43.554)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Originator: ASN 0 node-address <None> discriminator: 100
      Name: FOO
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 10
      ID: 1
      Source: 10.1.1.1
      Stale: no
      Checkpoint flags: 0x00000000
      Steering:
        Client: BGP
          Disabled: no
          Ignore prefix label: yes
      Explicit: segment-list sl-to_3-epe_36 (valid)
        Weight: 1, Metric Type: TE
        IGP area: 2
          SID[0]: 16003 [Prefix-SID: 10.1.1.3, Algorithm: 0]
          SID[1]: 24036 [Adjacency-SID, 2001:db8:20:3:6::3 - 2001:db8:20:3:6::6]
  LSPs:

. . .

  Attributes:
    Binding SID: 24030
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
  Notification to clients:
    Binding SID: 24030
    Bandwidth : 0 Kbps (0 Kbps)
    State: UP
    Flags: [add] [ipv6_caps] [ignore_prefix_label]
    Metric Type: NONE
    Metric Value: 2147483647
    Admin Distance: 100
  ifhandle: 0x00000170
  Source: 10.1.1.1
  Transition count: 1
  LSPs created count: 1
  Reoptimizations completed count: 1
  Retry Action Flags: 0x00000000, ()
  Last Retry Timestamp: never (0 seconds ago)
  Policy reference: 0x1f81e50
```

The following output depicts the details of the IPv6 BGP global route (2001:db8:abcd::/48) being steered over the binding SID of the previoulsy shown SR-TE policy (24030):

```
Router1# show bgp ipv6 labeled-unicast 2001:db8:abcd::/48 detail

BGP routing table entry for 2001:db8:abcd::/48
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               2003         2003
    Local Label: 81718 (no rewrite);
```

```
    Flags: 0x003e1001+0x30010000;
Last Modified: Nov 23 16:59:17.891 for 00:00:03
Paths: (1 available, best #1)
  Advertised IPv6 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
    0.3
  Path #1: Received by speaker 0
  Flags: 0xa480000001060205+0x01, import: 0x020
  Advertised IPv6 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv6 Labeled-unicast paths to update-groups (with more than one peer):
    0.3
  300, (Received from a RR-client)
    10.1.1.2 C:10 (bsid:24030) (admin 100) (metric 2147483647) from 10.1.1.100 (10.1.1.2),
 if-handle 0x00000170
      Prefix Label not imposed due to SR policy config
```

Enabling SR-TE with Next-Hop Independent Scaling Optimization

The Next-Hop Independent Scaling Optimization programs an SR-TE policy using a recursive forwarding chain with 2 levels of ECMP. Decoupling next-hop programming results in lower consumption of ASIC resources.

This optimization is disabled by default. Perform the following steps to enable the next-hop independent scaling optimization .

1. Delete any existing SR policies using no form of the command.
2. Enable the optimization using the **segment-routing traffic-eng separate-next-hop** command.

   ```
   Router(config)# segment-routing traffic-eng separate-next-hop
   ```
3. Reload the router.
4. Configure the SR policies again. See Instantiation of an SR Policy, on page 352.

Usage Guidelines and Limitations for Next-Hop Independent Scaling Optimization

The following features are supported:

- SR-TE On-Demand Next Hop/Automated Steering (ODN/AS) for IPv4 BGP and IPv6 BGP global routes
- SR-TE BSID-based AS
- SR-TE head-end path computation
- LFA at SR-TE head-end
- Per-SR policy BSID label counters
- Per-SR policy aggregate counters

- Per-SR policy, per-segment list aggregate counters
- Per-SR policy, per-segment list, per-protocol aggregate counters:
  - Unlabeled IP – Unlabeled IPv4 and IPv6 traffic steered over SR policy
  - Labeled MPLS – Labeled traffic with BSID as top of label stack steered over SR policy

The following features are not supported:

- PCEP at SR-TE head-end
- SR-TE ODN/AS for 6PE, VPNv4, VPNv6 (6vPE), EVPN
- TI-LFA at SR-TE head-end
- SR-TE per-flow policy (PFP)
- SR-TE policy with autoroute-include-based steering
- Static Route Traffic-Steering using SR-TE Policy
- LDP over SR-TE Policy
- Per-SR policy, per-segment list, per-path aggregate counters

Miscellaneous

Segment Routing Encapsulation Object Optimization

Table 50: Feature History Table

Feature Name	Release Information	Feature Description
Segment Routing Encapsulation Object Optimization	Release 7.5.4	The SR Encapsulation object optimization minimizes the forwarding ASIC's Encapsulation resource consumption during programming of an SR-MPLS network, thanks to globally significant label values. With this feature, the forwarding chain of a labeled prefix with ECMP consumes only a single global encapsulation entry in the hardware, instead of an encapsulation entry for each outgoing path. New command: • **hw-module profile cef sropt enable**

When programming an SR-MPLS network, the Segment Routing Encapsulation (Encap) object optimization minimizes the encapsulation resource consumption of the forwarding ASIC. This is because Segment Routing uses globally significant label values.

With this feature, instead of consuming an encapsulation entry for each outgoing path, the forwarding chain of a labeled prefix with ECMP consumes only a single global encapsulation entry.

Usage Guidelines and Limitations

- SR Encap object optimization is triggered only when all ECMP paths of a labeled prefix (primary and backup) perform the same egress action (either all pop or all swap); and have the same outgoing label for the swap egress action. If this condition is not met, then the prefix is programmed with a dedicated Encap object per outgoing path.
- SR Encap object optimization is supported for both labeled IPv4 /32 (SR-MPLSv4) and labeled IPv6 /128 (SR-MPLSv6).
- All paths associated with the prefix (primary and backup) must have the same outgoing label value for SR Encap object optimization to be triggered. For example:
  - For prefixes with LFA backup paths, the SR Encap object optimization is triggered because these backup paths do not require an extra label to be pushed.
  - For prefixes with TI-LFA backup paths requiring extra labels to be pushed, the SR Encap object optimization is not triggered because all the paths associated with the prefix do not have the same outgoing label value.
- Per-label per-interface egress counters are not supported when SR Encap object optimization is enabled. Instead, per-label aggregate egress counters are supported.
- SR MicroLoop Avoidance is not supported when SR Encap object optimization is enabled.

Configuration

Use the **hw-module profile cef sropt enable** command to enable SR Encap object optimization.

Note After you enter this command, you must reload the router.

```
Router(config)# hw-module profile cef sropt enable

In order to activate/deactivate SROPT feature, you must manually reload the chassis/all
line cards

Router(config)# commit

Router(config)# end

Router# show hw-module profile cef

-----------------------------------------------------------------
Knob                          Status          Applied   Action
-----------------------------------------------------------------
CBF Enable                    Unconfigured    N/A       None
CBF forward-class-list        Unconfigured    N/A       None
BGPLU                         Unconfigured    N/A       None
```

```
LPTS ACL                     Unconfigured    N/A       None
Dark Bandwidth               Unconfigured    N/A       None
SR-OPT Enable                Configured      No        Reload
IP Redirect Punt             Unconfigured    N/A       None
IPv6 Hop-limit Punt          Unconfigured    N/A       None
MPLS Per Path Stats          Unconfigured    N/A       None
Tunnel TTL Decrement         Unconfigured    N/A       None
High-Scale No-LDP-Over-TE    Unconfigured    N/A       None
Label over TE counters       Unconfigured    N/A       None
Highscale LDPoTE No SRoTE    Unconfigured    N/A       None
LPTS Pifib Entry Counters    Unconfigured    N/A       None

Router# reload location all
Thu Jan 26 20:15:32.557 UTC
Proceed with reload? [confirm] y


Router# show hw-module profile cef

-----------------------------------------------------------------
Knob                         Status          Applied   Action
-----------------------------------------------------------------
CBF Enable                   Unconfigured    N/A       None
CBF forward-class-list       Unconfigured    N/A       None
BGPLU                        Unconfigured    N/A       None
LPTS ACL                     Unconfigured    N/A       None
Dark Bandwidth               Unconfigured    N/A       None
SR-OPT Enable                Configured      Yes       None
IP Redirect Punt             Unconfigured    N/A       None
IPv6 Hop-limit Punt          Unconfigured    N/A       None
MPLS Per Path Stats          Unconfigured    N/A       None
Tunnel TTL Decrement         Unconfigured    N/A       None
High-Scale No-LDP-Over-TE    Unconfigured    N/A       None
Label over TE counters       Unconfigured    N/A       None
Highscale LDPoTE No SRoTE    Unconfigured    N/A       None
LPTS Pifib Entry Counters    Unconfigured    N/A       None
```

Verification

```
Router# show mpls forwarding labels 19001 detail
Tue Feb  5 19:50:13.992 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
19001  Pop         SR Pfx (idx 3001)  Hu0/1/0/1    18.0.0.2        0
     Updated: Feb  1 23:07:39.595
     Version: 27, Priority: 1
     Label Stack (Top -> Bottom): { Imp-Null }
     NHID: 0x0, Encap-ID: 0x1380900000002, Path idx: 0, Backup path idx: 0, Weight: 0
     MAC/Encaps: 14/14, MTU: 1500
     Outgoing Interface: HundredGigE0/1/0/1 (ifhandle 0x008000c0)
     Packets Switched: 0

  Traffic-Matrix Packets/Bytes Switched: 0/0
  Total Packets/Bytes Switched: 6592788/843876864
```

SR-TE Reoptimization Timers

SR-TE path re-optimization occurs when the head-end determines that there is a more optimal path available than the one currently used. For example, in case of a failure along the SR-TE LSP path, the head-end could detect and revert to a more optimal path by triggering re-optimization.

Re-optimization can occur due to the following events:

- The explicit path hops used by the primary SR-TE LSP explicit path are modified
- The head-end determines the currently used path-option are invalid due to either a topology path disconnect, or a missing SID in the SID database that is specified in the explicit-path
- A more favorable path-option (lower index) becomes available

For event-based re-optimization, you can specify various delay timers for path re-optimization. For example, you can specify how long to wait before switching to a reoptimized path

Additionally, you can configure a timer to specify how often to perform reoptimization of policies. You can also trigger an immediate reoptimization for a specific policy or for all policies.

SR-TE Reoptimization

To trigger an immediate SR-TE reoptimization, use the **segment-routing traffic-eng reoptimization** command in Exec mode:

```
Router# segment-routing traffic-eng reoptimization {all | name  policy}
```

Use the **all** option to trigger an immediate reoptimization for all policies. Use the **name** *policy* option to trigger an immediate reoptimization for a specific policy.

Configuring SR-TE Reoptimization Timers

Use these commands in SR-TE configuration mode to configure SR-TE reoptimization timers:

- **timers candidate-path cleanup-delay** *seconds*—Specifies the delay before cleaning up candidate paths, in seconds. The range is from 0 (immediate clean-up) to 86400; the default value is 120
- **timers cleanup-delay** *seconds*—Specifies the delay before cleaning up previous path, in seconds. The range is from 0 (immediate clean-up) to 300; the default value is 10.
- **timers init-verify-restart** *seconds* —Specifies the delay for topology convergence after the topology starts populating due to a restart, in seconds. The range is from 10 to 10000; the default is 40.
- **timers init-verify-startup** *seconds*—Specifies the delay for topology convergence after topology starts populating for due to startup, in seconds. The range is from 10 to 10000; the default is 300
- **timers init-verify-switchover** *seconds*—Specifies the delay for topology convergence after topology starts populating due to a switchover, in seconds. The range is from 10 to 10000; the default is 60.
- **timers install-delay** *seconds*—Specifies the delay before switching to a reoptimized path, in seconds. The range is from 0 (immediate installation of new path) to 300; the default is 10.
- **timers periodic-reoptimization** *seconds*—Specifies how often to perform periodic reoptimization of policies, in seconds. The range is from 0 to 86400; the default is 600.

Example Configuration

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# timers
Router(config-sr-te-timers)# candidate-path cleanup-delay 600
Router(config-sr-te-timers)# cleanup-delay 60
Router(config-sr-te-timers)# init-verify-restart 120
Router(config-sr-te-timers)# init-verify-startup 600
Router(config-sr-te-timers)# init-verify-switchover 30
Router(config-sr-te-timers)# install-delay 60
Router(config-sr-te-timers)# periodic-reoptimization 3000
```

Running Config

```
segment-routing
 traffic-eng
  timers
   install-delay 60
   periodic-reoptimization 3000
   cleanup-delay 60
   candidate-path cleanup-delay 600
   init-verify-restart 120
   init-verify-startup 600
   init-verify-switchover 30
  !
 !
!
```

Circuit-Style SR-TE Policies

Table 51: Feature History Table

Feature Name	Release Information	Feature Description
Circuit-Style SR-TE Policies	Release 7.8.1	This solution allows Segment Routing to meet the requirements of a connection-oriented transport network, which was historically delivered over circuit-switched SONET/SDH networks. Circuit-style SR-TE policies allow a common network infrastructure to be used for both connection-oriented services and classic IP-based transport. This eliminates the need for multiple parallel networks, which greatly reduces both capital expenditures (CapEx) and operating expenditures (OpEx).

Segment Routing provides an architecture that caters to both connectionless transport (such as IP) as well as connection-oriented transport (such as TDM). IP-centric transport uses the benefits of ECMP and automated/optimum protection from TI-LFA. On the other hand, connection-oriented transport, which was historically delivered over circuit-switched SONET/SDH networks, requires the following:

- End-to-end bidirectional transport that provides congruent forward and reverse paths, predictable latency, and disjointness
- Bandwidth commitment to ensure there is no impact on the SLA due to network load from other services
- Monitoring and maintenance of path integrity with end-to-end 50-msec path protection
- Persistent end-to-end paths regardless of control-plane state

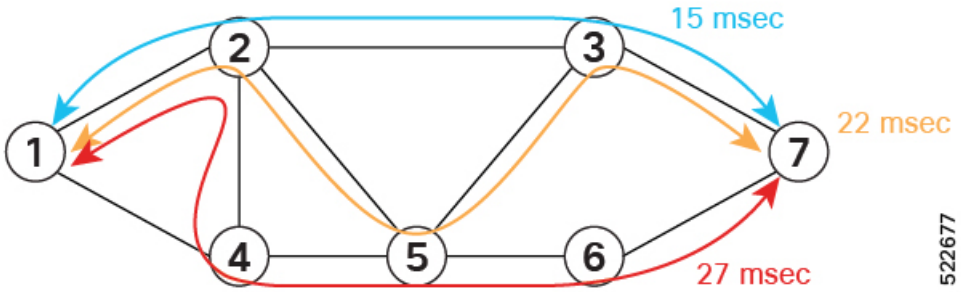
An SR network can satisfy these requirements by leveraging Circuit-Style SR-TE policies (CS-SR policies).

Properties of Circuit-Style SR Policies

CS-SR polices have the following properties:

- **Guaranteed Latency over Non-ECMP Paths**

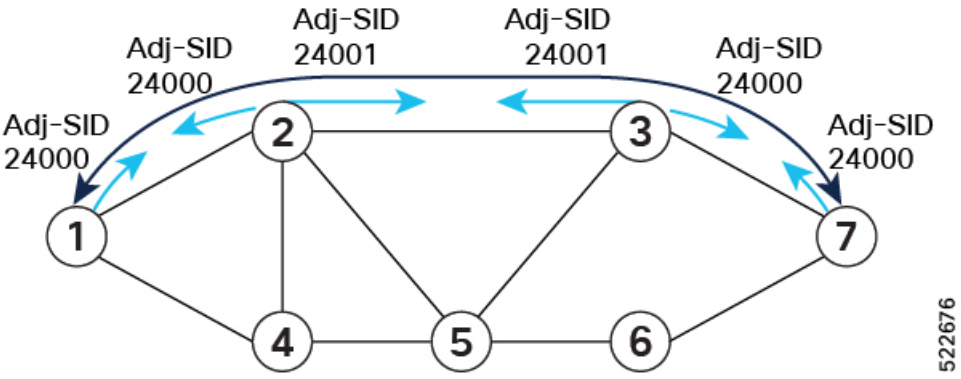
  Consider the network below with three possible paths from node 1 to node 7. Of the three paths, the best end-to-end delay is provided by the blue path (1 -> 2 -> 3 -> 7). The chosen path is then encoded with Adj-SIDs corresponding to the traversed interfaces to avoid any ECMP, and therefore guarantee the latency over the path.



- **Control-Plane Independent Persistency**

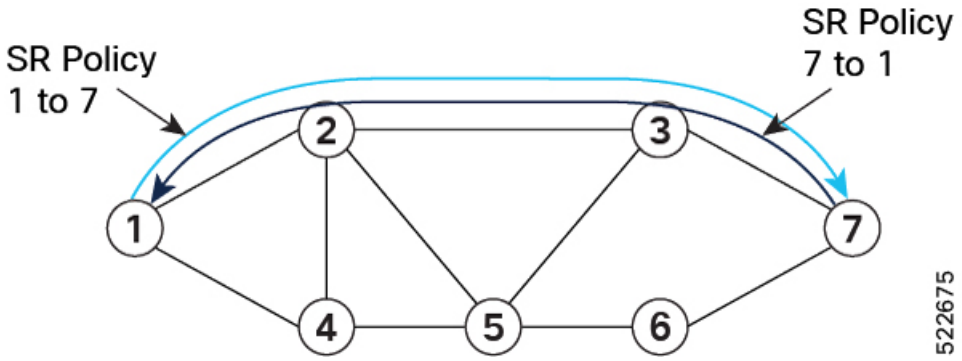
  Adjacency SIDs can provide a persistent path independent from control-plane changes (such as IGP neighbor flaps), as well as network events (such as interface additions or interface flaps) and even the presence of IP on an interface. To achieve this, adjacency SIDs can be manually allocated to ensure persistent values, for example after a node reload event. In addition, adjacency SIDs can be programmed as non-protected to avoid any local TI-LFA protection.

  With the Adj-SIDs depicted in the figure below, the path from node 1 to node 7 is encoded with the segment list of {24000, 24001, 24000}. By manually allocating the same Adj-SID values for other direction, the path from node 7 to node 1 is encoded with the same segment list of {24000, 24001, 24000}.



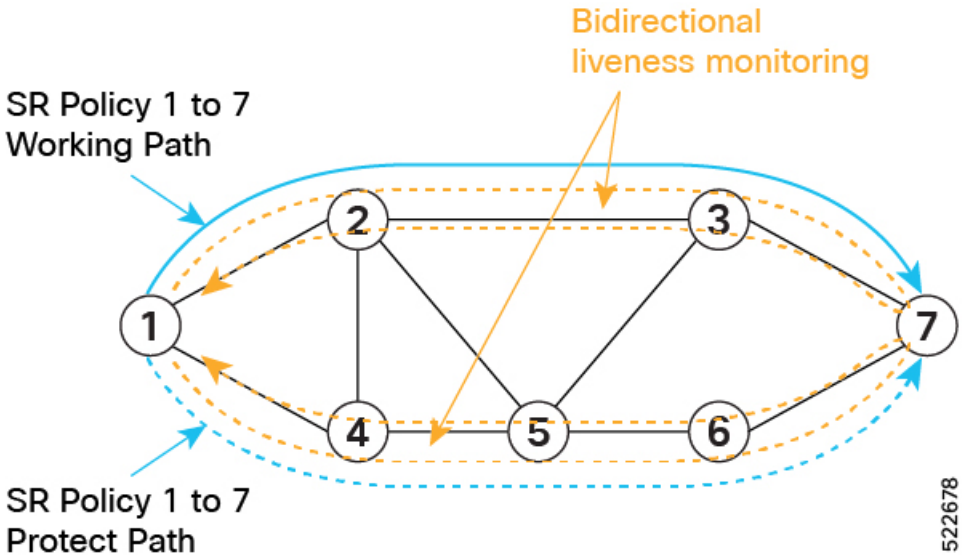
- **Co-Routed Bidirectional Path**

  Forward and return SR Policies with congruent paths are routed along the same nodes/interfaces.



- **Liveness Monitoring with Path Protection Switching**

  Bi-directional liveness monitoring on the working and protect paths ensures fast and consistent switchover, while a protect path is pre-programmed over disjoint facilities.



- **Guaranteed Bandwidth**

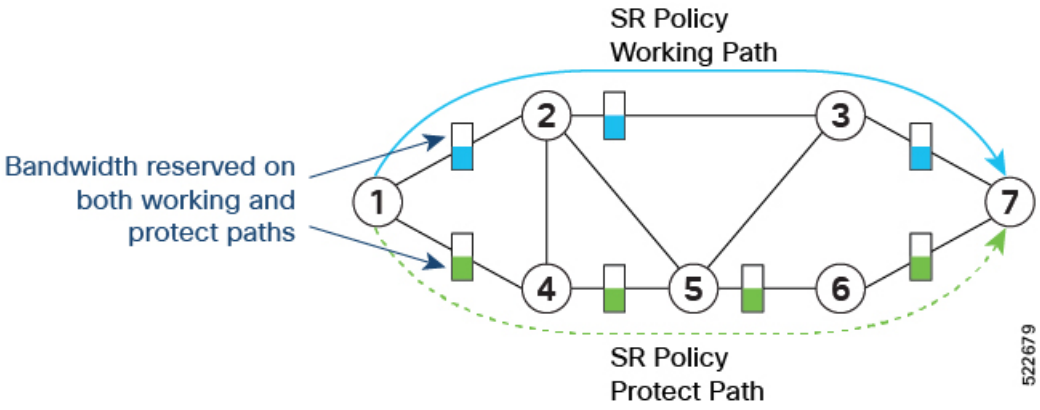
  Most services carried over the CS-SR policy are constant-rate traffic streams. Any packet loss due to temporary congestion leads to bit errors at the service layer. Therefore, bandwidth must be managed very tightly and guaranteed to the services mapped to CS-SR policies.

  A centralized controller manages the bandwidth reservation. The controller maintains the reserved bandwidth on each link based on the traffic usage:

  - Monitors amount of traffic forwarded to each CS-SR policy in the network
  - Uses knowledge of the active path used by the policy
  - Computes the per-link reservable bandwidth accordingly

A per-hop behavior (as documented in RFC3246 [Expedited Forwarding] or RFC2597 [Assured Forwarding]) ensures that the specified bandwidth is available to CS-SR policies at all times independent of any other traffic.

Bandwidth is reserved on both the working and protect paths.



In addition, you can allocate one MPLS-EXP value for traffic steered over the CS SR-TE polices and use QoS (interface queueing) configuration to isolate the circuit traffic from the rest:

- QoS on headend nodes:
  - Define EXP value associated with CS services
  - Enforce rate limiting and perform EXP marking on service ingress interfaces
- QoS on transit nodes:
  - Classify incoming packets based on EXP value associated with CS services.
  - Enforce guaranteed bandwidth for the classified traffic on egress interfaces using bandwidth queues or priority queue with shaper.

Refer to Classify Packets to Identify Specific Traffic chapter in the *Modular QoS Configuration Guide for Cisco 8000 Series Routers*.

Components of the Circuit-Style Solution

CS-SR policy paths are computed and maintained by a stateful PCE. The stateful PCE has a centralized view of the network that it can leverage to compute the co-routed bidirectional end-to-end paths and perform bandwidth allocation control, as well as monitor capabilities to ensure SLA compliance for the life of the CS-SR Policy.

- Centralized Controller
  - Computes the path
  - Encodes the path in a list of Adj-SIDs
  - Monitors and controls bandwidth for SLA guarantee

- QoS configuration on every link to isolate guaranteed traffic

Usage Guidelines and Limitations

Observe the following guidelines and limitations:

- The maximum SID depth (MSD) is 8.
- CS SR policy end-point IP address must be the router-ID of the intended node.
- SR policy path protection is required for both directions.
- SR policy with dynamic path bandwidth constraint is required for both directions and must have the same value for both directions.
- Candidate path (CP) behavior:
  - The working path is associated with the candidate path of the highest preference value.
  - The protect path is associated with the candidate path of the second-highest preference value.
  - The restore path is associated with the candidate path of the third-highest preference value and is configured as "backup ineligible".
  - Candidate paths with the same role in both directions (working, protect, restore) must have the same preference value.
- Bi-directional path behavior:
  - All paths must be configured as co-routed.
  - All paths with the same role in both directions (working, protect, restore) must have the same bi-directional association ID value.
  - The bi-directional association ID value must be globally unique.
- Disjointness constraint:
  - The working and protect paths under the CS SR policy must be configured with a disjointness constraint using the same disjoint association ID and disjointness type.
    - The disjointness association ID for a working and protect path pair in one direction must be globally unique from the corresponding working and protect path pair in the opposite direction.
  - Node and link disjoint constraint types are supported.
  - The disjoint type used in both directions must be the same.
  - The restore path must not be configured with a disjointness constraint.
- Path optimization objectives supported are TE, IGP, and latency.
- The path optimization objective must match across working, protect, and restore paths in both directions.
- Segment type constraint:
  - Working, protect, and restore paths must all be configured with unprotected-only segment type constraint.

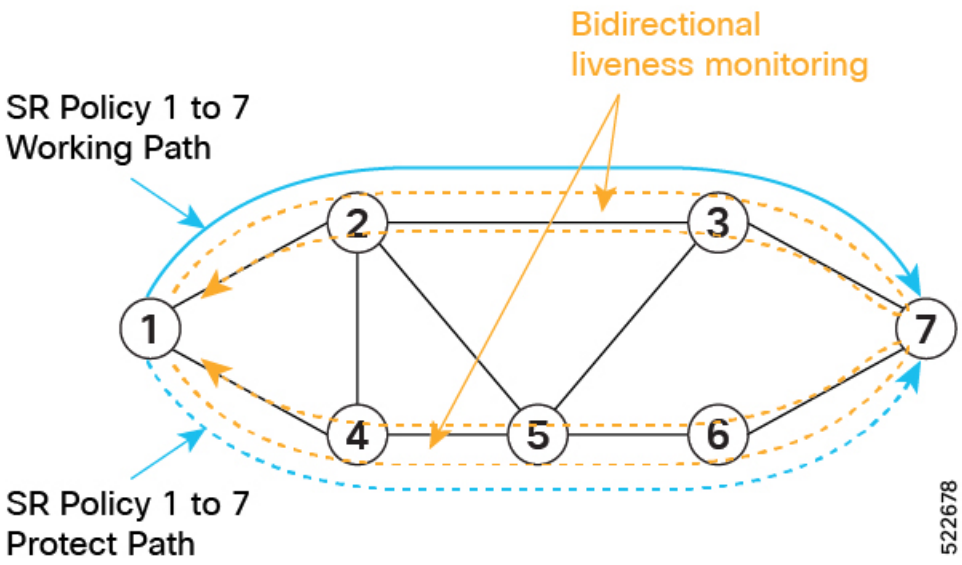
- Working, protect, and restore paths must all be configured with Adj-SID-only segment type constraint.
- To ensure persistency throughout link failure events, manual adjacency SIDs allocated from the SRLB range should be created on all interfaces used by CS policies.

- Revert/recovery behavior:
  - When both working and protect paths are down, the restore path becomes active.
  - The restore path remains active until the working or protect path recovers (partial recovery) and the lock duration timer expires.
  - The lock duration timer is configured under the protect and restore CPs.

- The following functionalities are not supported:
  - Affinity constraint
  - Flex-Algo constraint
  - Metric-bounds constraint

Configure Performance Measurement

Performance Measurement (PM) provides proper detection of candidate path liveness and effective path protection. See SR Policy Liveness Monitoring, on page 550.



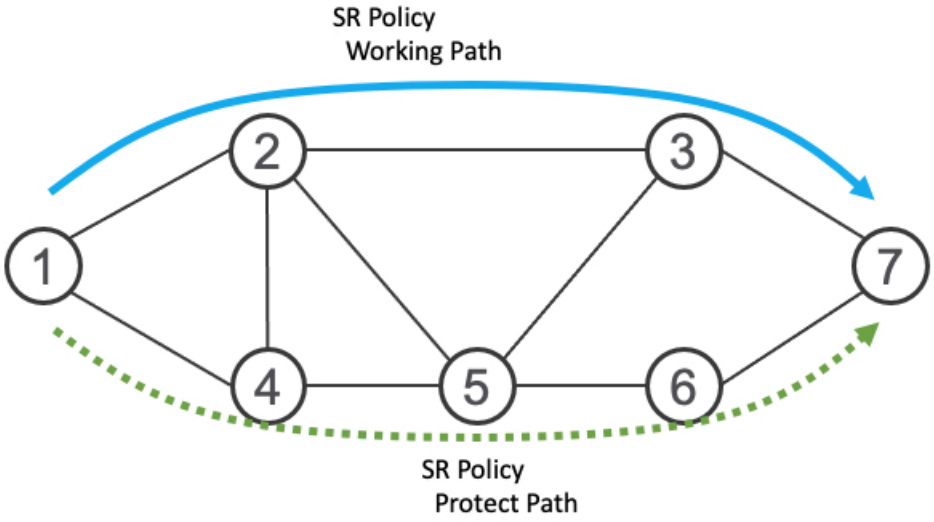
The following example shows how to create a liveness profile for the working and protect paths.

```
Router_1(config)# performance-measurement
Router_1(config-perf-meas)# liveness-profile name profile-WORKING
Router_1(config-pm-ld-profile)# liveness-detection
Router_1(config-pm-ld-profile-ld)# multiplier 3
Router_1(config-pm-ld-profile-ld)# exit
Router_1(config-pm-ld-profile)# probe
```

```
Router_1(config-pm-ld-probe)# tx-interval 30000
Router_1(config-pm-ld-probe)# exit
Router_1(config-pm-ld-profile)# exit
Router_1(config-perf-meas)# liveness-profile name profile-PROTECT
Router_1(config-pm-ld-profile)# liveness-detection
Router_1(config-pm-ld-profile-ld)# multiplier 3
Router_1(config-pm-ld-profile-ld)# exit
Router_1(config-pm-ld-profile)# probe
Router_1(config-pm-ld-probe)# tx-interval 100000
```

Configuring CS SR-TE Policy

The following example shows how to configure a circuit-style SR policy from node 1 to node 7 with three candidate paths: working, protect, and restore.



Create the SR-TE Policy

Configure the CS SR-TE policy

Use the **bandwidth** *bandwidth* command in SR-TE policy configuration mode to configure the guaranteed reservable bandwidth for the policy. The range for *bandwidth* is from 1 to 4294967295 in kbps.

Use the **path-protection** command in SR-TE policy configuration mode to enable end-to-end path protection.

```
Router_1(config)# segment-routing
Router_1(config-sr)# traffic-eng
Router_1(config-sr-te)# policy cs-srte-to-node7
Router_1(config-sr-te-policy)# bandwidth 10000
Router_1(config-sr-te-policy)# color 10 end-point ipv4 10.1.1.7
Router_1(config-sr-te-policy)# path-protection
Router_1(config-sr-te-path-pref-protection)# exit
Router_1(config-sr-te-policy)#
```

Enable Liveness Monitoring Under SR Policy

The following example shows how to enable liveness monitoring under SR Policy, associate the working and protect (backup) liveness-profiles, and configure the invalidation action.

```
Router_1(config)# segment-routing
Router_1(config-sr)# traffic-eng
Router_1(config-sr-te)# policy cs-srte-to-node7
Router_1(config-sr-te-policy)# performance-measurement
Router_1(config-sr-te-policy-perf-meas)# liveness-detection
Router_1(config-sr-te-policy-live-detect)# liveness-profile name profile-WORKING
Router_1(config-sr-te-policy-live-detect)# liveness-profile backup name profile-PROTECT
Router_1(config-sr-te-policy-live-detect)# exit
Router_1(config-sr-te-policy-perf-meas)# exit
Router_1(config-sr-te-policy)#
```

Configure the Working Candidate Path

The working CP has the following characteristics:

- The working CP has the highest preference.
- The working CP uses unprotected-only Adj-SIDs in the segment list.
- The working CP is bidirectional and co-routed.
- The working CP in both directions must have the same bi-directional association ID value.
- The disjoint path constraint for the working CP must have the same group ID and disjoint type as the protect CP.

```
Router_1(config)# segment-routing
Router_1(config-sr)# traffic-eng
Router_1(config-sr-te)# policy cs-srte-to-node7
Router_1(config-sr-te-policy)# candidate-paths
Router_1(config-sr-te-policy-path)# preference 100
Router_1(config-sr-te-policy-path-pref)# dynamic
Router_1(config-sr-te-pp-info)# pcep
Router_1(config-sr-te-path-pcep)# exit
Router_1(config-sr-te-pp-info)# metric
Router_1(config-sr-te-path-metric)# type te
Router_1(config-sr-te-path-metric)# exit
Router_1(config-sr-te-pp-info)# exit
Router_1(config-sr-te-policy-path-pref)# constraints
Router_1(config-sr-te-path-pref-const)# segments
Router_1(config-sr-te-path-pref-const-seg)# protection unprotected-only
Router_1(config-sr-te-path-pref-const-seg)# adjacency-sid-only
Router_1(config-sr-te-path-pref-const-seg)# exit
Router_1(config-sr-te-path-pref-const)# disjoint-path group-id 3 type node
Router_1(config-sr-te-path-pref-const)# exit
Router_1(config-sr-te-policy-path-pref)# bidirectional
Router_1(config-sr-te-path-pref-bidir)# co-routed
Router_1(config-sr-te-path-pref-bidir)# association-id 1100
Router_1(config-sr-te-path-pref-bidir)# exit
Router_1(config-sr-te-policy-path-pref)# exit
Router_1(config-sr-te-policy-path)#
```

Configure the Protect Candidate Path

The protect CP has the following characteristics:

- The protect path is associated with the candidate path of the second-highest preference.
- The protect CP uses unprotected-only Adj-SIDs in the segment list.

- The protect CP is bidirectional and co-routed.
- The protect CP in both directions must have the same bi-directional association ID value.
- The disjoint path constraint for the protect CP must have the same group ID and disjoint type as the working CP.
- When the working path is invalid, the protect path becomes active. After the working path has recovered, the protect path remains active until the default lock duration (300 seconds) expires. You can configure a different lock duration using the **lock duration** *duration* command. The *duration* range is 0 (disabled) to 3000 seconds. If the lock duration is 0 (disabled), then the working path becomes active as soon as it recovers. If *duration* is not specified, the protect path remains active.

```
Router_1(config-sr-te-policy-path)# preference 50
Router_1(config-sr-te-policy-path-pref)# dynamic
Router_1(config-sr-te-pp-info)# pcep
Router_1(config-sr-te-path-pcep)# exit
Router_1(config-sr-te-pp-info)# metric
Router_1(config-sr-te-path-metric)# type te
Router_1(config-sr-te-path-metric)# exit
Router_1(config-sr-te-pp-info)# exit
Router_1(config-sr-te-policy-path-pref)# lock duration 30
Router_1(config-sr-te-policy-path-pref)# constraints
Router_1(config-sr-te-path-pref-const)# segments
Router_1(config-sr-te-path-pref-const-seg)# protection unprotected-only
Router_1(config-sr-te-path-pref-const-seg)# adjacency-sid-only
Router_1(config-sr-te-path-pref-const-seg)# exit
Router_1(config-sr-te-path-pref-const)# disjoint-path group-id 3 type node
Router_1(config-sr-te-path-pref-const)# exit
Router_1(config-sr-te-policy-path-pref)# bidirectional
Router_1(config-sr-te-path-pref-bidir)# co-routed
Router_1(config-sr-te-path-pref-bidir)# association-id 1050
Router_1(config-sr-te-path-pref-bidir)# exit
Router_1(config-sr-te-policy-path-pref)# exit
Router_1(config-sr-te-policy-path)#
```

Configure the Restore Candidate Path

The restore CP has the following characteristics:

- The restore path is associated with the candidate path of the the third-highest preference.
- The restore CP uses unprotected-only Adj-SIDs in the segment list.
- The restore CP is bidirectional and co-routed.
- The restore CP in both directions must have the same bidirectional association ID value.
- The restore CP must be configured with **backup-ineligible**. This configuration prevents the restore CP from being used as a fast reroute backup. The restore path is not computed until both working and protect paths become unavailable.
- Disjointness constraint is not configured on the restore CP.
- If both working and protect paths are unavailable, the restore path becomes active. After either the working or protect path has recovered, the restore path remains active until the default lock duration (300 seconds) expires. You can configure a different lock duration using the **lock duration** *duration* command. The *duration* range is 0 (disabled) to 3000 seconds. If the lock duration is 0 (disabled), then the working

or protect path becomes active as soon as either recovers. If *duration* is not specified, the restore path remains active.

```
Router_1(config-sr-te-policy-path)# preference 10
Router_1(config-sr-te-policy-path-pref)# dynamic
Router_1(config-sr-te-pp-info)# pcep
Router_1(config-sr-te-path-pcep)# exit
Router_1(config-sr-te-pp-info)# metric
Router_1(config-sr-te-path-metric)# type te
Router_1(config-sr-te-path-metric)# exit
Router_1(config-sr-te-pp-info)# exit
Router_1(config-sr-te-policy-path-pref)# backup-ineligible
Router_1(config-sr-te-policy-path-pref)# lock duration 30
Router_1(config-sr-te-policy-path-pref)# constraints
Router_1(config-sr-te-path-pref-const)# segments
Router_1(config-sr-te-path-pref-const-seg)# protection unprotected-only
Router_1(config-sr-te-path-pref-const-seg)# adjacency-sid-only
Router_1(config-sr-te-path-pref-const-seg)# exit
Router_1(config-sr-te-path-pref-const)# exit
Router_1(config-sr-te-policy-path-pref)# bidirectional
Router_1(config-sr-te-path-pref-bidir)# co-routed
Router_1(config-sr-te-path-pref-bidir)# association-id 1010
Router_1(config-sr-te-path-pref-bidir)# exit
Router_1(config-sr-te-policy-path-pref)# exit
Router_1(config-sr-te-policy-path)#
```

Running Config

```
Router_1# show running-config

. . .
segment-routing
 traffic-eng
  policy cs-srte-to-node7
   bandwidth 10000
   color 10 end-point ipv4 10.1.1.7
   path-protection
   !
   candidate-paths
    preference 10
     dynamic
      pcep
      !
      metric
       type te
      !
     !
     lock
      duration 30
     !
     backup-ineligible
     !
     constraints
      segments
       protection unprotected-only
       adjacency-sid-only
      !
     !
     bidirectional
      co-routed
```

```
      association-id 1010
     !
    !
    preference 50
     dynamic
      pcep
      !
      metric
       type te
      !
     !
     lock
      duration 30
     !
     constraints
      segments
       protection unprotected-only
       adjacency-sid-only
      !
      disjoint-path group-id 3 type node
     !
     bidirectional
      co-routed
      association-id 1050
     !
    !
    preference 100
     dynamic
      pcep
      !
      metric
       type te
      !
     !
     constraints
      segments
       protection unprotected-only
       adjacency-sid-only
      !
      disjoint-path group-id 3 type node
     !
     bidirectional
      co-routed
      association-id 1100
     !
    !
   !
   performance-measurement
    liveness-detection
     liveness-profile backup name profile-PROTECT
     liveness-profile name profile-WORKING
     invalidation-action down
    !
   !
  !
 !
!
root
performance-measurement
 liveness-profile name profile-PROTECT
  liveness-detection
   multiplier 3
  !
  probe
```

```
   tx-interval 100000
  !
 !
 liveness-profile name profile-WORKING
  liveness-detection
   multiplier 3
  !
  probe
   tx-interval 30000
  !
 !
!
```

Verification

Use the **show segment-routing traffic-eng policy detail** command to display the details of the CS SR policy:

```
Router_1# show segment-routing traffic-eng policy detail

SR-TE policy database
---------------------

Color: 10, End-point: 10.1.1.7
  Name: srte_c_10_ep_10.1.1.7
  Status:
    Admin: up  Operational: up for 00:02:24 (since Nov 30 08:03:36.588)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Name: cs-srte-to-node7
      Requested BSID: 8000
      PCC info:
        Symbolic name: cfg_cs-srte-to-node7_discr_100
        PLSP-ID: 2
      Constraints:
        Protection Type: unprotected-only
        Maximum SID Depth: 8
        Adjacency SIDs Only: True
      Performance-measurement:
        Reverse-path Label: Not Configured
        Delay-measurement: Disabled
        Liveness-detection: Enabled
          Profile: profile-WORKING
          Invalidation Action: down
          Logging:
            Session State Change: No
        Statistics:
          Session Create    : 1
          Session Update    : 12
          Session Delete    : 4
          Session Up        : 8
          Session Down      : 3
          Delay Notification: 0
          Session Error     : 0
      Dynamic (pce 192.168.0.5) (valid)
        Metric Type: TE,   Path Accumulated Metric: 10
          SID[0]: 24001 [Adjacency-SID, 10.10.10.1 - 10.10.10.2]
        Reverse path:
          SID[0]: 24000 [Adjacency-SID, 10.10.10.2 - 10.10.10.1]
      Protection Information:
        Role: WORKING
        Path Lock: Timed
        Lock Duration: 300(s)
    Preference: 50 (configuration) (protect)
```

```
    Name: cs-srte-to-node7
    Requested BSID: 8000
    PCC info:
      Symbolic name: cfg_cs-srte-to-node7_discr_50
      PLSP-ID: 1
    Constraints:
      Protection Type: unprotected-only
      Maximum SID Depth: 8
      Adjacency SIDs Only: True
    Performance-measurement:
      Reverse-path Label: Not Configured
      Delay-measurement: Disabled
      Liveness-detection: Enabled
        Profile: profile-PROTECT
        Invalidation Action: down
        Logging:
          Session State Change: No
      Statistics:
        Session Create    : 0
        Session Update    : 9
        Session Delete    : 0
        Session Up        : 1
        Session Down      : 0
        Delay Notification: 0
        Session Error     : 0
    Dynamic (pce 192.168.0.5) (valid)
      Metric Type: TE,   Path Accumulated Metric: 10
        SID[0]: 24002 [Adjacency-SID, 11.11.11.1 - 11.11.11.2]
      Reverse path:
        SID[0]: 24003 [Adjacency-SID, 11.11.11.2 - 11.11.11.1]
    Protection Information:
      Role: PROTECT
      Path Lock: Timed
      Lock Duration: 30(s)
  Preference: 10 (configuration) (inactive)
    Name: cs-srte-to-node7
    Requested BSID: 8000
    Constraints:
      Protection Type: unprotected-only
      Maximum SID Depth: 8
      Adjacency SIDs Only: True
    Performance-measurement:
      Reverse-path Label: Not Configured
      Delay-measurement: Disabled
      Liveness-detection: Enabled
        Profile: working
        Invalidation Action: down
        Logging:
          Session State Change: No
      Statistics:
        Session Create    : 0
        Session Update    : 0
        Session Delete    : 0
        Session Up        : 0
        Session Down      : 0
        Delay Notification: 0
        Session Error     : 0
    Dynamic (pce) (inactive)
      Metric Type: TE,   Path Accumulated Metric: 0
    Protection Information:
      Role: RESTORE
      Path Lock: Timed
      Lock Duration: 30(s)
LSPs:
```

```
  LSP[0]:
    LSP-ID: 3 policy ID: 1 (standby)
    Local label: 24037
    State: Standby programmed state
    Performance-measurement:
      Reverse-path Label: Not Configured
      Delay-measurement: Disabled
      Liveness-detection: Enabled
        Profile: profile-WORKING
        Invalidation Action: down
        Logging:
          Session State Change: No
        Session State: up, for 1d12h (since Nov 30 08:03:37.859)
  LSP[1]:
    LSP-ID: 7 policy ID: 1 (active)
    Local label: 24036
    State: Programmed
    Binding SID: 8000
    Performance-measurement:
      Reverse-path Label: Not Configured
      Delay-measurement: Disabled
      Liveness-detection: Enabled
        Profile: profile-WORKING
        Invalidation Action: down
        Logging:
          Session State Change: No
        Session State: up, for 05:42:36 (since Dec  1 15:11:36.203)
Attributes:
  Binding SID: 8000
  Forward Class: Not Configured
  Steering labeled-services disabled: no
  Steering BGP disabled: no
  IPv6 caps enable: yes
  Bandwidth Requested: 10000 kbps
  Bandwidth Current: 10000 kbps
  Invalidation drop enabled: no
  Max Install Standby Candidate Paths: 0
```

Reporting of SR-TE Policies Using BGP- Link State

Table 52: Feature History Table

Feature Name	Release Information	Feature Description
Reporting of SR-TE Policies Using BGP-Link State	Release 24.1.1	BGP- Link State (LS) is a mechanism by which LS and Traffic Engineering (TE) information can be collected from networks and shared with external components (such as, Segment Routing Path Computation Element (SR-PCE) or Crossword Optimization Engine (COE)) using the BGP routing protocol. This feature gathers the Traffic Engineering Policy information that is locally available in a node and advertises it in BGP-LS for SR-MPLS and SRv6. The operators can now take informed decisions based on the information that is gathered on their network's path computation, reoptimization, service placement, network visualization, and so on. The feature introduces these changes: **CLI:** • **distribute link-state** **YANG Data Model:** • New XPaths for `module Cisco-IOS-XR-infra-xtc-agent-cfg.yang` (see GitHub, YANG Data Models Navigator)

This function is achieved using a BGP Network Layer Reachability Information (NLRI) encoding format. BGP-LS consumes structured IGP data (for example, router-id, remote-IP-address of a link, local-IP address of a link, link identifier, and so on). and creates BGP-LS (NLRI) or attributes that BGP or other components like Cisco IOS XR Traffic Controller (XTC) can consume. Current implementation of BGP-LS can report topology using Nodes, Links, and Prefixes.

Modern Segment Routing (SR) networks often use SR Traffic Engineering (SR-TE) to influence the path that each specific traffic takes over the network. SR-TE tunnels can be provisioned manually on the tunnel head, but often they are calculated and provisioned by the central controller. Often operator of the network wants the ability to force the traffic over specific nodes and links.

Now the operators have the option to collect reports of the SR-TE and Policy information that is locally available in a node and advertise it into BGP-LS updates, which can be used by external components. Refer the IEFT for examples. This feature is implemented so that the operators have control over their network's path computation, reoptimization, service placement, network visualization, and so on.

Note Circuit Style (CS) SR policies are reported but without the CS policies' specific attributes, like the bidirectional constraints, per-hop behavior, and so on.

Restrictions to Reporting of SRTE Policies using BGP-LS

Some important points to be aware related to this feature are as follows:

- This feature has high availability, ensuring BGP-LS reporting at all times.
- This feature works with PCE State Sync. The policy information learned via the state-sync channel is not advertised in BGP-LS by the PCE.
- The feature works with PCE redundancy. Both PCEs advertise the BGP-LS policy information. The consumers can select only one policy based on the BGP best path logic.

Configure Reporting of SRTE Policies using BGP-LS

The reporting of policies to BGP-LS is disabled by default. Configuring the **distribute link-state** under the SR-PCE or SR-TE configuration enables this feature. Once enabled, all the existing SR policies or Candidate Path (CPs) are encoded to BGP-LS. You can disable this feature by removing the configuration and all the SR policies or CPs are withdrawn from BGP which deletes all of the previously encoded SR policies or CPs.

Note When SR policies that are reporting in BGP-LS are enabled by the operator, the Head End only reports the Active Candidate Path (CPs) (CPs installed in the forwarding). There are monitoring use cases that require reporting of inactive CPs. The following CLI is needed to report inactive CPs.

For both SR-TE and SR-PCE, you need to use the global CLI to enable the reporting of policies or Circuit Style (CS) to BGP-LS:

Configuration Example

Configure the following command to enable reporting and syncing of SR policies in BGP-LS at the Head End:

```
Router# config
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# distribute link-state
Router(config-sr-te-distribute-ls)# report-candidate-path-inactive
Router(config-sr-te-distribute-ls)# commit
Router(config-sr-te-distribute-ls)# exit
```

Note The **report-candidate-path-inactive** command is an optional command to report inactive CPs.

Running Configuration

This is a sample running configuration which shows that you have configured BGP-LS reporting feature.

```
segment-routing
 traffic-eng
  distribute link-state
   report-candidate-path-inactive
  !
 !
!
```

Verification

Use the **show pce segment-routing traffic-eng policy private** and **show pce distributed-ls events** to verify the configuration.

```
Router#show pce segment-routing traffic-eng policy private

PCE's policy database:
----------------------
PCC Address: 192.168.2.1
  Color: 100, Endpoint: 192.0.2.1
    Name: srte_c_100_ep_192.0.2.1
    Self Pointer: 0x317d7f0
    Active C-path Pointer: 0x317d9b0
    Candidate-paths:
      Symbolic-name: cfg_foo_discr_100 (Active)
        PLSP-ID: 1
        NB-API Notification pending flag: 0
        Self Pointer: 0x317d9b0
        Tunnel Pointer: 0x317d4e0
        Policy Pointer: 0x317d7f0
        Cached distribute LS element:
          Sense: TRUE
          Refcount: 1
          Is on queue: FALSE
          Node identifiers:
            Protocol: 9 router ID: 192.0.2.1
          Policy identifiers:
            Color: 100 Endpoint: 192.0.2.1
            Flags: 0x00
          Candidate path identifiers:
            Originator: 0.0.0.0 protocol: 3 ASN: 0
            Flags: 0x00
            Discriminator: 100
          Candidate path attributes:
            Name: foo
            Policy name: srte_c_100_ep_192.0.2.1
            State:
              Priority: 0 flags: 0x5800 (active, evaluated, valid-sid-list)
              Preference: 100
            BSID:
              Flags: 0x4000 (alloc)
              BSID: 24021
              Specified BSID: 0
            Constraints:
              Bitfield: 0x0020 flags: 0x4000 (protected-only) MT-ID: 0
              Algorithm: 0
              Bandwidth: 0 kbps
              Metric constraints[0]:
                Type: 0 flags: 0x80 (optimization)
                Margin: 0 bound: 0
              Metric constraints[1]:
                Type: 4 flags: 0x10 (bound)
                Margin: 0 bound: 10
```

```
           Segment lists:
             Segment list[0]:
               Flags: 0x3800 (computed, verified, first-seg-resolved) MT-ID: 0 algorithm:
 0 weight: 0
               Metric[0]:
                 Type: 0 flags: 0x10 (value)
                 Margin: 0 bound: 0 value: 10
               Segments:
                 Segment[0]:
                   Bitfield: 0x0000 type: 3 flags: 0x8000 (sid-present)
                   SID: 102000
                   Descriptor:
                     Algorithm: 0
                     Local address: 192.0.2.1 [0] remote address: 0.0.0.0 [0]
```

```
Router#show pce distribute-ls events

Distribute LS events:
----------------
Event history (oldest first):
  Time                    Event
  Apr 11 01:50:48.985     [UID: 0] SR CP NLRI: node ID: SR RID 192.0.2.1, ls_id 0, asn 0,
bitf 0x00000200 policy ID: endpoint: 192.168.0.2 color: 100 CP ID: originator: config 0.0.0.0
 asn: 0 discriminator: 100 (oper: add)
  Apr 11 01:50:50.046     [UID: 1] SR CP NLRI: node ID: SR RID 192.0.2.1, ls_id 0, asn 0,
bitf 0x00000200 policy ID: endpoint: 192.168.0.2 color: 100 CP ID: originator: config 0.0.0.0
 asn: 0 discriminator: 100 (oper: add)
RP/0/0/CPU0:rtrX#show pce distribute-ls summary
Tue Apr 11 02:03:36.289 PDT
Distribution enabled: yes
Connected to LS-LIB: yes
Encode queue size: 0
Estimated encoding rate: 17280/s
NLRIs encoded to LS-LIB:
  SR candidate path: 2 added, 0 removed, 0 errored, 0 replaced
Element stats:
  SR candidate path: 1 (watermark: 2)
Throttle timer:
  Running: no
```

Below is the example show output for SRv6.

```
Router# show segment-routing traffic-eng policy color 200 private

SR-TE policy database
---------------------

Color: 200, End-point: 192::2 ID: 2
  Name: srte_c_200_ep_192::2
  Status:
    Admin: up  Operational: up for 00:01:07 (since Mar  6 11:17:42.580)
  Candidate-paths:
    Preference: 100 (configuration) (active)
      Originator: ASN 0 node-address <None> discriminator: 100
      Name: bar
      Requested BSID: dynamic
      PCC info:
        Symbolic name: cfg_bar_discr_100
        PLSP-ID: 2
        Is orphan: no
        State timer:
          Running: no
      Constraints:
        Protection Type: protected-preferred
```

```
    Maximum SID Depth: 10
  ID: 1
  Source: 192::1
  Stale: no
  Checkpoint flags: 0x00000000
  Path Type: SRV6
  Performance-measurement:
    Reverse-path segment-list:
    Delay-measurement: Disabled
    Liveness-detection: Disabled
  Dynamic (pce 192.168.0.3) (valid)
    Metric Type: IGP,   Path Accumulated Metric: 10
    IGP area: 0
      SID[0]: fccc:ccc1:2::/48 Behavior: uN (PSP/USD) (48)
              Format: f3216
              LBL:32 LNL:16 FL:0 AL:80
              Address: 192::2
  SRv6 Information:
    Locator: loc1Algo0
    Binding SID requested: Dynamic
    Binding SID behavior: uB6 (Insert.Red)
  Cached distribute LS element:
    Sense: TRUE
    Refcount: 1
    Is on queue: FALSE
    Node identifiers:
      Protocol: 9 router ID: 192::1
    Policy identifiers:
      Color: 200 Endpoint: 192::2
      Flags: 0x80 (endpoint-v6)
    Candidate path identifiers:
      Originator: 0.0.0.0 protocol: 3 ASN: 0
      Flags: 0x00
      Discriminator: 100
    Candidate path attributes:
      Name: bar
      Policy name: srte_c_200_ep_192::2
      State:
        Priority: 0 flags: 0x5A00 (active, evaluated, valid-sid-list, delegated)
        Preference: 100
      SRv6 BSID:
        Flags: 0x8000 (alloc)
        BSID: fccc:ccc1:1:e018::
        Specified BSID: ::
        Endpoint:
          Endpoint function: 71 flags: 0x00 algorithm: 0
        Structure:
         Locator block length: 32 locator node length: 16 function length: 16 arguments
length: 0
      Constraints:
        Bitfield: 0x0020 flags: 0xC000 (dataplane-v6, protected) MT-ID: 0
        Algorithm: 0
        Bandwidth: 0 kbps
        Metric constraints[0]:
          Type: 0 flags: 0x80 (optimization)
          Margin: 0 bound: 0
        Metric constraints[1]:
          Type: 4 flags: 0x10 (bound)
          Margin: 0 bound: 10
      Segment lists:
        Segment list[0]:
          Flags: 0xB800 (dataplane-v6, computed, verified, first-seg-resolved) MT-ID:
0 algorithm: 0 weight: 1
```

```
                Metric[0]:
                  Type: 0 flags: 0x10 (value)
                  Margin: 0 bound: 0 value: 10
                Segments:
                  Segment[0]:
                    Bitfield: 0x0003 type: 9 flags: 0x8000 (sid-present)
                    SID: fccc:ccc1:2::
                    Descriptor:
                      Algorithm: 0
                      Local address: 192::2 [0] remote address: :: [0]
                    Endpoint:
                      Endpoint function: 48 flags: 0x00 algorithm: 0
                    Structure:
                      Locator block length: 32 locator node length: 16 function length: 0
arguments length: 80
  LSPs:
    LSP[0]:
      LSP-ID: 2 policy ID: 2 (active)
      State: Programmed
      Binding SID: fccc:ccc1:1:e018::
      Install timer:
        Running: no
      Cleanup timer:
        Running: no
      Delete timer:
        Running: no
      Revert timer:
        Running: no
      SM chain:
        Init -> Egress paths
        Egress paths pending -> BSID RW
        BSID rewrite pending -> Success
      Forwarding flags: 0x00000008
      Candidate path ID: 1
      Flags:
      SL-ID:
        Sent/Received/Transferred: 1/1/0
      SLs:
        SL[0]:
          Name: dynamic
          Type: Dynamic PCE
          Checkpoint id: 1
          NH SRV6 SID: fccc:ccc1:2::
          SL ID: 0xa000001
          Flags:
          Normalized Weight: 1
          ENS: 1
          Paths:
            Path[0]:
              Interface version: 1
              Flags:
              Outgoing interface: Gi0/2/0/0
              Weight: 1
              SID stack:
              Total SID count: 0
              Underlay Normalized Weight: 1
            Path[1]:
              Interface version: 1
              Flags:
              Outgoing interface: Gi0/2/0/1
              Weight: 1
              SID stack:
              Total SID count: 0
              Underlay Normalized Weight: 1
```

```
            Path[2]:
              Interface version: 1
              Flags:
              Outgoing interface: Gi0/2/0/2
              Weight: 1
              SID stack:
              Total SID count: 0
              Underlay Normalized Weight: 1
  Attributes:
    Binding SID: fccc:ccc1:1:e018::
    Forward Class: Not Configured
    Steering labeled-services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
    Path Type: SRV6
  Notification to clients:
    Binding SID: fccc:ccc1:1:e018::
    Bandwidth : 0 Kbps (0 Kbps)
    State: UP
    Flags: [add] [ipv6_caps]
    Metric Type: IGP
    Metric Value: 10
    Admin Distance: 30
  ifhandle: 0x00000000
  Source: 192::1
  Transition count: 1
  LSPs created count: 1
  Reoptimizations completed count: 1
  Retry Action Flags: 0x00000000, ()
  Last Retry Timestamp: never (0 seconds ago)
  Policy reference: 0x1222c10
  Event history (oldest first):
    Time                    Event
    Mar  6 11:17:40.563     POLICY CREATE
    Mar  6 11:17:40.564     (x3) CP PCRPT: 1 hops:
    Mar  6 11:17:41.071     CP PCUPD: CP-ID: 1, path change: true, notify: true, hops:
fccc:ccc1:2::
    Mar  6 11:17:41.072     CP PCRPT: 1 hops: fccc:ccc1:2::
    Mar  6 11:17:41.072     LSP CREATE: 2, need BSID RW: true, BSID: ::
    Mar  6 11:17:41.072     CP CHANGE: PREF: 100 [PROTO: 30, ORIGIN: 0/<None>, DISC: 100]
    Mar  6 11:17:42.579     SRv6 BSID RW REQ: ID 2
    Mar  6 11:17:42.580     SRv6 BSID RW RES: ID 2 Status: success
    Mar  6 11:17:42.580     LSP PCRPT: 2, hops: fccc:ccc1:2::
    Mar  6 11:17:42.580     CP PCRPT REMOVE: 1
    Mar  6 11:17:42.580     IM STATE CHANGE: UNKNOWN to UP, count: 0


Router# show pce segment-routing traffic-eng policy private

PCE's policy database:
----------------------

PCC Address: 192.168.2.1
  Color: 200, Endpoint: 192::2
    Name: srte_c_200_ep_192::2
    Self Pointer: 0x26cd890
    Active C-path Pointer: 0x26cda00
    Candidate-paths:
      Symbolic-name: cfg_bar_discr_100 (Active)
        PLSP-ID: 2
        NB-API Notification pending flag: 0
```

```
        Self Pointer: 0x26cda00
        Tunnel Pointer: 0x26cd580
        Policy Pointer: 0x26cd890
        Cached distribute LS element:
          Sense: TRUE
          Refcount: 1
          Is on queue: FALSE
          Node identifiers:
            Protocol: 9 router ID: 192::1
          Policy identifiers:
            Color: 200 Endpoint: 192::2
            Flags: 0x80 (endpoint-v6)
          Candidate path identifiers:
            Originator: 0.0.0.0 protocol: 3 ASN: 0
            Flags: 0x00
            Discriminator: 100
          Candidate path attributes:
            Name: bar
            Policy name: srte_c_200_ep_192::2
            State:
              Priority: 0 flags: 0x5A00 (active, evaluated, valid-sid-list, delegated)
              Preference: 100
            SRv6 BSID:
              Flags: 0x8000 (alloc)
              BSID: fccc:ccc1:1:e018::
              Specified BSID: ::
              Endpoint:
                Endpoint function: 71 flags: 0x00 algorithm: 0
              Structure:
               Locator block length: 32 locator node length: 16 function length: 16 arguments
length: 0
            Constraints:
              Bitfield: 0x0020 flags: 0xC000 (dataplane-v6, protected) MT-ID: 0
              Algorithm: 0
              Bandwidth: 0 kbps
              Metric constraints[0]:
                Type: 0 flags: 0x80 (optimization)
                Margin: 0 bound: 0
              Metric constraints[1]:
                Type: 4 flags: 0x10 (bound)
                Margin: 0 bound: 10
            Segment lists:
              Segment list[0]:
               Flags: 0xB800 (dataplane-v6, computed, verified, first-seg-resolved) MT-ID:
0 algorithm: 0 weight: 0
                Metric[0]:
                  Type: 0 flags: 0x10 (value)
                  Margin: 0 bound: 0 value: 10
                Segments:
                  Segment[0]:
                    Bitfield: 0x0003 type: 9 flags: 0x8000 (sid-present)
                    SID: fccc:ccc1:2::
                    Descriptor:
                      Algorithm: 0
                      Local address: 192::2 [0] remote address: :: [0]
                    Endpoint:
                      Endpoint function: 48 flags: 0x00 algorithm: 0
                    Structure:
                      Locator block length: 32 locator node length: 16 function length: 0
arguments length: 80
```

```
Router# show bgp link-state link-state | i \\[SP\\]

*>i[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.1][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
*>
[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.3][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
```

```
Router# show bgp link-state link-state
[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.1][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
BGP routing table entry for
[SP][SR][I0x0][N[c100][b0.0.0.0][q192.168.0.1][te192::1]][C[po0x3][f0x80][e192::2][cl0xc8][as0][oa0.0.0.0][di100]]/792
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                128          128
Last Modified: Mar  6 11:17:43.000 for 00:04:09
Last Delayed at: ---
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.168.2.1 (metric 20) from 192.168.2.1 (192.168.0.1)
      Origin IGP, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 128
      Link-state:
                  SRTE-CP-State: Priority: 0 Flags: 0x5a00 Preference: 100
                  SR-Policy-CP-name: bar
                  SR-Policy-name: srte_c_200_ep_192::2
                  SRTE-CP-Constraints: Flags: 0xc000 Mtid: 0 Algorithm: 0
                  SRTE-CP-Constraints-Metric: Type: 0 Flags: 0x80 Margin: 0
                  Bound: 0
                  SRTE-CP-Constraints-Metric: Type: 4 Flags: 0x10 Margin: 0
                  Bound: 10
                  SRTE-Segment-List: Flags: 0xb800 Mtid: 0 Algorithm: 0
                  Weight: 1
                 SRTE-Segment: Segment-Type: 9 Flags: 0x8000 SID: fccc:ccc1:2:: Algorithm:
 0
                   Local-node: 192::2
                  SRv6-Endpoint-Fn: 48 Flags: 0x0 Algo: 0
                  SRv6-SID-Struct: LBL: 32 LNL: 16 FL: 0 AL: 80
                  SRTE-Segment-List-Metric: Type: 0 Flags: 0x10 Margin: 0
                  Bound: 0 Value: 10
                  SRTE-CP-SRV6-BSID: Flags: 0x8000 BSID: fccc:ccc1:1:e018::
                   Specified BSID: ::
                  SRv6-Endpoint-Fn: 71 Flags: 0x0 Algo: 0
                  SRv6-SID-Struct: LBL: 32 LNL: 16 FL: 16 AL: 0
```

SR Policy Path Computation for IPv6

Table 53: Feature History Table

Feature Name	Release Information	Feature Description
SR Policy Path Computation for IPv6	Release 24.1.1	We have introduced the capability for SR Policy to support segment lists with IPv6 addresses, which can be either dynamically computed or explicitly set at the SRTE headend.

You can now use this feature when you want SR Policy to support segment lists with IPv6 addressed.

Note It uses the exiting configurations outlined in *Chapter: Configure SR-TE Policies* in *Segment Routing Configuration Guide for Cisco 8000 Series Routers*, with the supported features detailed in the Usage Guidelines section that follows.

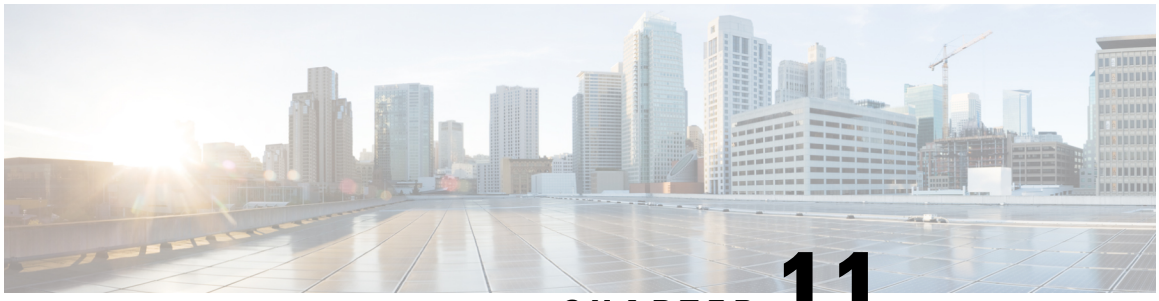
Usage Guidelines and Limitations

The supported features are listed below in the same order as the Table of Contents within the *Chapter: Configure SR-TE Policies* in *Segment Routing Configuration Guide for Cisco 8000 Series Routers*

Supported Features

- Instantiation of SR Policy
  - On-Demand SR Policy – SR On-Demand Next-Hop
  - Manually Provisioned SR Policy
- SR-TE BGP Soft Next-Hop Validation For ODN Policies
- SR-TE Policy Path Types
  - Dynamic Paths
    - Optimization Objectives
    - Constraints
    - Configure SR Policy with Dynamic Path
  - Explicit Paths
    - SR-TE Policy with Explicit Path
    - Explicit Path with Affinity Constraint Validation
    - Explicit Path with Segment Protection-Type Constraint
- Traffic Steering
  - Automated Steering
  - Color-Only Automated Steering
  - Static Route over Segment Routing Policy
- Services Supported
  - IPv4 BGP Global Routes
  - IPv6 BGP Global Routes
- Miscellaneous

- SR-TE Reoptimization Timers
- Sharing the Extended Label Switch Path Array



CHAPTER 11

Segment Routing Tree Segment Identifier

Tree Segment Identifier (Tree-SID) is a tree-building solution that uses a Segment Routing Path Computation Element (SR-PCE) using path computation element protocol (PCEP) to calculate the point-to-multipoint (P2MP) tree using SR policies. Tree-SID uses a single MPLS label for building a multicast replication tree in an SR network. Tree-SID does not require multicast control protocols such as RSVP, mLDP, and PIM.

A P2MP SR policy provides an SR-based TE solution for transporting multicast traffic. It works on existing data-plane (MPLS and IP) and supports TE capabilities and single/multi routing domains. At each node of the tree, the forwarding state is represented by the same segment (using a global Tree-SID specified from the SRLB range of labels). P2MP SR policy prevents transient loop and packet loss when updating the path of a P2MP SR policy.

A P2MP SR policy request contains the following:

- Policy name
- SID for the P2MP Tree (Tree-SID)
- Address of the root-node
- Addresses of the leaf-nodes
- TE optimization criteria (for example, TE or IGP metric) and constraints

- Configure Segment Routing Tree-SID, on page 451
- Running Config, on page 453
- Multicast VPN: Tree-SID MVPN, on page 455
- Multicast: Cisco Nonstop Forwarding for Tree-SID, on page 470
- Multicast VPN IPv6: Dynamic Tree-SID Multicast VPN IPv6, on page 471

Configure Segment Routing Tree-SID

To configure Segment Routing Tree-SID for Point-to-Multipoint (P2MP) SR policies, complete the following configurations:

1. Configure Path Computation Element Protocol (PCEP) Path Computation Client (PCC) on all nodes involved in the Tree-SID path (root, mid-point, leaf)
2. Configure Affinity Maps on the SR-PCE
3. Configure P2MP SR Policy on SR-PCE

4. Configure Multicast on the Root and Leaf Nodes

Configure PCEP PCC on All Nodes in Tree-SID Path

Configure all nodes involved in the Tree-SID path (root, mid-point, leaf) as PCEP PCC. For detailed PCEP PCC configuration information, see the Configure the Head-End Router as PCEP PCC, on page 384 section.

Configure Affinity Maps on the SR-PCE

Use the **affinity bit-map** *COLOR bit-position* command in PCE SR-TE sub-mode to define affinity maps. The bit-position range is from 0 to 255.

```
Router# configure
Router(config)# pce
Router(config-pce)# segment-routing traffic-eng
Router(config-pce-sr-te)# affinity bit-map RED 23
Router(config-pce-sr-te)# affinity bit-map BLUE 24
Router(config-pce-sr-te)# affinity bit-map CROSS 25
Router(config-pce-sr-te)#
```

Configure P2MP SR Policy on SR-PCE

Configure the end-point name and addresses, Tree-SID label, and constraints for the P2MP policy.

Use the **endpoint-set** *NAME* command in SR-PCE P2MP sub-mode to enter the name of the end-point set and to define the set of end-point addresses.

```
Router(config-pce-sr-te)# p2mp
Router(config-pce-sr-te-p2mp)# endpoint-set BAR
Router(config-pce-p2mp-ep-set)# ipv4 1.1.1.2
Router(config-pce-p2mp-ep-set)# ipv4 1.1.1.3
Router(config-pce-p2mp-ep-set)# ipv4 1.1.1.4
Router(config-pce-p2mp-ep-set)# exit
Router(config-pce-sr-te-p2mp)#
```

Use the **policy** *policy* command to configure the P2MP policy name and enter P2MP Policy sub-mode. Configure the source address, endpoint-set color, Tree-SID label, affinity constraints, and metric type.

```
Router(config-pce-sr-te-p2mp)# policy FOO
Router(config-pce-p2mp-policy)# source ipv4 1.1.1.6
Router(config-pce-p2mp-policy)# color 10 endpoint-set BAR
Router(config-pce-p2mp-policy)# treesid mpls 15200
Router(config-pce-p2mp-policy)# candidate-paths
Router(config-pce-p2mp-policy-path)# constraints
Router(config-pce-p2mp-path-const)# affinity
Router(config-pce-p2mp-path-affinity)# exclude BLUE
Router(config-pce-p2mp-path-affinity)# exit
Router(config-pce-p2mp-path-const)# exit
Router(config-pce-p2mp-policy-path)# preference 100
Router(config-pce-p2mp-policy-path-preference)# dynamic
Router(config-pce-p2mp-path-info)# metric type te
Router(config-pce-p2mp-path-info)# root
Router(config)#
```

Configure Multicast on the Root and Leaf Nodes

On the root node of the SR P2MP segment, use the **router pim** command to enter Protocol Independent Multicast (PIM) configuration mode to statically steer multicast flows into an SR P2MP policy.

Note Enter this configuration only on an SR P2MP segment. Multicast traffic cannot be steered into a P2P policy.

```
Router(config)# router pim
Router(config-pim)# vrf name
Router(config-pim-name)# address-family ipv4
Router(config-pim-name-ipv4)# sr-p2mp-policy FOO
Router(config-pim-name-ipv4-srp2mp)# static-group 235.1.1.5 1.1.1.6
Router(config-pim-name-ipv4-srp2mp)# root
Router(config)#
```

On the root and leaf nodes of the SR P2MP tree, use the **mdt static segment-routing** command to configure the multicast distribution tree (MDT) core as Tree-SID from the multicast VRF configuration submode.

```
Router(config)# multicast-routing
Router(config-mcast)# vrf TEST
Router(config-mcast-TEST)# address-family ipv4
Router(config-mcast-TEST-ipv4)# mdt static segment-routing
```

On the leaf nodes of an SR P2MP segment, use the **static sr-policy** *p2mp-policy* command to configure the static SR P2MP Policy from the multicast VRF configuration submode to statically decapsulate multicast flows.

```
Router(config)# multicast-routing
Router(config-mcast)# vrf TEST
Router(config-mcast-TEST)# address-family ipv4
Router(config-mcast-TEST-ipv4)# static sr-policy FOO
```

Running Config

The following example shows how to configure the end point addresses and P2MP SR policy with affinity constraints on SR-PCE.

```
pce
 segment-routing
  traffic-eng
   affinity bit-map
    RED 23
    BLUE 24
    CROSS 25
   !
   p2mp
    endpoint-set BAR
     ipv4 1.1.1.2
     ipv4 1.1.1.3
     ipv4 1.1.1.4
    !
    policy FOO
     source ipv4 1.1.1.6
     color 10 endpoint-set BAR
     treesid mpls 15200
     candidate-paths
      preference 100
       dynamic
        metric
         type te
```

```
        !
       !
      !
      constraints
       affinity
        exclude
         BLUE
        !
       !
      !
     !
    !
   !
  !
 !
!
```

The following example shows how to statically decapsulate multicast flows on the leaf nodes.

```
multicast-routing
 vrf TEST
  address-family ipv4
   static sr-policy FOO
  !
 !
!
```

The following example shows to configure the multicast distribution tree (MDT) core as Tree-SID on the root and leaf nodes.

```
multicast-routing
 vrf TEST
  address-family ipv4
   mdt static segment-routing
  !
 !
!
```

The following example shows how to steer traffic to the SR P2MP policy on the root node.

```
router pim
 vrf TEST
  address-family ipv4
   sr-p2mp-policy FOO
    static-group 232.1.1.5 1.1.1.6
   !
  !
 !
!
```

Multicast VPN: Tree-SID MVPN

Table 54: Feature History Table

Feature Name	Release Information	Feature Description
Multicast VPN: Tree-SID MVPN	Release 7.8.1	With this feature, you can use SR and MVPN for optimally transporting IP VPN multicast traffic over the SP network, using SR-PCE as a controller. With SR's minimal source router configuration requirement, its ability to implement policies with specific optimization objectives and constraints and use SR-PCE to dynamically generate optimal multicast trees (including when topology changes occur in the multicast tree), the SR-enabled SP network can transport IP multicast traffic efficiently.

Prerequisites for Multicast VPN: Tree-SID MVPN

- The underlay OSPF/IS-IS network is configured, and OSPF/IS-IS adjacency is formed between routers, across the network.
- BGP is configured for the network, and BGP adjacency is formed between routers. BGP MVPN configuration information is provided in this feature document.
- To understand the benefits, know-how, and configuration of SR and SR-TE policies, see About Segment Routing and Configure SR-TE Policies.

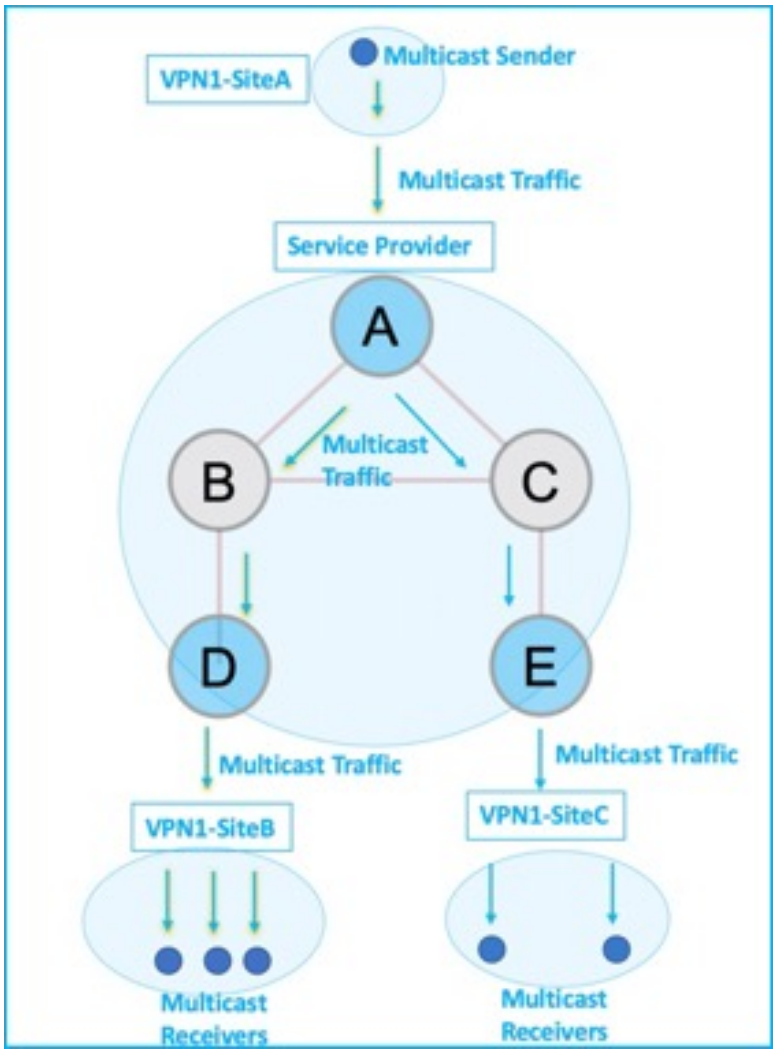
Information About Multicast VPN: Tree-SID MVPN

Typically, a customer's IP VPN is spread across VPN sites. IP VPN customer traffic is sent from one site to another over a VPN Service Provider (SP) network.

When IP multicast traffic within a (BGP/MPLS) IP VPN is transported over an SP network (say, from **VPN1-Site-A** to **VPN1-Site-B**, as shown in the image), the SP network requires protocols and procedures to optimally transport multicast traffic from a multicast sender in Site-A to multicast receivers in Site-B.

This use case explains how to enable SR multicast for an SP network, and efficiently transport IP VPN multicast traffic (sent from **VPN1-Site-A** and) received at PE router A, through to PE routers D and E, towards receivers in sites **VPN1-Site-B** and **VPN1-Site-C**.

Figure 35: IP VPN Multicast Traffic Flow Over An SP Network



To enable the *Multicast VPN: Tree-SID MVPN* feature, the following protocols and software applications are used.

OSPF/IS-IS - The underlay network is created with OSPF/IS-IS routing protocol, and reachability is established across the network. See *Configure Segment Routing for IS-IS Protocol*or *Configure Segment Routing for OSPF Protocol* chapter for details.

BGP Multicast VPN (MVPN) – The PE routers (A, D, and E) are IP VPN end-points for IP multicast traffic arriving at the SP network (at PE router A) and exiting the SP network (at PE routers D and E). So, BGP MVPN is enabled on the PE routers. NSO is used to configure BGP MVPN on the PE routers.

BGP Auto-Discovery (AD) - To enable distributed VPN end-point discovery and C-multicast flow mapping and signalling, BGP AD function is configured on the PE routers. A BGP Auto-Discovery route contains multicast router (loopback IP address) and tree identity (segment ID) information. It carries the information in the Provider Multicast Service Interface (PMSI) Tunnel Attribute (PTA).

C-multicast states are signaled using BGP.

SR - To transport IP multicast traffic between the VPN end-points (PE routers A, D, and E), Provider (or P-) tunnels are used. In a P-tunnel, the PE devices are the tunnel end-points. P-tunnels can be generated using different technologies (RSVP-TE, P2MP LSPs, PIM trees, mLDP P2MP LSPs, and mLDP MP2MP LSPs). In this use case, Segment Routing (SR) is used for its benefits that were noted earlier.

With SR and SR-PCE, a Tree-SID Point-to-Multipoint (P2MP) segment is used to create P-Tunnels for MVPN. You can specify SR policy optimization objectives (such as *metrics*) and constraints (such as *affinity*) in an SR policy and send it to the SR-PCE controller, so that it can dynamically create SR multicast trees for traffic flow.

SR-PCE - This is a controller which, based on the provided SR policy information, computes optimal paths for a multicast tree, and deploys the tree forwarding state on the multicast routers. When a topology change occurs, SR-PCE automatically computes a new, optimal multicast tree, and deploys the new tree forwarding state on the multicast routers.

Overview of Multicast VPN: Tree-SID MVPN

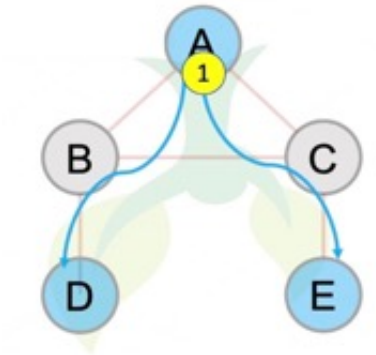
The following sections provide an overview of Tree-SID MVPN. The topology remains the same, with PE routers A, D, and E acting as VPN end-points for carrying IP VPN multicast traffic.

Tree-SID MVPN Overview

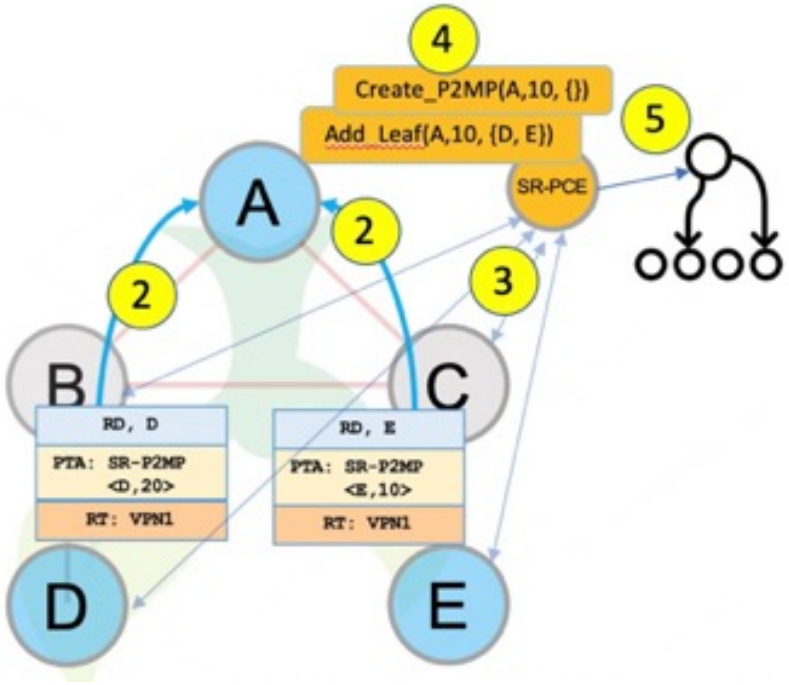
1. For SR, A is designated as the SR head-end router, and D and E are designated as the SR end-points.

   For multicast traffic, A is the root of the SR multicast tree, and D and E are leaf routers of the tree. B and C are the other multicast routers. The objective is to send the IP multicast traffic arriving at A to D and E, as needed

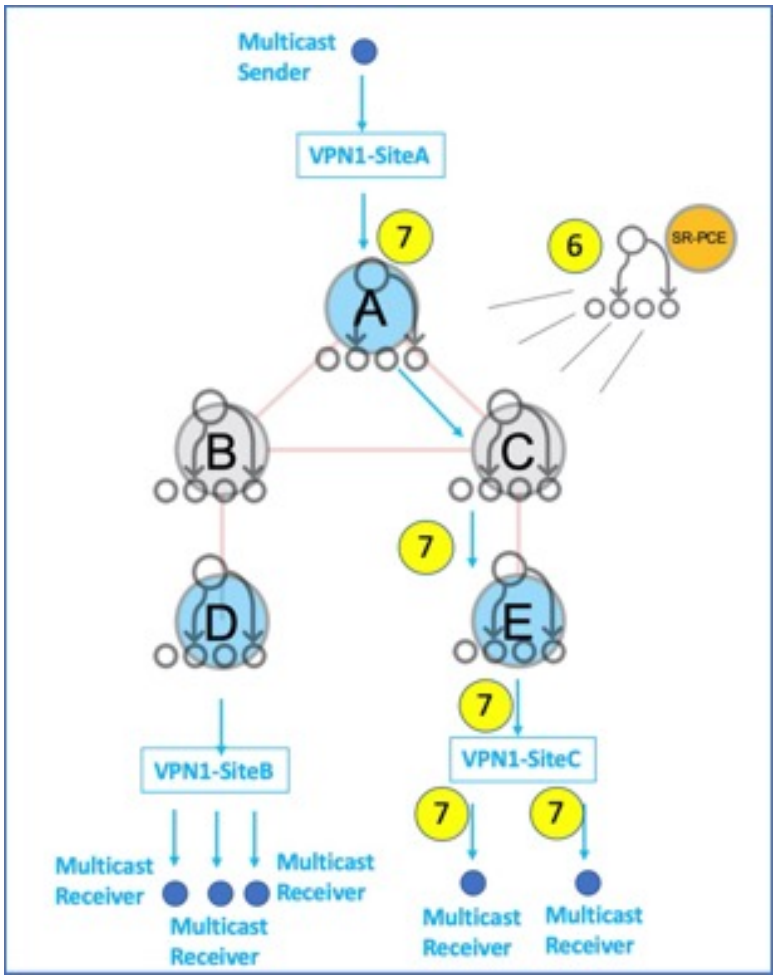
   Figure 36: Multicast Tree



2. A discovers leaf routers' information through BGP MVPN.
3. Path Computation Element Protocol (PCEP) is used for the SR multicast policy communication between A and the SR-PCE server, and communication between PE routers and the SR-PCE server.
4. When the head-end router SR policy is created on A, and PCEP configurations are enabled on the SR-PCE server and all multicast routers, SR-PCE receives the SR policy and leaf router identity information from A.
5. Based on the policy information it receives, including TE objectives and constraints, SR-PCE builds multicast distribution trees in the underlay for efficient VPN traffic delivery.



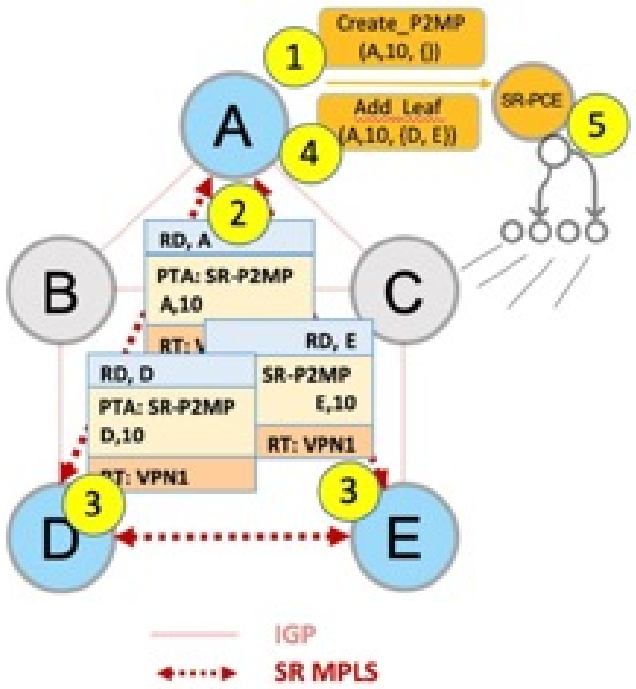
6. SR-PCE assigns an SID for the SR multicast tree policy, and deploys the multicast tree forwarding state on the multicast routers.
7. When IP multicast traffic is sent from VPN1-SiteA to PE router A, it steers it into the SR policy, and sends it towards D and E, which forward it to multicast traffic receivers in the sites VPN1-SiteB and VPN1-SiteC.
8. When a leaf/multicast router is added or removed, PE router A updates the SR multicast policy and sends it to SR-PCE. SR-PCE computes new multicast routes, and deploys the multicast tree forwarding state information on the multicast routers.



SR Multicast Tree Types

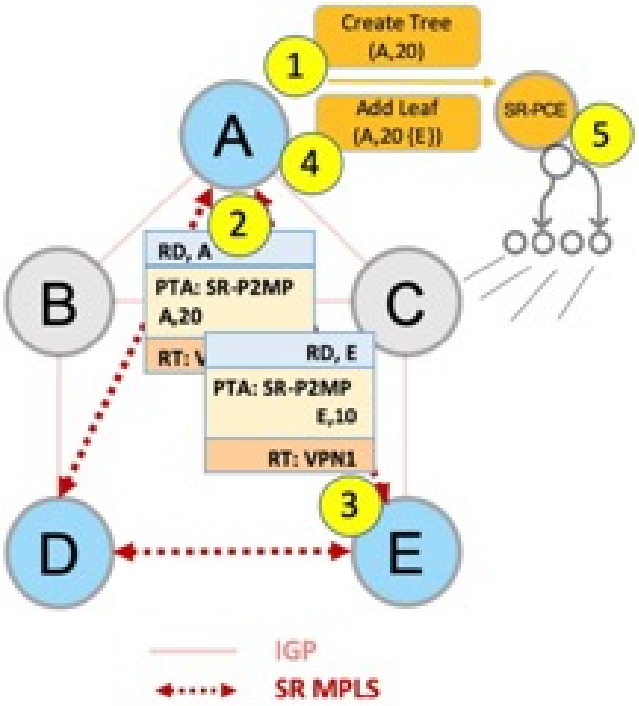
This is an overview of the types of SR multicast trees you can configure, depending on your requirement. You can create a full mesh, on-demand, or optimal multicast tree for IP VPN multicast flow in the SP network.

Figure 37: Full Mesh Multicast Tree



1. A assigns Tree-ID 10 and invokes a Create an SR multicast tree request by sending the multicast router and tree ID information (A, 10) towards SR-PCE.
2. A announces BGP AD Inclusive PMSI (I-PMSI) route with the PTA (A, 10). Inclusive PMSI - Traffic that is multicast by a PE router on an I-PMSI is received by all other PEs in the MVPN. I-PMSIs are generated by Inclusive P-tunnels .
3. A discovers VPN endpoints D and E from their BGP AD Type I-PMSI route messages.
4. A invokes an Add SR multicast leaf router request (for D and E) to SR-PCE.
5. SR-PCE computes and generates the multicast tree forwarding state information on all the routers that are part of the tree.

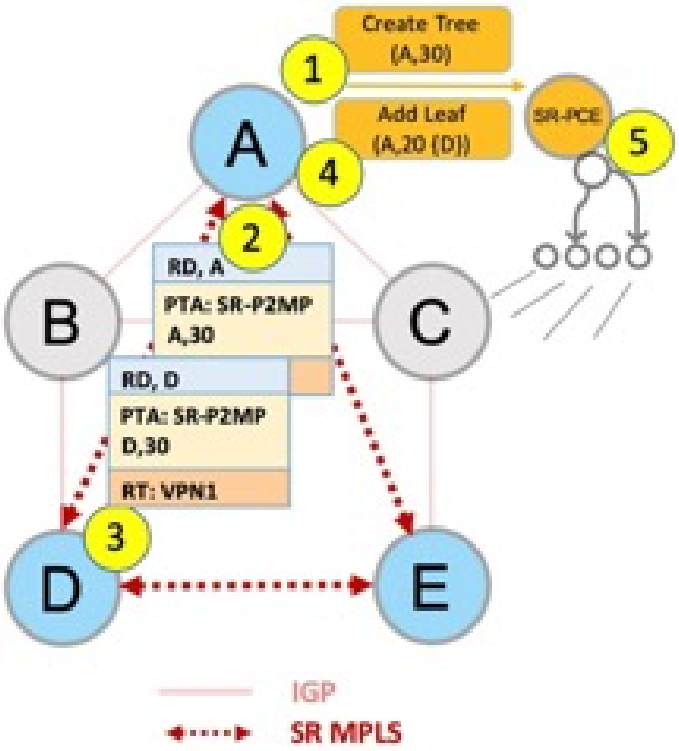
Figure 38: On-Demand SR Multicast Tree



1. A assigns Tree-ID 20 and invokes a Create an SR multicast tree request by sending the multicast router and tree ID information (A, 20) towards SR-PCE.
2. A announces BGP AD Selective PMSI (or S-PMSI) route with PTA (A, 20). A sets the leaf-info-required to discover endpoint interest set.

   Selective PMSI - Traffic multicast by a PE on an S-PMSI is received by some PEs in the MVPN. S-PMSIs are generated by Selective P-tunnels.
3. E has a receiver behind it, and announces a BGP-AD leaf route towards A. A discovers service endpoint E for the on-demand tree.
4. A invokes an Add SR multicast leaf router request (for E) to SR-PCE.
5. SR-PCE computes and generates the multicast tree information for all the routers that are part of the tree.

Figure 39: Optimal Multicast Tree



1. A decides to optimize a flow and assigns Tree-ID 30 and invokes a Create an SR multicast tree request by sending the multicast router and tree ID information (A, 30) towards SR-PCE.
2. A announces BGP AD I-PMSI route with PTA (A,30). A sets the leaf-info-required to discover endpoint interest set.
3. D has a receiver behind it, and announces a BGP-AD leaf route towards A. A discovers service endpoint D for optimized flow.
4. A invokes an Add SR multicast leaf router request (for D) to SR-PCE.
5. SR-PCE computes and generates the multicast tree information for all the routers that are part of the tree.

Configurations

Head End Router Configuration (Router A) - The following configuration is specific to the head end router.

Configure TE Constraints and Optimization Parameters

```
Router# configure terminal
Router(config)# segment-routing traffic-engineering
```

An affinity bit-map is created so that it can be applied to a link or interface.

```
Router(config-sr-te)# affinity-map name 10 bit-position 24
Router(config-sr-te)# commit
```

An affinity (or relationship) is created between the SR policy path and the link color so that SR-TE computes a path that includes or excludes links, as specified. The head-end router automatically follows the actions defined in the ODN template (for color 10) upon the arrival of VPN routes with a BGP color extended community that matches color 10.

```
Router(config)# segment-routing traffic-engineering
Router(config-sr-te)# on-demand color 10 dynamic
Router(config-sr-te-color-dyn)# affinity include-all name red
Router(config-sr-te-color-dyn)# affinity include-any name blue
Router(config-sr-te-color-dyn)# affinity exclude-any name green
Router(config-sr-te-color-dyn)# metric type te
Router(config-sr-te-color-dyn)# commit
```

The SR policy configuration on the head-end router A will be sent to the SR-PCE server, after a connection is established between A and SR-PCE.

Multicast Router Configuration

Configure PCEP Client on Multicast Routers

Associate each multicast router as a client of the SR-PCE server. The **pce address ipv4** command specifies the SR-PCE server's IP address.

```
Router# configure terminal
Router(config)# segment-routing traffic-engineering
Router(config-sr-te)# pcc pce address ipv4 3.3.3.3
Router(config-pcc-pce)# commit
```

SR PCE Server Configuration

Configure Label Range for Multicast Trees

Configure the label range to be used for transporting IP multicast traffic in SP network.

```
Router(config)# pce segment-routing traffic-eng p2mp label-range min 30000 max 60000
Router(config)# commit
```

Disable ECMP load splitting

To disable ECMP load splitting of different trees on the SR-PCE server, configure the **multipath-disable** command.

```
Router(config)# pce segment-routing traffic-eng p2mp multipath-disable
Router(config)# commit
```

Multicast Routing Configuration On PE Routers

The following MVPN configurations are required for VPN end-points, the 3 PE routers.

Configure Default MDT SR P2MP MVPN Profile

In this configuration, an MDT profile of the type *default* is created, and the SR multicast policy with color 10 will be used to send IP multicast traffic, as per the constraints and optimizations of the policy, through the multicast tree.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv4
Router(config-mcast-cust1-ipv4)# mdt default segment-routing mpls color 10
Router(config-mcast-cust1-ipv4)# commit
```

Configure Partitioned MDT SR P2MP MVPN Profile

In this configuration, an MDT profile of the type *partitioned* is created, and the SR multicast policy with color 10 will be used to send IP multicast traffic, as per the constraints and optimizations of the policy, through the multicast tree.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv4
Router(config-mcast-cust1-ipv4)# mdt partitioned segment-routing mpls color 10
Router(config-mcast-cust1-ipv4)# commit
```

The following Data MVPN configuration is required at the Ingress PE (router A) where the multicast flows need to be steered onto the *data* MDT for SR multicast traffic flow.

Note - *Data* MDT can be configured for *Default* and *Partitioned* profiles.

Configure Data MDT for SR P2MP MVPN

In this configuration, an MDT profile of the type *data* is created, and the SR multicast policy with color 10 will be used to send IP multicast traffic, as per the constraints and optimizations of the policy, through the multicast tree.

- As an alternative to the color keyword, you can specify a route policy in the **route-policy** command, and define the route policy separately (as mentioned in the next configuration).
- The **threshold** command specifies the threshold above which a multicast flow is switched onto the data MDT. The **immediate-switch** keyword enables an immediate switch of a multicast flow to the data MDT, without waiting for threshold limit to be crossed.
- The **customer-route-acl** keyword specifies an ACL to enable specific multicast flows to be put on to the data MDT.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv4
Router(config-mcast-cust1-ipv4)# mdt data segment-routing mpls 2 color 10
Router(config-mcast-cust1-ipv4)# commit
```

Route Policy Example

The route policy designates multicast flow-to-SR multicast policy mapping, with different colors.

- With this configuration, IP multicast flows for the 232.0.0.1 multicast group are steered into the SR multicast policy created with the on-demand color 10, while flows for 232.0.0.2 are steered into the policy created with color 20.
- Route policies can also be used to match other parameters, such as source address.

```
Router(config)# route-policy TSID-DATA
Router(config-rpl)# if destination in (232.0.0.1) then
Router(config-rpl-if)# set on-demand-color 10
Router(config-rpl-if)# pass
Router(config-rpl-if)# elseif destination in (232.0.0.2) then
Router(config-rpl-elseif)# set on-demand-color 20
Router(config-rpl-elseif)# pass
Router(config-rpl-elseif)# endif
Router(config-rpl)# end-policy
Router(config)# commit
```

Configure MVPN BGP Auto-Discovery for SR P2MP

The following configuration is required on all PE routers, and is mandatory for *default* MDT, *partitioned* MDT, and *data* MDT.

Configure the BGP Auto-Discovery function for transporting IP multicast traffic.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv4
Router(config-mcast-cust1-ipv4)# bgp auto-discovery segment-routing
Router(config-mcast-cust1-ipv4-bgp-ad)# commit
```

Verification

View MVPN Context Information - You can view MVPN VRF context information with these commands.

View Default MDT Configuration

This command displays SR multicast tree information, including the MDT details (of *default* type, etc), and customer VRF information (route target, route distinguisher, etc).

```
Router# show mvpn vrf vpn1 context

MVPN context information for VRF vpn1 (0x9541cf0)

RD: 1:10 (Valid, IID 0x1), VPN-ID: 0:0
Import Route-targets : 2
  RT:192.168.0.4:0, BGP-AD
  RT:192.168.0.4:17, BGP-AD
BGP Auto-Discovery Enabled (I-PMSI added)
SR P2MP Core-tree data:
  MDT Name: TRmdtvpn1, Handle: 0x4150, idb: 0x956fc30
  MTU: 1376, MaxAggr: 255, SW_Int: 30, AN_Int: 60
  RPF-ID: 3, C:0, O:1, D:0, CP:0
  Static Type : - / -
  Def MDT ID: 524289 (0x93993f0), added: 1, HLI: 0x80001, Cfg: 1/0
  Part MDT ID: 0 (0x0), added: 0, HLI: 0x00000, Cfg: 0/0
  Ctrl Trees : 0/0/0, Ctrl ID: 0 (0x0), Ctrl HLI: 0x00000
```

View Partitioned MDT Configuration

This command displays SR multicast tree information, including the MDT details (of *partitioned* type, etc), and customer VRF information (route target, route distinguisher, etc).

```
Router# show mvpn vrf vpn1 context

MVPN context information for VRF vpn1 (0x9541cf0)

RD: 1:10 (Valid, IID 0x1), VPN-ID: 0:0
Import Route-targets : 2
  RT:192.168.0.4:0, BGP-AD
  RT:192.168.0.4:17, BGP-AD
BGP Auto-Discovery Enabled (I-PMSI added) , MS-PMSI sent
SR P2MP Core-tree data:
  MDT Name: TRmdtvpn1, Handle: 0x4210, idb: 0x956fc30
  MTU: 1376, MaxAggr: 255, SW_Int: 30, AN_Int: 60
  RPF-ID: 1, C:0, O:1, D:0, CP:0
  Static Type : - / -
  Def MDT ID: 0 (0x0), added: 0, HLI: 0x00000, Cfg: 0/0
  Part MDT ID: 524292 (0x9399318), added: 1, HLI: 0x80004, Cfg: 1/0
  Ctrl Trees : 0/0/0, Ctrl ID: 0 (0x0), Ctrl HLI: 0x00000
```

View Partitioned MDT Ingress PE Configuration

This command displays SR multicast tree information on the PE router that receives the multicast traffic on the SP network. The information includes PE router details, MDT details, Tree-SID details, and the specified customer VRF information.

```
Router# show mvpn vrf vpn1 pe

MVPN Provider Edge Router information

VRF : vpn1

PE Address : 192.168.0.3 (0x9570240)
  RD: 0:0:0 (null), RIB_HLI 0, RPF-ID 13, Remote RPF-ID 0, State: 0, S-PMSI: 2
  PPMP_LABEL: 0, MS_PMSI_HLI: 0x00000, Bidir_PMSI_HLI: 0x00000, MLDP-added: [RD 0, ID 0,
```

```
Bidir ID 0, Remote Bidir ID 0], Counts(SHR/SRC/DM/DEF-MD): 0, 0, 0, 0, Bidir: GRE RP Count
 0, MPLS RP Count 0RSVP-TE added: [Leg 0, Ctrl Leg 0, Part tail 0 Def Tail 0, IR added:
[Def Leg 0, Ctrl Leg 0, Part Leg 0, Part tail 0, Part IR Tail Label 0
  Tree-SID Added: [Def/Part Leaf 1, Def Egress 0, Part Egress 0, Ctrl Leaf 0]
  bgp_i_pmsi: 1,0/0 , bgp_ms_pmsi/Leaf-ad: 1/1, bgp_bidir_pmsi: 0, remote_bgp_bidir_pmsi:
0, PMSIs: I 0x9570378, 0x0, MS 0x94e29d0, Bidir Local: 0x0, Remote: 0x0, BSR/Leaf-ad 0x0/0,
 Autorp-disc/Leaf-ad 0x0/0, Autorp-ann/Leaf-ad 0x0/0
 IIDs: I/6: 0x1/0x0, B/R: 0x0/0x0, MS: 0x1, B/A/A: 0x0/0x0/0x0

  Bidir RPF-ID: 14, Remote Bidir RPF-ID: 0
  I-PMSI:   Unknown/None (0x9570378)
  I-PMSI rem:  (0x0)
  MS-PMSI:   Tree-SID [524290, 192.168.0.3] (0x94e29d0)
  Bidir-PMSI:  (0x0)
  Remote Bidir-PMSI:  (0x0)
  BSR-PMSI:  (0x0)
  A-Disc-PMSI:  (0x0)
  A-Ann-PMSI:  (0x0)
  RIB Dependency List: 0x0
  Bidir RIB Dependency List: 0x0
    Sources: 0, RPs: 0, Bidir RPs: 0
```

View Partitioned MDT Egress PE Configuration

This command displays SR multicast tree information on the MVPN egress PE router that sends multicast traffic from the SP network towards multicast receivers in the destination sites. The information includes PE router, Tree-SID, MDT, and the specified customer VRF details.

```
Router# show mvpn vrf vpn1 pe

MVPN Provider Edge Router information

PE Address : 192.168.0.4 (0x9fa38f8)
  RD: 1:10 (valid), RIB_HLI 0, RPF-ID 15, Remote RPF-ID 0, State: 1, S-PMSI: 2
  PPMP_LABEL: 0, MS_PMSI_HLI: 0x00000, Bidir_PMSI_HLI: 0x00000, MLDP-added: [RD 0, ID 0,
Bidir ID 0, Remote Bidir ID 0], Counts(SHR/SRC/DM/DEF-MD): 1, 1, 0, 0, Bidir: GRE RP Count
 0, MPLS RP Count 0RSVP-TE added: [Leg 0, Ctrl Leg 0, Part tail 0 Def Tail 0, IR added:
[Def Leg 0, Ctrl Leg 0, Part Leg 0, Part tail 0, Part IR Tail Label 0
  Tree-SID Added: [Def/Part Leaf 0, Def Egress 0, Part Egress 1, Ctrl Leaf 0]
  bgp_i_pmsi: 1,0/0 , bgp_ms_pmsi/Leaf-ad: 1/0, bgp_bidir_pmsi: 0, remote_bgp_bidir_pmsi:
0, PMSIs: I 0x9f77388, 0x0, MS 0x9fa2f98, Bidir Local: 0x0, Remote: 0x0, BSR/Leaf-ad 0x0/0,
 Autorp-disc/Leaf-ad 0x0/0, Autorp-ann/Leaf-ad 0x0/0
 IIDs: I/6: 0x1/0x0, B/R: 0x0/0x0, MS: 0x1, B/A/A: 0x0/0x0/0x0

  Bidir RPF-ID: 16, Remote Bidir RPF-ID: 0
  I-PMSI:   Unknown/None (0x9f77388)
  I-PMSI rem:  (0x0)
  MS-PMSI:   Tree-SID [524292, 192.168.0.4] (0x9fa2f98)
  Bidir-PMSI:  (0x0)
  Remote Bidir-PMSI:  (0x0)
  BSR-PMSI:  (0x0)
  A-Disc-PMSI:  (0x0)
  A-Ann-PMSI:  (0x0)
  RIB Dependency List: 0x9f81370
  Bidir RIB Dependency List: 0x0
    Sources: 1, RPs: 1, Bidir RPs: 0
```

View Data MDT Information

The commands in this section displays SR multicast tree information for *data* MDTs. The information includes cache, router-local, and remote MDT information.

View Data MDT Cache Information

```
Router# show pim vrf vpn1 mdt cache
Core Source      Cust (Source, Group)              Core Data        Expires
192.168.0.3      (26.3.233.1, 232.0.0.1)           [tree-id 524292]  never

192.168.0.4      (27.3.233.6, 232.0.0.1)           [tree-id 524290]  never

   Leaf AD:  192.168.0.3
```

View Local MDTs Information

```
Router# show pim vrf vpn1 mdt sr-p2mp local

Tree                          MDT             Cache DIP Local VRF Routes  On-demand
Identifier                    Source          Count     Entry Using Cache Color
[tree-id 524290 (0x80002)]    192.168.0.4     1     N   Y     1           10
   Tree-SID Leaf:  192.168.0.3
```

View Remote MDTs Information

```
Router # show pim vrf vpn1 mdt sr-p2mp remote

Tree                          MDT             Cache DIP Local VRF Routes  On-demand
Identifier                    Source          Count     Entry Using Cache Color
[tree-id 524290 (0x80002)]    192.168.0.4     1     N   N     1           0
```

View MRIB MPLS Forwarding Information

This command displays labels used for transporting IP multicast traffic, on a specified router.

```
Router# show mrib mpls forwarding

LSP information (XTC) :
  LSM-ID: 0x00000, Role: Head, Head LSM-ID: 0x80002
    Incoming Label       : (18000)
    Transported Protocol : <unknown>
    Explicit Null        : None
    IP lookup            : disabled

    Outsegment Info #1 [H/Push, Recursive]:
      OutLabel: 18000, NH: 192.168.0.3, Sel IF: GigabitEthernet0/2/0/0

LSP information (XTC) :
  LSM-ID: 0x00000, Role: Tail, Peek
  RPF-ID: 0x00011, Assoc-TIDs: 0xe0000011/0x0, MDT: TRmdtvpn1
    Incoming Label       : 18001
    Transported Protocol : <unknown>
    Explicit Null        : None
    IP lookup            : enabled

    Outsegment Info #1 [T/Pop]:
      No info.
```

SR-PCE Show Commands

View Tree Information On PCE Server

This command displays SR multicast tree information on the SR-PCE server.

```
Router# show pce lsp p2mp

Tree: sr_p2mp_root_192.168.0.1_tree_id_524290
 Label:    18000     Operational: up  Admin: up
 Metric Type: TE
 Transition count: 3
 Uptime: 00:00:03 (since Fri Jan 24 14:57:51 PST 2020)
 Source: 192.168.0.1
```

```
 Destinations: 192.168.0.4
 Nodes:
  Node[0]: 192.168.0.2 (rtrM)
   Role: Transit
   Hops:
    Incoming: 18000 CC-ID: 4
    Outgoing: 18000 CC-ID: 4 (17.17.17.4) [rtrR]
  Node[1]: 192.168.0.1 (rtrL1)
   Role: Ingress
   Hops:
    Incoming: 18000 CC-ID: 5
    Outgoing: 18000 CC-ID: 5 (12.12.12.2) [rtrM]
  Node[2]: 192.168.0.4 (rtrR)
   Role: Egress
   Hops:
    Incoming: 18000 CC-ID: 6
```

For dynamic SR multicast trees created for MVPN, the **show** command has filters to view root multicast router and Tree-ID information. When the root router is specified, all multicast trees from that root are displayed. When root and Tree-ID are specified, only the specified tree information is displayed.

```
Router# show pce lsp p2mp root ipv4 10.1.1.1 524289

Tree: sr_p2mp_root_10.1.1.1_tree_id_524289, Root: 10.1.1.1 ID: 524289
 Label:    20000     Operational: up  Admin: up
 PCC: 10.1.1.1
 Local LFA FRR: Disabled
 Metric Type: TE
 Transition count: 11
 Uptime: 00:03:37 (since Mon May 11 12:53:33 PDT 2020)
 Destinations: 10.1.1.3, 10.1.1.4, 10.1.1.5
 Nodes:
  Node[0]: 10.1.1.1 (root1)
   Role: Ingress
   Hops:
    Incoming: 20000 CC-ID: 26
    Outgoing: 20000 CC-ID: 26 (192.168.114.4) [mid-4]
    Outgoing: 20000 CC-ID: 26 (192.168.112.2) [mid-2]
  Node[1]: 10.1.1.4 (mid-4)
   Role: Egress
   Hops:
    Incoming: 20000 CC-ID: 27
  Node[2]: 10.1.1.2 (mid-2)
   Role: Transit
   Hops:
    Incoming: 20000 CC-ID: 28
    Outgoing: 20000 CC-ID: 28 (192.168.123.3) [leaf-3]
    Outgoing: 20000 CC-ID: 28 (192.168.125.5) [leaf-5]
  Node[3]: 10.1.1.3 (leaf-3)
   Role: Egress
   Hops:
    Incoming: 20000 CC-ID: 29
  Node[4]: 10.1.1.5 (leaf-5)
   Role: Egress
   Hops:
    Incoming: 20000 CC-ID: 30
```

Multicast Tree Information on Routers

```
Router# show segment-routing traffic-eng p2mp policy

SR-TE P2MP policy database:
---------------------

Policy: sr_p2mp_root_192.168.0.1_tree_id_524290  LSM-ID: 0x2
```

```
 Role: Leaf
 Replication:
  Incoming label: 18001 CC-ID: 6

Policy: sr_p2mp_root_192.168.0.4_tree_id_524290  LSM-ID: 0x80002 (PCC-initiated)
 Color: 0
 Role: Root
 Replication:
  Incoming label: 18000 CC-ID: 2
  Interface: None [192.168.0.3!]  Outgoing label: 18000 CC-ID: 2
 Endpoints:
  192.168.0.1, 192.168.0.2
```

For SR multicast policies originated locally on the router (root router of a dynamic MVPN multicast policy) additional policy information is displayed.

For dynamic SR multicast trees created for MVPN, the **show** command has filters for displaying root multicast router and Tree-ID information. When the root router is specified, all multicast trees for that root are displayed. When root and Tree-ID are specified, only the specified tree information is displayed.

```
Router# show segment-routing traffic-eng p2mp policy root ipv4 1.1$

SR-TE P2MP policy database:
---------------------

Policy: sr_p2mp_root_10.1.1.1_tree_id_524289  LSM-ID: 0x691
 Root: 10.1.1.1, ID: 524289
 Role: Transit
 Replication:
  Incoming label: 20000 CC-ID: 28
  Interface: Bundle-Ether23 [192.168.123.3]  Outgoing label: 20000 CC-ID: 28
  Interface: Bundle-Ether25 [192.168.125.5]  Outgoing label: 20000 CC-ID: 28

Policy: sr_p2mp_root_10.1.1.1_tree_id_524290  LSM-ID: 0x692
 Root: 10.1.1.1, ID: 524290
 Role: Transit
 Replication:
  Incoming label: 19999 CC-ID: 28
  Interface: Bundle-Ether23 [192.168.123.3]  Outgoing label: 19999 CC-ID: 28
  Interface: Bundle-Ether25 [192.168.125.5]  Outgoing label: 19999 CC-ID: 28
```

Multicast: Cisco Nonstop Forwarding for Tree-SID

Table 55: Feature History Table

Feature Name	Release Information	Feature Description
Multicast: Cisco Nonstop Forwarding for Tree-SID	Release 7.10.1	Starting from this release, Multicast Nonstop Forwarding supports Tree-SID (Tree Segment Identifier). This ensures that traffic forwarding continues without interruptions whenever the active RSP fails over to the standby RSP. This feature prevents hardware or software failures on the control plane from disrupting the forwarding of existing packet flows through the router for Tree-SID. Thus, ensuring improved network availability, network stability, preventing routing flaps, and no loss of user sessions while the routing protocol information is being restored. The feature modifies the **show mrib nsf private** command.

Note This section captures only the Cisco Nonstop Forwarding feature in relation with Tree-SID. For more information on the Cisco Nonstop Forwarding feature, see Multicast Nonstop Forwarding.

Multicast now supports hitless Route Processor Fail Over (RPFO). During RPFO, the software deletes IP routes from the Static Tree-SID profile in the headend router. The Dynamic Tree-SID does not have this issue, because in this case, the BGP advertises the states that supports Nonstop Routing (NSR). To overcome this problem for static Tree-SID, there are checkpoints to check the feature in Protocol Independent Multicast (PIM). On switchover, the checkpoint reads to check if the feature is there or not and push Protocol Independent Multicast (PIM) to Cisco Nonstop Forwarding state.

Verification Steps

The **show mrib nsf private** command is enhanced to display the XTC info as well.

```
Router#show mrib nsf private
Mon Jul 31 13:27:05.056 UTC
IP MRIB Non-Stop Forwarding Status:
Multicast routing state: Normal
   NSF Lifetime:       00:03:00
   Respawn Count: 6
   Last NSF On triggered: Tue Jul 25 13:20:49 2023, 6d00h
   Last NSF Off triggered: Tue Jul 25 13:22:49 2023, 6d00h
   Last NSF ICD Notification sent: Tue Jul 25 13:22:49 2023, 6d00h
   Last Remote NSF On triggered: Tue Jul 25 13:10:18 2023, 6d00h
   Last Remote NSF Off triggered: Tue Jul 25 13:10:27 2023, 6d00h
   Last Label TE NSF On triggered: Tue Jul 25 13:10:18 2023, 6d00h
   Last Label TE NSF Off triggered: Tue Jul 25 13:10:27 2023, 6d00h
   Last Label mLDP NSF On triggered: Tue Jul 25 13:10:18 2023, 6d00h
   Last Label mLDP NSF Off triggered: Tue Jul 25 13:10:27 2023, 6d00h
   Last Label PIM NSF On triggered: Tue Jul 25 13:20:49 2023, 6d00h
   Last Label PIM NSF Off triggered: Tue Jul 25 13:22:49 2023, 6d00h
```

```
   Last Label PIM6 NSF On triggered: Tue Jul 25 13:31:22 2023, 5d23h
   Last Label PIM6 NSF Off triggered: Tue Jul 25 13:33:22 2023, 5d23h
   Last Label XTC NSF On triggered: Tue Jul 25 13:41:51 2023, 5d23h
   Last Label XTC NSF Off triggered: Tue Jul 25 13:41:52 2023, 5d23h

IP NSF :- Active: N, Assume N
MRIB connect timer: Inactive
NSF statistics:
   Enabled Cnt - 4, Disabled Cnt - 4
   Last Enabled: 6d00h, Last Disabled: 6d00h
Multicast COFO routing state: Normal
Current LMRIB clients: LDP RSVP_TE PIM PIM6 XTC
LMRIB NSF clients: LDP RSVP_TE PIM PIM6 XTC
Converged LMRIB clients: LDP RSVP_TE PIM PIM6 XTC
```

Multicast VPN IPv6: Dynamic Tree-SID Multicast VPN IPv6

Table 56: Feature History Table

Feature Name	Release Information	Feature Description
Multicast VPN: Dynamic Tree-SID Multicast VPN IPv6	Release 7.10.1	This feature allows Dynamic Tree Segment Identifier (Tree-SID) deployment where IPv6 Multicast payload is used for optimally transporting IP VPN multicast traffic over the provider network, using SR-PCE as a controller. This implementation supports IPv6 only for the Dynamic Tree-SID. Currently, the Static Tree-SID supports IPV4 payloads only, not the IPv6 payloads.

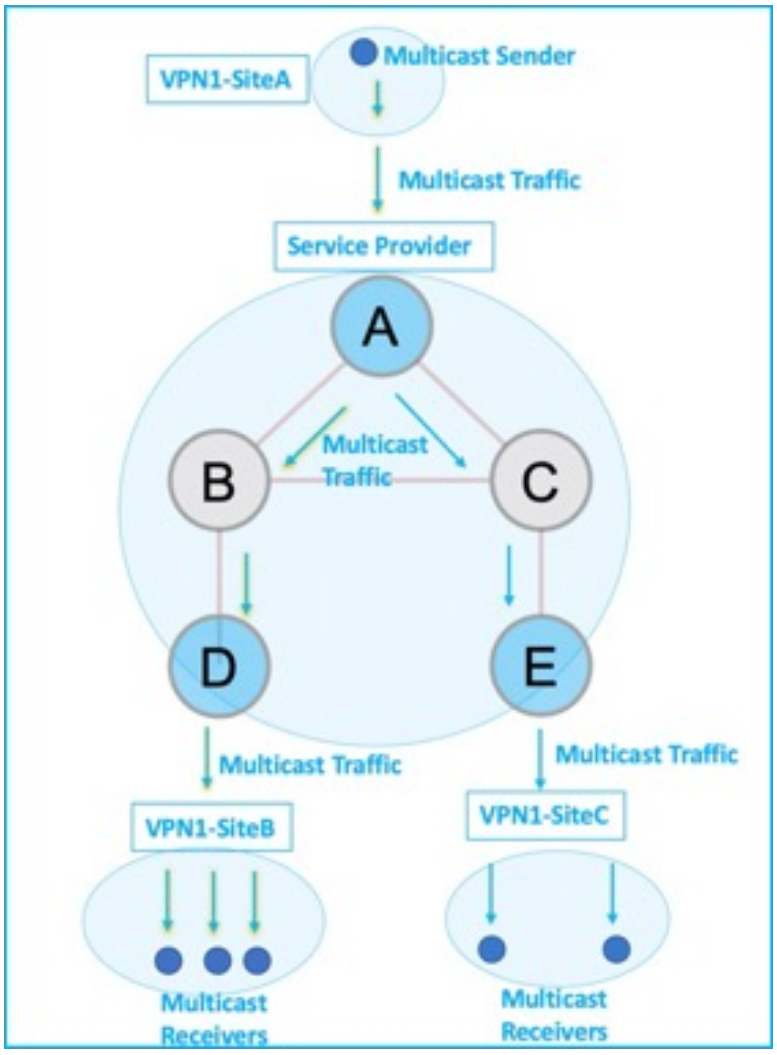
Overview of Multicast VPN: Tree-SID Multicast VPN

Typically, a customer's IP VPN is spread across VPN sites. IP VPN customer traffic is sent from one site to another over a VPN Service Provider (SP) network.

When IP Multicast traffic within a (BGP/MPLS) IP VPN is transported over a provider network (say, from **VPN1-Site-A** to **VPN1-Site-B**, as shown in the image), the provider network requires protocols and procedures to optimally transport multicast traffic from a multicast sender in Site-A to multicast receivers in Site-B.

This use case explains how to enable SR multicast for a provider network, and efficiently transport IP VPN multicast traffic (sent from **VPN1-Site-A** and) received at PE router A, through to PE routers D and E, toward receivers in sites **VPN1-Site-B** and **VPN1-Site-C**.

Figure 40: IP VPN Multicast Traffic Flow Over A Provider Network



To enable the *Multicast VPN: Tree-SID multicast VPN* feature, the following protocols and software applications are used:

- **OSPF/IS-IS** - The underlay network is created with OSPF/IS-IS routing protocol, and reachability is established across the network. See *Configure Segment Routing for IS-IS Protocol* or *Configure Segment Routing for OSPF Protocol* chapter for details, within this Guide.
- **BGP Multicast VPN (multicast VPN)** – The PE routers (A, D, and E) are IP VPN endpoints for IP Multicast traffic arriving at the provider network (at PE router A) and exiting the provider network (at PE routers D and E). So, BGP multicast VPN is enabled on the PE routers. NSO is used to configure BGP multicast VPN on the PE routers. See, *Configure Segment Routing for BGP* chapter for details, within this guide
- **BGP Auto-Discovery (AD)** - To enable distributed VPN endpoint discovery and C-multicast flow mapping and signaling, BGP AD function is configured on the PE routers. A BGP Auto-Discovery route contains multicast router (loopback IP address) and tree identity (segment ID) information. It carries the

information in the Provider Multicast Service Interface (PMSI) Tunnel Attribute (PTA). See, *Configure Segment Routing for BGP* chapter for details, within this guide

- **C-multicast states** are signaled using BGP. See, *Configure Segment Routing for BGP* chapter for details, within this guide
- **SR** - To transport IP Multicast traffic between the VPN endpoints (PE routers A, D, and E), Provider (or P-) tunnels are used. In a P-tunnel, the PE devices are the tunnel endpoints. P-tunnels can be generated using different technologies (RSVP-TE, point-to-multipoint LSPs, PIM trees, mLDP point-to-multipoint LSPs, and mLDP MP2MP LSPs). In this use case, Segment Routing (SR) is used for its benefits that were noted earlier.
- With SR and SR-PCE, a Tree-SID point-to-multipoint (P2MP) segment is used to create P-Tunnels for multicast VPN. You can specify SR policy optimization objectives (such as *metrics*) and constraints (such as *affinity*) in an SR policy and send it to the SR-PCE controller, so that it can dynamically create SR multicast trees for traffic flow.
- **SR-PCE** - This is a controller which, based on the provided SR policy information, computes optimal paths for a multicast tree, and deploys the tree forwarding state on the multicast routers. When a topology change occurs, SR-PCE automatically computes a new, optimal multicast tree, and deploys the new tree forwarding state on the multicast routers.
-

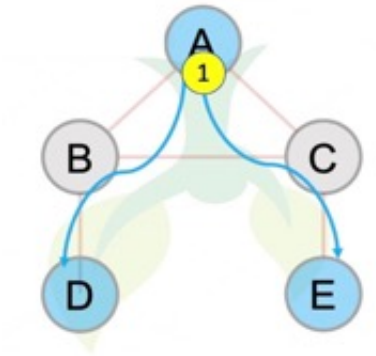
Tree-SID multicast VPN

The topology remains the same, with PE routers A, D, and E acting as VPN endpoints for carrying IP VPN multicast traffic.

1. For SR, A is designated as the SR headend router, and D and E are designated as the SR endpoints.

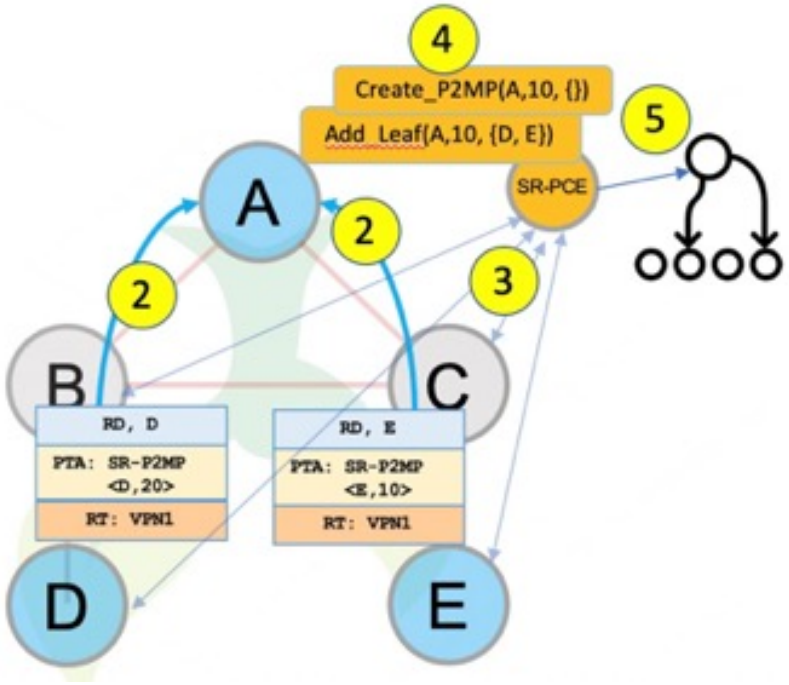
   For multicast traffic, A is the root of the SR multicast tree, and D and E are leaf routers of the tree. B and C are the other multicast routers. The objective is to send the IP Multicast traffic arriving at A to D and E, as needed.

   Figure 41: Multicast Tree

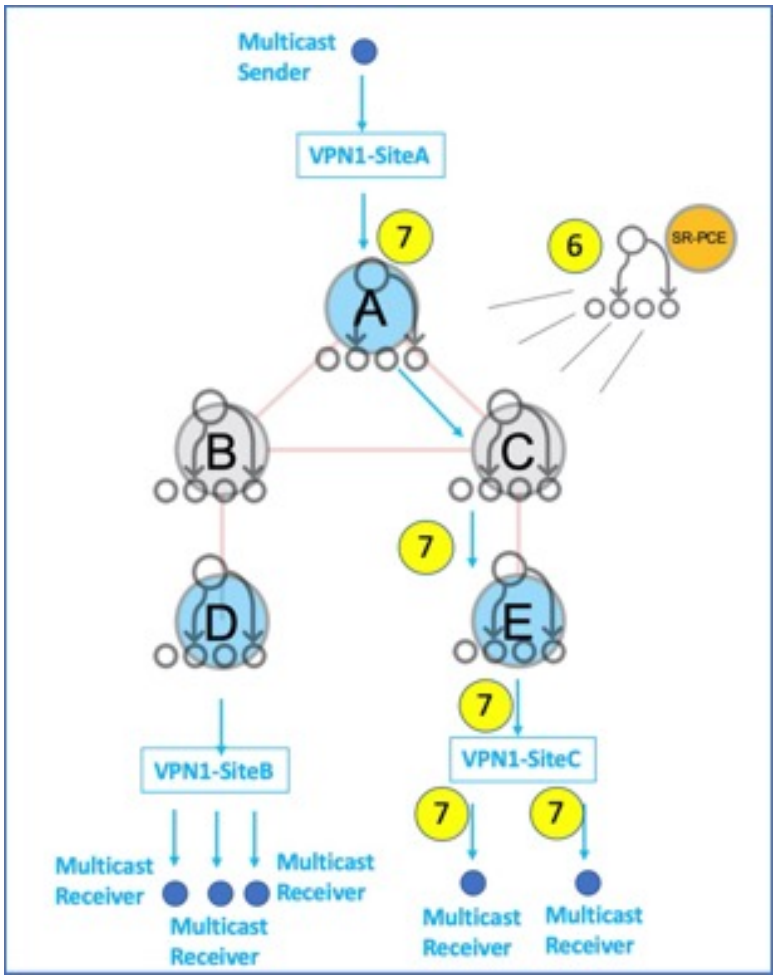


2. A discovers leaf routers' information through BGP multicast VPN.
3. Path Computation Element Protocol (PCEP) is used for the SR multicast policy communication between A and the SR-PCE server, and communication between PE routers and the SR-PCE server.

4. When the headend router SR policy is created on A, and PCEP configurations are enabled on the SR-PCE server and all multicast routers, SR-PCE receives the SR policy and leaf router identity information from A.
5. Based on the policy information it receives, including traffic engineering objectives and constraints, SR-PCE builds multicast distribution trees in the underlay for efficient VPN traffic delivery.



6. SR-PCE assigns an SID for the SR multicast tree policy, and deploys the multicast tree forwarding state on the multicast routers.
7. When IP Multicast traffic is sent from VPN1-SiteA to PE router A, it steers it into the SR policy, and sends it toward D and E, which forward it to multicast traffic receivers in the sites VPN1-SiteB and VPN1-SiteC.
8. When a leaf or multicast router is added or removed, PE router A updates the SR multicast policy and sends it to SR-PCE. SR-PCE computes new multicast routes, and deploys the multicast tree forwarding state information on the multicast routers.

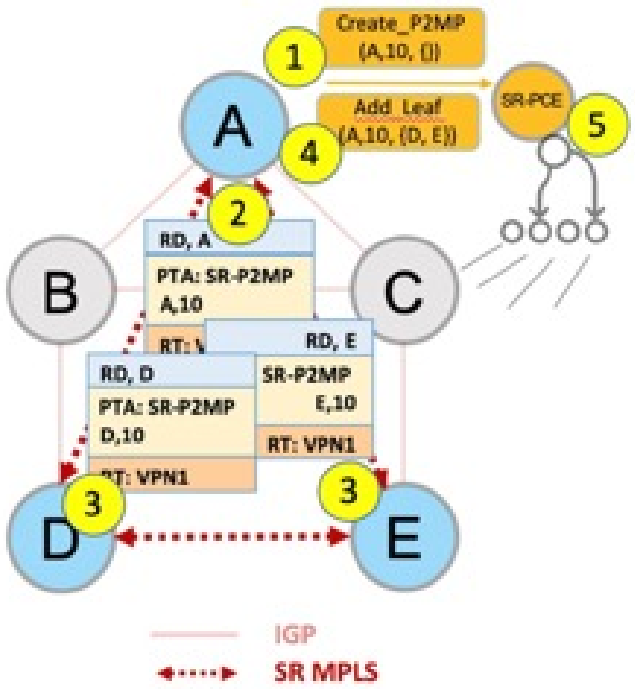


SR Multicast Tree Types

This is an overview of the types of SR multicast trees that you can configure, depending on your requirement. You can create the following tree types for IP VPN multicast flow in the provider network:

- Full Mesh Multicast Tree
- On-Demand SR Multicast Tree
- Optimal Multicast Tree
- **Full Mesh Multicast Tree**

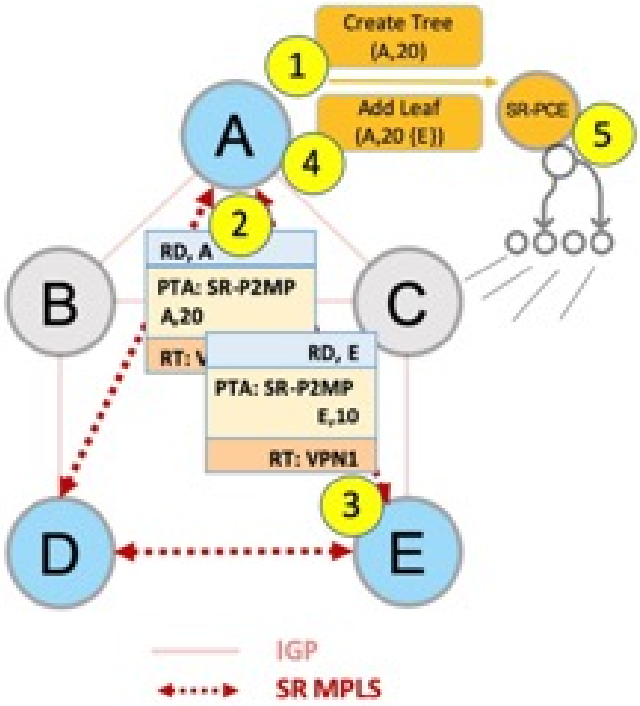
Figure 42: Full Mesh Multicast Tree



1. A assigns Tree-ID 10 and invokes a Create an SR multicast tree request by sending the multicast router and tree ID information (A, 10) toward SR-PCE.
2. A announces BGP AD Inclusive PMSI (I-PMSI) route with the PTA (A, 10). Inclusive PMSI - Traffic that is multicast by a PE router on an I-PMSI is received by all other PEs in the multicast VPN. I-PMSIs are generated by Inclusive P-tunnels.
3. A discovers VPN endpoints D and E from their BGP AD Type I-PMSI route messages.
4. A invokes an Add SR multicast leaf router request (for D and E) to SR-PCE.
5. SR-PCE computes and generates the multicast tree forwarding state information on all the routers that are part of the tree.

- **On-Demand SR Multicast Tree**

Figure 43: On-Demand SR Multicast Tree

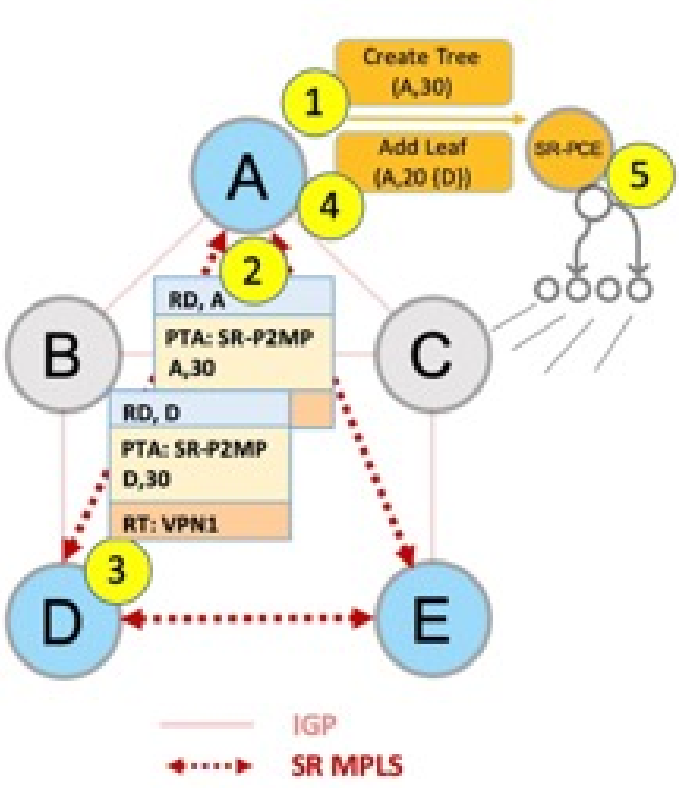


1. A assigns Tree-ID 20 and invokes a Create an SR multicast tree request by sending the multicast router and tree ID information (A, 20) toward SR-PCE.
2. A announces BGP AD Selective PMSI (or S-PMSI) route with PTA (A, 20). A sets the leaf-info-required to discover endpoint interest set.

   Selective PMSI - Traffic multicast by a PE on an S-PMSI is received by some PEs in the multicast VPN. S-PMSIs are generated by Selective P-tunnels.
3. E has a receiver behind it, and announces a BGP-AD leaf route toward A. A discovers service endpoint E for the on-demand tree.
4. A invokes an Add SR multicast leaf router request (for E) to SR-PCE.
5. SR-PCE computes and generates the multicast tree information for all the routers that are part of the tree.

- **Optimal Multicast Tree**

Figure 44: Optimal Multicast Tree



1. A decides to optimize a flow and assigns Tree-ID 30 and invokes a Create an SR multicast tree request by sending the multicast router and tree ID information (A, 30) toward SR-PCE.
2. A announces BGP AD I-PMSI route with PTA (A, 30). A sets the leaf-info-required to discover endpoint interest set.
3. D has a receiver behind it, and announces a BGP-AD leaf route toward A. A discovers service endpoint D for optimized flow.
4. A invokes an Add SR multicast leaf router request (for D) to SR-PCE.
5. SR-PCE computes and generates the multicast tree information for all the routers that are part of the tree.

Prerequisites for Tree-SID mVPN IPv6

Listed are the prerequisites for Tree-SID Multicast VPN IPv6:

- The underlay OSPF or IS-IS network is configured, and OSPF/IS-IS adjacency forms between routers, across the network.
- BGP is configured for the network, and BGP adjacency is formed between routers. BGP multicast VPN configuration information is provided in this feature document.

Restrictions to Tree-SID mVPN IPv6

Listed are the restrictions related to this feature:

- The following are not supported for MVPN SR P2MP:
  - SRv6 SR P2MP policies
  - Hitless RP failover
  - IPV6 Multicast payload
  - PCE redundancy
  - PIM Bidir is not supported
- The following are not supported for SR P2MP:
  - PCE server restart not supported for REST initiated SR P2MP policies
  - Co-existence of static MVPN SR P2MP profiles in a VRF of a PE.
  - Co-existence with other MVPN profiles (MLDP, P2MP RSVP-TE, Ingress Replication) that need BGP MVPN Auto-Discovery in a VRF of a PE.
  - PIM C-Multicast signaling (only BGP C-multicast is supported)

Configure Tree-SID mVPN IPv6

Configuration Examples

Following are examples to configure Tree-SID multicase VPN IPv6

- **Headend Router Configuration (Router A)** - The following configuration is specific to the headend router.
  - **Configure traffic engineering Constraints and Optimization Parameters**

```
Router# configure terminal
Router(config)# segment-routing traffic-engineering
```

An affinity bit-map is created so that it can be applied to a link or interface.

```
Router(config-sr-te)# affinity-map name 10 bit-position 24
Router(config-sr-te)# commit
```

An affinity (or relationship) is created between the SR policy path and the link color so that SR-TE computes a path that includes or excludes links, as specified. The headend router automatically follows the actions that are defined in the ODN template (for color 10) upon the arrival of VPN routes with a BGP color extended community that matches color 10.

```
Router(config)# segment-routing traffic-engineering
Router(config-sr-te)# on-demand color 10 dynamic
Router(config-sr-te-color-dyn)# affinity include-all name red
Router(config-sr-te-color-dyn)# affinity include-any name blue
Router(config-sr-te-color-dyn)# affinity exclude-any name green
Router(config-sr-te-color-dyn)# metric type te
Router(config-sr-te-color-dyn)# commit
```

The SR policy configuration on the headend router A will be sent to the SR-PCE server, after a connection is established between A and SR-PCE.

- **Multicast Router Configuration**
  - **Configure PCEP Client on Multicast Routers** - Associate each multicast router as a client of the SR-PCE server. The **pce address ipv6** command specifies the SR-PCE server's IP address.

```
Router# configure terminal
Router(config)# segment-routing traffic-engineering
Router(config-sr-te)# pcc pce address ipv6 10.3.3.3
Router(config-pcc-pce)# commit
```

- **SR PCE Server Configuration**
  - **Configure Label Range for Multicast Trees** - Configure the label range to be used for transporting Cisco IOS IP Multicast traffic in provider network.

```
Router(config)# pce segment-routing traffic-eng p2mp label-range min 30000 max 60000
Router(config)# commit
```

  - **Disable ECMP load splitting** - To disable ECMP load splitting of different trees on the SR-PCE server, configure the **multipath-disable** command.

```
Router(config)# pce segment-routing traffic-eng p2mp multipath-disable
Router(config)# commit
```

- **Multicast Routing Configuration On PE Routers** - The following multicast VPN configurations are required for VPN endpoints, the 3 PE routers.
  - **Configure Default MDT SR point-to-multipoint multicast VPN Profile** - In this configuration, an MDT profile of the type *default* is created, and the SR multicast policy with color 10 will be used to send Cisco IOS IP Multicast traffic, as per the constraints and optimizations of the policy, through the multicast tree.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv6
Router(config-mcast-cust1-ipv6)# mdt default segment-routing mpls color 10
Router(config-mcast-cust1-ipv6)# commit
```

  - **Configure Partitioned MDT SR point-to-multipoint multicast VPN Profile** - In this configuration, an MDT profile of the type *partitioned* is created, and the SR multicast policy with color 10 will be used to send Cisco IOS IP Multicast traffic, as per the constraints and optimizations of the policy, through the multicast tree.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv6
Router(config-mcast-cust1-ipv6)# mdt partitioned segment-routing mpls color 10
Router(config-mcast-cust1-ipv6)# commit
```

  The following Data multicast VPN configuration is required at the Ingress PE (router A) where the multicast flows need to be steered onto the *data* MDT for SR multicast traffic flow.

  Note - *Data* MDT can be configured for *Default* and *Partitioned* profiles.

  Configure Data MDT for SR point-to-multipoint multicast VPN - In this configuration, an MDT profile of the type *data* is created, and the SR multicast policy with color 10 will be used to send Cisco IOS IP Multicast traffic, as per the constraints and optimizations of the policy, through the multicast tree.

- As an alternative to the color keyword, you can specify a route policy in the **route-policy** command, and define the route policy separately (as mentioned in the next configuration).
- The **threshold** command specifies the threshold above which a multicast flow is switched onto the data MDT. The **immediate-switch** keyword enables an immediate switch of a multicast flow to the data MDT, without waiting for threshold limit to be crossed.
- The **customer-route-acl** keyword specifies an access control list to enable specific multicast flows to be put on to the data MDT.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv6
Router(config-mcast-cust1-ipv6)# mdt data segment-routing mpls 2 color 10
Router(config-mcast-cust1-ipv6)# commit
```

Route Policy Example

The route policy designates multicast flow-to-SR multicast policy mapping, with different colors.

- With this configuration, Cisco IOS IP Multicast flows for the 232.0.0.1 multicast group are steered into the SR multicast policy that is created with the on-demand color 10, while flows for 232.0.0.2 are steered into the policy created with color 20.
- Route policies can also be used to match other parameters, such as source address.

```
Router(config)# route-policy TSID-DATA
Router(config-rpl)# if destination in (232.0.0.1) then
Router(config-rpl-if)# set on-demand-color 10
Router(config-rpl-if)# pass
Router(config-rpl-if)# elseif destination in (232.0.0.2) then
Router(config-rpl-elseif)# set on-demand-color 20
Router(config-rpl-elseif)# pass
Router(config-rpl-elseif)# endif
Router(config-rpl)# end-policy
Router(config)# commit
```

Configure multicast VPN BGP Auto-Discovery for SR point-to-multipoint

The following configuration is required on all PE routers, and is mandatory for *default* MDT, *partitioned* MDT, and *data* MDT.

Configure the BGP Auto-Discovery function for transporting Cisco IOS IP Multicast traffic.

```
Router(config)# multicast-routing vrf cust1
Router(config-mcast-cust1)# address-family ipv6
Router(config-mcast-cust1-ipv6)# bgp auto-discovery segment-routing
Router(config-mcast-cust1-ipv6-bgp-ad)# commit
```

Verify Tree-SID mVPN IPv6 With TI-LFA

This section guides you through the verification options:

- **View multicast VPN Context Information** - You can view multicast VPN virtual routing and forwarding context information with these commands.
- **View Default MDT Configuration** - This command displays SR multicast tree information, including the MDT details (of *default* type, and so on), and customer virtual routing and forwarding information (route target, route distinguisher, and so on).

```
Router# show mvpn vrf vpn1 context

MVPN context information for VRF vpn1 (0x9541cf0)

RD: 1:10 (Valid, IID 0x1), VPN-ID: 0:0
Import Route-targets : 2
  RT:192.168.0.4:0, BGP-AD
  RT:192.168.0.4:17, BGP-AD
BGP Auto-Discovery Enabled (I-PMSI added)
SR P2MP Core-tree data:
  MDT Name: TRmdtvpn1, Handle: 0x4150, idb: 0x956fc30
  MTU: 1376, MaxAggr: 255, SW_Int: 30, AN_Int: 60
  RPF-ID: 3, C:0, O:1, D:0, CP:0
  Static Type : - / -
  Def MDT ID: 524289 (0x93993f0), added: 1, HLI: 0x80001, Cfg: 1/0
  Part MDT ID: 0 (0x0), added: 0, HLI: 0x00000, Cfg: 0/0
  Ctrl Trees : 0/0/0, Ctrl ID: 0 (0x0), Ctrl HLI: 0x00000
```

- **View Partitioned MDT Configuration** - This command displays SR multicast tree information, including the MDT details (of *partitioned* type, and so on), and customer virtual routing and forwarding information (route target, route distinguisher, and so on).

```
Router# show mvpn vrf vpn1 context

MVPN context information for VRF vpn1 (0x9541cf0)

RD: 1:10 (Valid, IID 0x1), VPN-ID: 0:0
Import Route-targets : 2
  RT:192.168.0.4:0, BGP-AD
  RT:192.168.0.4:17, BGP-AD
BGP Auto-Discovery Enabled (I-PMSI added) , MS-PMSI sent
SR P2MP Core-tree data:
  MDT Name: TRmdtvpn1, Handle: 0x4210, idb: 0x956fc30
  MTU: 1376, MaxAggr: 255, SW_Int: 30, AN_Int: 60
  RPF-ID: 1, C:0, O:1, D:0, CP:0
  Static Type : - / -
  Def MDT ID: 0 (0x0), added: 0, HLI: 0x00000, Cfg: 0/0
  Part MDT ID: 524292 (0x9399318), added: 1, HLI: 0x80004, Cfg: 1/0
  Ctrl Trees : 0/0/0, Ctrl ID: 0 (0x0), Ctrl HLI: 0x00000
```

- **View Partitioned MDT Ingress PE Configuration** - This command displays SR multicast tree information on the PE router that receives the multicast traffic on the provider network. The information includes PE router details, MDT details, Tree-SID details, and the specified customer virtual routing and forwarding information.

```
Router# show mvpn vrf vpn1 pe

MVPN Provider Edge Router information

VRF : vpn1

PE Address : 192.168.0.3 (0x9570240)
  RD: 0:0:0 (null), RIB_HLI 0, RPF-ID 13, Remote RPF-ID 0, State: 0, S-PMSI: 2
  PPMP_LABEL: 0, MS_PMSI_HLI: 0x00000, Bidir_PMSI_HLI: 0x00000, MLDP-added: [RD 0, ID
0, Bidir ID 0, Remote Bidir ID 0], Counts(SHR/SRC/DM/DEF-MD): 0, 0, 0, 0, Bidir: GRE
RP Count 0, MPLS RP Count 0RSVP-TE added: [Leg 0, Ctrl Leg 0, Part tail 0 Def Tail 0,
IR added: [Def Leg 0, Ctrl Leg 0, Part Leg 0, Part tail 0, Part IR Tail Label 0
  Tree-SID Added: [Def/Part Leaf 1, Def Egress 0, Part Egress 0, Ctrl Leaf 0]
  bgp_i_pmsi: 1,0/0 , bgp_ms_pmsi/Leaf-ad: 1/1, bgp_bidir_pmsi: 0, remote_bgp_bidir_pmsi:
 0, PMSIs: I 0x9570378, 0x0, MS 0x94e29d0, Bidir Local: 0x0, Remote: 0x0, BSR/Leaf-ad
0x0/0, Autorp-disc/Leaf-ad 0x0/0, Autorp-ann/Leaf-ad 0x0/0
 IIDs: I/6: 0x1/0x0, B/R: 0x0/0x0, MS: 0x1, B/A/A: 0x0/0x0/0x0
```

```
  Bidir RPF-ID: 14, Remote Bidir RPF-ID: 0
  I-PMSI:   Unknown/None (0x9570378)
  I-PMSI rem:  (0x0)
  MS-PMSI:   Tree-SID [524290, 192.168.0.3] (0x94e29d0)
  Bidir-PMSI:  (0x0)
  Remote Bidir-PMSI:  (0x0)
  BSR-PMSI:  (0x0)
  A-Disc-PMSI:  (0x0)
  A-Ann-PMSI:  (0x0)
  RIB Dependency List: 0x0
  Bidir RIB Dependency List: 0x0
    Sources: 0, RPs: 0, Bidir RPs: 0
```

- **View Partitioned MDT Egress PE Configuration** - This command displays SR multicast tree information on the multicast VPN egress PE router that sends multicast traffic from the provider network toward multicast receivers in the destination sites. The information includes PE router, Tree-SID, MDT, and the specified customer virtual routing and forwarding details.

```
Router# show mvpn vrf vpn1 pe

MVPN Provider Edge Router information

PE Address : 192.168.0.4 (0x9fa38f8)
  RD: 1:10 (valid), RIB_HLI 0, RPF-ID 15, Remote RPF-ID 0, State: 1, S-PMSI: 2
  PPMP_LABEL: 0, MS_PMSI_HLI: 0x00000, Bidir_PMSI_HLI: 0x00000, MLDP-added: [RD 0, ID
0, Bidir ID 0, Remote Bidir ID 0], Counts(SHR/SRC/DM/DEF-MD): 1, 1, 0, 0, Bidir: GRE
RP Count 0, MPLS RP Count 0RSVP-TE added: [Leg 0, Ctrl Leg 0, Part tail 0 Def Tail 0,
IR added: [Def Leg 0, Ctrl Leg 0, Part Leg 0, Part tail 0, Part IR Tail Label 0
  Tree-SID Added: [Def/Part Leaf 0, Def Egress 0, Part Egress 1, Ctrl Leaf 0]
  bgp_i_pmsi: 1,0/0 , bgp_ms_pmsi/Leaf-ad: 1/0, bgp_bidir_pmsi: 0, remote_bgp_bidir_pmsi:
 0, PMSIs: I 0x9f77388, 0x0, MS 0x9fa2f98, Bidir Local: 0x0, Remote: 0x0, BSR/Leaf-ad
0x0/0, Autorp-disc/Leaf-ad 0x0/0, Autorp-ann/Leaf-ad 0x0/0
 IIDs: I/6: 0x1/0x0, B/R: 0x0/0x0, MS: 0x1, B/A/A: 0x0/0x0/0x0

  Bidir RPF-ID: 16, Remote Bidir RPF-ID: 0
  I-PMSI:   Unknown/None (0x9f77388)
  I-PMSI rem:  (0x0)
  MS-PMSI:   Tree-SID [524292, 192.168.0.4] (0x9fa2f98)
  Bidir-PMSI:  (0x0)
  Remote Bidir-PMSI:  (0x0)
  BSR-PMSI:  (0x0)
  A-Disc-PMSI:  (0x0)
  A-Ann-PMSI:  (0x0)
  RIB Dependency List: 0x9f81370
  Bidir RIB Dependency List: 0x0
    Sources: 1, RPs: 1, Bidir RPs: 0
```

- **View Data MDT Information** - The commands in this section display SR multicast tree information for *data* MDTs. The information includes cache, router-local, and remote MDT information.
- **View Data MDT Cache Information**

```
Router# show pim vrf vpn1 mdt cache
Core Source      Cust (Source, Group)              Core Data        Expires
192.168.0.3      (10.3.4.1, 203.0.0.1)            [tree-id 524292]  never

192.168.0.4      (10.3.4.6, 203.0.0.1)            [tree-id 524290]  never

   Leaf AD:  192.168.0.3
```

- **View Local MDTs Information**

```
Router# show pim vrf vpn1 mdt sr-p2mp local
```

```
Tree                            MDT             Cache DIP Local VRF Routes  On-demand
Identifier                      Source          Count     Entry Using Cache Color
[tree-id 524290 (0x80002)]      192.168.0.4     1     N   Y     1           10
   Tree-SID Leaf:  192.168.0.3
```

- **View Remote MDTs Information**

```
Router # show pim vrf vpn1 mdt sr-p2mp remote

Tree                            MDT             Cache DIP Local VRF Routes  On-demand
Identifier                      Source          Count     Entry Using Cache Color
[tree-id 524290 (0x80002)]      192.168.0.4     1     N   N     1           0
```

- **View MRIB MPLS Forwarding Information** - This command displays labels that are used for transporting Cisco IOS IP Multicast traffic, on a specified router.

```
Router# show mrib mpls forwarding

LSP information (XTC) :
  LSM-ID: 0x00000, Role: Head, Head LSM-ID: 0x80002
    Incoming Label       : (18000)
    Transported Protocol : <unknown>
    Explicit Null        : None
    IP lookup            : disabled

    Outsegment Info #1 [H/Push, Recursive]:
      OutLabel: 18000, NH: 192.168.0.3, Sel IF: GigabitEthernet0/2/0/0

LSP information (XTC) :
  LSM-ID: 0x00000, Role: Tail, Peek
  RPF-ID: 0x00011, Assoc-TIDs: 0xe0000011/0x0, MDT: TRmdtvpn1
    Incoming Label       : 18001
    Transported Protocol : <unknown>
    Explicit Null        : None
    IP lookup            : enabled

    Outsegment Info #1 [T/Pop]:
      No info.
```

- **SR-PCE Show Commands**
  - **View Tree Information On PCE Server** - This command displays SR multicast tree information on the SR-PCE server.

```
Router# show pce lsp p2mp

Tree: sr_p2mp_root_192.168.0.1_tree_id_524290
 Label:    18000     Operational: up  Admin: up
 Metric Type: TE
 Transition count: 3
 Uptime: 00:00:03 (since Fri Jan 24 14:57:51 PST 2020)
 Source: 192.168.0.1
 Destinations: 192.168.0.4
 Nodes:
  Node[0]: 192.168.0.2 (rtrM)
   Role: Transit
   Hops:
    Incoming: 18000 CC-ID: 4
    Outgoing: 18000 CC-ID: 4 (10.17.17.4) [rtrR]
  Node[1]: 192.168.0.1 (rtrL1)
   Role: Ingress
   Hops:
    Incoming: 18000 CC-ID: 5
    Outgoing: 18000 CC-ID: 5 (10.12.12.2) [rtrM]
  Node[2]: 192.168.0.4 (rtrR)
```

```
   Role: Egress
   Hops:
    Incoming: 18000 CC-ID: 6
```

For dynamic SR multicast trees created for multicast VPN, the **show** command has filters to view root multicast router and Tree-ID information. When the root router is specified, all multicast trees from that root are displayed. When root and Tree-ID are specified, only the specified tree information is displayed.

```
Router# show pce lsp p2mp root ipv6 10.1.1.1 524289

Tree: sr_p2mp_root_10.1.1.1_tree_id_524289, Root: 10.1.1.1 ID: 524289
 Label:    20000     Operational: up  Admin: up
 PCC: 10.1.1.1
 Local LFA FRR: Disabled
 Metric Type: TE
 Transition count: 11
 Uptime: 00:03:37 (since Mon May 11 12:53:33 PDT 2020)
 Destinations: 10.1.1.3, 10.1.1.4, 10.1.1.5
 Nodes:
  Node[0]: 10.1.1.1 (root1)
   Role: Ingress
   Hops:
    Incoming: 20000 CC-ID: 26
    Outgoing: 20000 CC-ID: 26 (192.168.114.4) [mid-4]
    Outgoing: 20000 CC-ID: 26 (192.168.112.2) [mid-2]
  Node[1]: 10.1.1.4 (mid-4)
   Role: Egress
   Hops:
    Incoming: 20000 CC-ID: 27
  Node[2]: 10.1.1.2 (mid-2)
   Role: Transit
   Hops:
    Incoming: 20000 CC-ID: 28
    Outgoing: 20000 CC-ID: 28 (192.168.123.3) [leaf-3]
    Outgoing: 20000 CC-ID: 28 (192.168.125.5) [leaf-5]
  Node[3]: 10.1.1.3 (leaf-3)
   Role: Egress
   Hops:
    Incoming: 20000 CC-ID: 29
  Node[4]: 10.1.1.5 (leaf-5)
   Role: Egress
   Hops:
    Incoming: 20000 CC-ID: 30
```

The following output shows that LFA FRR is enabled on the hop from rtrR to rtrM. Unlike typical multicast replication where the address displayed is the remote address on the link to a downstream router, the IP address 192.168.0.3 (displayed with an exclamation mark) is the router-ID of the downstream router rtrM. The output also displays the LFA FRR state for the multicast tree.

```
Router# show pce lsp p2mp

Tree: sr_p2mp_root_192.168.0.4_tree_id_524290
 Label:    18000     Operational: up  Admin: up
 LFA FRR: Enabled
 Metric Type: TE
 Transition count: 1
 Uptime: 3d19h (since Thu Feb 13 13:43:40 PST 2020)
 Source: 192.168.0.4
 Destinations: 192.168.0.1, 192.168.0.2
 Nodes:
  Node[0]: 192.168.0.3 (rtrM)
   Role: Transit
   Hops:
```

```
    Incoming: 18000 CC-ID: 1
    Outgoing: 18000 CC-ID: 1 (10.12.12.1) [rtrL1]
    Outgoing: 18000 CC-ID: 1 (10.15.15.2) [rtrL2]
  Node[1]: 192.168.0.4 (rtrR)
   Role: Ingress
   Hops:
    Incoming: 18000 CC-ID: 2
    Outgoing: 18000 CC-ID: 2 (192.168.0.3!) [rtrM]
  Node[2]: 192.168.0.1 (rtrL1)
   Role: Egress
   Hops:
    Incoming: 18000 CC-ID: 3
  Node[3]: 192.168.0.2 (rtrL2)
   Role: Egress
   Hops:
    Incoming: 18000 CC-ID: 4
```

- **Multicast Tree Information on Routers**

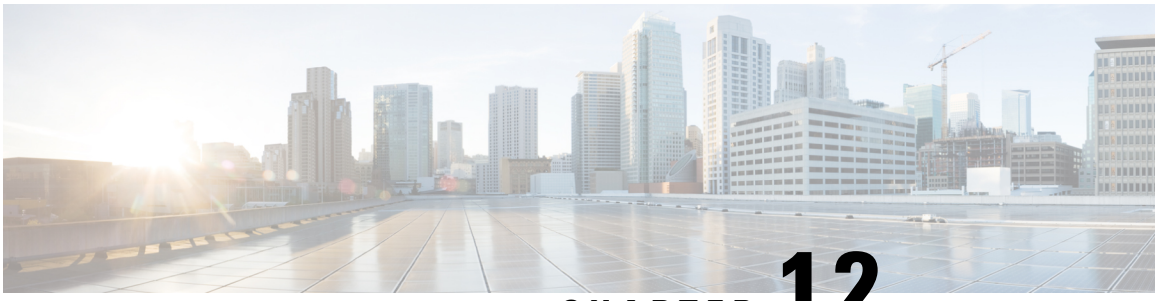
```
Router# show segment-routing traffic-eng p2mp policy

SR-TE P2MP policy database:
---------------------
! - Replications with Fast Re-route

Policy: sr_p2mp_root_192.168.0.1_tree_id_524290  LSM-ID: 0x2
 Role: Leaf
 Replication:
  Incoming label: 18001 CC-ID: 6

Policy: sr_p2mp_root_192.168.0.4_tree_id_524290  LSM-ID: 0x80002 (PCC-initiated)
 Color: 0
 LFA FRR: Disabled
 Role: Root
 Replication:
  Incoming label: 18000 CC-ID: 2
  Interface: None [192.168.0.3!]  Outgoing label: 18000 CC-ID: 2
 Endpoints:
  192.168.0.1, 192.168.0.2
```

For SR multicast policies originated locally on the router (root router of a dynamic multicast VPN multicast policy) additional policy information is displayed. The information includes color, end points, and whether LFA FRR is requested by the local application. When the SR-PCE server enables LFA FRR on a specific hop, the outgoing information shows the address of the next router with an exclamation mark and None is displayed for the outgoing interface.

For dynamic SR multicast trees created for multicast VPN, the **show** command has filters for displaying root multicast router and Tree-ID information. When the root router is specified, all multicast trees for that root are displayed. When root and Tree-ID are specified, only the specified tree information is displayed.

```
Router# show segment-routing traffic-eng p2mp policy root ipv6 1.1$

SR-TE P2MP policy database:
---------------------
! - Replications with Fast Re-route, * - Stale dynamic policies/endpoints

Policy: sr_p2mp_root_10.1.1.1_tree_id_524289  LSM-ID: 0x691
 Root: 10.1.1.1, ID: 524289
 Role: Transit
 Replication:
  Incoming label: 20000 CC-ID: 28
  Interface: Bundle-Ether23 [192.168.123.3]  Outgoing label: 20000 CC-ID: 28
```

```
  Interface: Bundle-Ether25 [192.168.125.5]  Outgoing label: 20000 CC-ID: 28

Policy: sr_p2mp_root_10.1.1.1_tree_id_524290  LSM-ID: 0x692
 Root: 10.1.1.1, ID: 524290
 Role: Transit
 Replication:
  Incoming label: 19999 CC-ID: 28
  Interface: Bundle-Ether23 [192.168.123.3]  Outgoing label: 19999 CC-ID: 28
  Interface: Bundle-Ether25 [192.168.125.5]  Outgoing label: 19999 CC-ID: 28
```



CHAPTER 12

Enabling Segment Routing Flexible Algorithm

Table 57: Feature History Table

Feature Name	Release Information	Feature Description
Segment Routing Flexible Algorithm	Release 7.3.1	The Segment Routing architecture associates prefix-SIDs to an algorithm that defines how the path is computed. This feature allows for user-defined algorithms where the IGP computes paths based on a combination of metric type and constraint. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, this feature provides a traffic-engineered path computed automatically by the IGP to any destination reachable by the IGP. This release supports the following functionality: • TI-LFA (IS-IS/OSPF) • Microloop Avoidance (IS-IS) • Inter-AS Support (IS-IS) • SID Redistribution (IS-IS) • Metric minimization—avoidance, multi-plane, delay (IS-IS/OSPF) • Affinity include (IS-IS/OSPF) • Affinity exclude (IS-IS/OSPF)

Segment Routing Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

The SR architecture associates prefix-SIDs to an algorithm which defines how the path is computed. Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraint.

This document describes the IS-IS and OSPF extensions to support Segment Routing Flexible Algorithm on an MPLS data-plane.

- Prerequisites for Flexible Algorithm, on page 490
- Building Blocks of Segment Routing Flexible Algorithm, on page 490
- Configuring Flexible Algorithm, on page 494
- Example: Configuring IS-IS Flexible Algorithm, on page 506
- Example: Configuring OSPF Flexible Algorithm, on page 506
- User-Defined Generic Metric Support for IS-IS Flex Algo, on page 508

Prerequisites for Flexible Algorithm

Segment routing must be enabled on the router before the Flexible Algorithm functionality is activated.

Building Blocks of Segment Routing Flexible Algorithm

This section describes the building blocks that are required to support the SR Flexible Algorithm functionality in IS-IS and OSPF.

Flexible Algorithm Definition

Many possible constrains may be used to compute a path over a network. Some networks are deployed with multiple planes. A simple form of constrain may be to use a particular plane. A more sophisticated form of constrain can include some extended metric, like delay, as described in [RFC7810]. Even more advanced case could be to restrict the path and avoid links with certain affinities. Combinations of these are also possible. To provide a maximum flexibility, the mapping between the algorithm value and its meaning can be defined by the user. When all the routers in the domain have the common understanding what the particular algorithm value represents, the computation for such algorithm is consistent and the traffic is not subject to looping. Here, since the meaning of the algorithm is not defined by any standard, but is defined by the user, it is called as Flexible Algorithm.

Flexible Algorithm Support Advertisement

An algorithm defines how the best path is computed by IGP. Routers advertise the support for the algorithm as a node capability. Prefix-SIDs are also advertised with an algorithm value and are tightly coupled with the algorithm itself.

An algorithm is a one octet value. Values from 128 to 255 are reserved for user defined values and are used for Flexible Algorithm representation.

Flexible Algorithm Definition Advertisement

To guarantee the loop free forwarding for paths computed for a particular Flexible Algorithm, all routers in the network must share the same definition of the Flexible Algorithm. This is achieved by dedicated router(s) advertising the definition of each Flexible Algorithm. Such advertisement is associated with the priority to make sure that all routers will agree on a single and consistent definition for each Flexible Algorithm.

Definition of Flexible Algorithm includes:

- Metric type
- Affinity constraints

To enable the router to advertise the definition for the particular Flexible Algorithm, **advertise-definition** command is used. At least one router in the area, preferably two for redundancy, must advertise the Flexible Algorithm definition. Without the valid definition being advertised, the Flexible Algorithm will not be functional.

Flexible Algorithm Prefix-SID Advertisement

To be able to forward traffic on a Flexible Algorithm specific path, all routers participating in the Flexible Algorithm will install a MPLS labeled path for the Flexible Algorithm specific SID that is advertised for the prefix. Only prefixes for which the Flexible Algorithm specific Prefix-SID is advertised is subject to Flexible Algorithm specific forwarding.

Calculation of Flexible Algorithm Path

A router may compute path for multiple Flexible Algorithms. A router must be configured to support particular Flexible Algorithm before it can compute any path for such Flexible Algorithm. A router must have a valid definition of the Flexible Algorithm before such Flexible Algorithm is used.

When computing the shortest path tree for particular Flexible Algorithm:

- All nodes that do not advertise support for such Flexible Algorithm will be pruned from the topology.
- If the Flexible Algorithm definition includes affinities that are excluded, then all links for which any of such affinities are advertised will be pruned from the topology.
- Router uses the metric that is part of the Flexible Algorithm definition. If the metric is not advertised for the particular link, such link will be pruned from the topology.

IS-IS supports Loop Free Alternate (LFA) paths, TI-LFA backup paths, and Microloop Avoidance paths for particular Flexible Algorithm. OSPF supports Loop Free Alternate (LFA) and TI-LFA backup paths for particular Flexible Algorithm. These paths are computed using the same constraints as the calculation of the primary paths for such Flexible Algorithm. These paths use Prefix-SIDs advertised specifically for such Flexible Algorithm in order to enforce a backup or microloop avoidance path.

Installation of Forwarding Entries for Flexible Algorithm Paths

Flexible Algorithm path to any prefix must be installed in the forwarding using the Prefix-SID that was advertised for such Flexible Algorithm. If the Prefix-SID for Flexible Algorithm is not known, such Flexible Algorithm path is not installed in forwarding for such prefix..

Only MPLS to MPLS entries are installed for a Flexible Algorithm path. No IP to IP or IP to MPLS entries are installed. These follow the native IGP paths computed based on the default algorithm and regular IGP metrics.

Flexible Algorithm Prefix-SID Redistribution

Table 58: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Algorithm Prefix-SID Redistribution for External Route Propagation	Release 7.5.2	You can now propagate flexible algorithm prefix-SIDs and their algorithm-specific metric between different IGP domains, such as OSPF to IS-IS RIP to OSPF. With this functionality enabling interdomain traffic engineering, you can export flexible algorithm labels from the OSPF domain to other domains and import the labels from other domains into OSPF. The **show ospf route flex-algo** command has been modified to include additional attributes to indicate the external routes.

Previously, prefix redistribution from IS-IS to another IS-IS instance or protocol was limited to SR algorithm 0 (regular SPF) prefix SIDs; SR algorithm 1 (Strict SPF) and SR algorithms 128-255 (Flexible Algorithm) prefix SIDs were not redistributed along with the prefix. The Segment Routing IS-IS Flexible Algorithm Prefix SID Redistribution feature allows redistribution of strict and flexible algorithms prefix SIDs from IS-IS to another IS-IS instance or protocols. This feature is enabled automatically when you configure redistribution of IS-IS Routes with strict or Flexible Algorithm SIDs.

Prefix redistribution from OSPF to another AS was limited to SR algorithm 0 (regular SPF) prefix SIDs; SR algorithm 1 (Strict SPF) and SR algorithms 128-255 (Flexible Algorithm) prefix SIDs were not redistributed along with the prefix. Starting from Cisco IOS XR Release 7.5.2, the Flexible Algorithm Prefix-SID Redistribution for External Route Propagation feature allows redistribution of strict and flexible algorithm prefixes SIDs from OSPF to another AS and also from another AS into OSPF.

Verification

This following show output displays the route-type as 'Extern' for the external routes.

```
Router#show ospf routes flex-algo 240 route-type external detail
Route Table of ospf-1 with router ID 192.168.0.2 (VRF default)

  Algorithm 240

  Route entry for 192.168.4.3/32, Metric 220, SID 536, Label 16536
   Priority : Medium

     Route type : Extern Type 1
Last updated : Apr 25 14:30:12.718
```

```
    Flags: Inuse

    Prefix Contrib Algo 240 SID 536
     From 192.168.0.4 Route-type 5
     Total Metric : 220 Base metric 20 FAPM 20
     Contrib Flags : Inuse, Reachable
     SID Flags : PHP off, Index, Global, Valid

     Path: 10.1.1.3, from 192.168.0.4, via GigabitEthernet0/2/0/2
      Out Label  : 16536
      Weight     : 0
      Area       : 0

     Path: 10.1.2.3, from 192.168.0.4, via GigabitEthernet0/2/0/3
      Out Label  : 16536
      Weight     : 0
      Area       : 0

     Path: 10.2.1.5, from 192.168.0.4, via GigabitEthernet0/2/0/4
      Out Label  : 16536
      Weight     : 0
      Area       : 0

  Route entry for 192.168.4.5/32, Metric 120, SID 556, Label 16556
   Priority : Medium

    Route type : Extern Type 1
    Last updated : Apr 25 14:30:12.724
    Flags: Inuse

    Prefix Contrib Algo 240 SID 556
     From 192.168.0.3 Route-type 5
     Total Metric : 120 Base metric 1 FAPM 20
     Contrib Flags : Inuse, Reachable
     SID Flags : PHP off, Index, Global, Valid

     Path: 10.1.1.3, from 192.168.0.3, via GigabitEthernet0/2/0/2
      Out Label  : 16556
      Weight     : 0
      Area       : 0

     Path: 10.1.2.3, from 192.168.0.3, via GigabitEthernet0/2/0/3
      Out Label  : 16556
      Weight     : 0
      Area       : 0
```

The following show output displays label information for flexible algorithm and its corresponding metric as added in RIB:

```
RP/0/RP0/CPU0:ios# show route 192.168.0.2/32 detail
Wed Apr  6 16:24:46.021 IST

Routing entry for 192.168.0.2/32
  Known via "ospf 1", distance 110, metric 2, labeled SR, type intra area
  Installed Apr  6 15:51:57.973 for 00:32:48
  Routing Descriptor Blocks
    10.10.10.2, from 192.168.0.2, via GigabitEthernet0/2/0/0, Protected
      Route metric is 2
      Label: 0x3 (3)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:0
      NHID:0x1(Ref:1)
      Backup path id:65
```

```
    OSPF area: 1
  10.11.11.2, from 192.168.0.2, via GigabitEthernet0/2/0/1, Backup (Local-LFA)
    Route metric is 6
    Label: 0x3 (3)
    Tunnel ID: None
    Binding Label: None
    Extended communities count: 0
    Path id:65              Path ref count:1
    NHID:0x2(Ref:1)
    OSPF area:
Route version is 0x12 (18)
Local Label: 0x3ee6 (16102)
Local Label Algo Set (ID, Label, Metric): (1, 16202, 0),(128, 17282, 2)
IP Precedence: Not Set
QoS Group ID: Not Set
Flow-tag: Not Set
Fwd-class: Not Set
Route Priority: RIB_PRIORITY_NON_RECURSIVE_MEDIUM (7) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 1, Download Version 38
No advertising protos.
```

Configuring Flexible Algorithm

Note For information about the commands usage, see the Segment Routing Command Reference for Cisco 8000 Series Routers.

The following ISIS and OSPF configuration sub-mode is used to configure Flexible Algorithm:

flex-algo *algorithm number*

algorithm number —value from 128 to 255

Commands under Flexible Algorithm Configuration Mode

The following commands are used to configure Flexible Algorithm definition under the flex-algo sub-mode:

metric-type delay

Note By default the regular IGP metric is used. If delay metric is enabled, the advertised delay on the link is used as a metric for Flexible Algorithm computation.

affinity {**include-any** | **include-all** | **exclude-any**} *name1, name2, …*

name—name of the affinity map

priority *priority value*

priority value—priority used during the Flexible Algorithm definition election.

The following command is used to enable advertisement of the Flexible Algorithm definition in IS-IS:

advertise-definition

Commands for Affinity Configuration

The following command is used for defining the affinity-map. Affinity-map associates the name with the particular bit positions in the Extended Admin Group bitmask.

```
affinity-map name bit-position bit number
```

- *name*—name of the affinity-map.
- *bit number*—bit position in the Extended Admin Group bitmask.

The following command is used to associate the affinity with an interface:

```
affinity flex-algo name 1, name 2, …
```

name—name of the affinity-map

Command for Prefix-SID Configuration

The following command is used to advertise prefix-SID for default and strict-SPF algorithm:

```
prefix-sid [strict-spf | algorithm algorithm-number]  [index | absolute] sid value
```

- *algorithm-number*—Flexible Algorithm number
- *sid value*—SID value

IS-IS Enhancements: max-metric and data plane updates

Table 59: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS Enhancements: max-metric and data plane updates	Release 7.8.1	The new **anomaly** optional keyword is introduced to **affinity flex-algo** command. This keyword helps to advertise the flex-algo affinity when the performance measurement signals a link anomaly, such as an excessive delay on a link. You could use the anomaly option to exclude the link from flex-algo path computations. affinity flex-algo

With the IOS XR Release 7.8.1, the new optional keyword **anomaly** is introduced to the **interface** submode of **affinity flex-algo**. This keyword option helps to advertise flex-algo affinity on PM anomaly. The following command is used to associate the affinity with an interface:

```
router isis instance interface type interface-path-id affinity flex-algo anomaly name 1,
name 2, …
```

```
router ospf process area area interface type interface-path-id affinity flex-algo anomaly
name 1, name 2, …
```

name—name of the affinity-map

You can configure both normal and anomaly values. For the following example, the **blue** affinity is advertised. However, if a metric is received with the anomaly flag set, it will change to **red**:

```
Router# configure
Router(config)# router isis 1
Router(config-isis)#flex-algo 128
Router(config-isis-flex-algo)# interface GigabitEthernet0/0/0/2
Router(config-isis-flex-algo)# affinity flex-algo blue
Router(config-isis-flex-algo)# affinity flex-algo anomaly red
```

Configuring Flexible Algorithm with Exclude SRLG Constraint

Table 60: Feature History Table

Feature Name	Release Information	Feature Description
Flexible Algorithm to Exclude SRLGs for OSPF	Release 7.5.2	You can now configure the flexible algorithm to exclude any link belonging to the Shared Risk Link Groups (SRLGs) from the path computation for OSPF. The ability to exclude the at-risk links ensures that the rest of the links in the network remain unaffected.

This feature allows the Flexible Algorithm definition to specify Shared Risk Link Groups (SRLGs) that the operator wants to exclude during the Flex-Algorithm path computation. A set of links that share a resource whose failure can affect all links in the set constitute a SRLG. An SRLG provides an indication of which links in the network might be at risk from the same failure.

This allows the setup of disjoint paths between two or more Flex Algos by leveraging deployed SRLG configurations. For example, multiple Flex Algos could be defined by excluding all SRLGs except one. Each FA will prune the links belonging to the excluded SRLGs from its topology on which it computes its paths.

This provides a new alternative to creating disjoint paths with FA, in addition to leveraging FA with link admin group (affinity) constraints.

The Flexible Algorithm definition (FAD) can advertise SRLGs that you want to exclude during the Flexible Algorithm path computation. The IS-IS Flexible Algorithm Exclude SRLG Sub-TLV (FAESRLG) is used to advertise the exclude rule that is used during the Flexible Algorithm path calculation, as specified in IETF draft https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/

The Flexible Algorithm path computation checks if an “exclude SRLG” rule is part of the FAD. If an “exclude SRLG” rule exists, it then checks if the link is part of an SRLG that is also part of the “exclude SRLG” rule. If the link is part of an excluded SRLG, the link is pruned from the path computation.

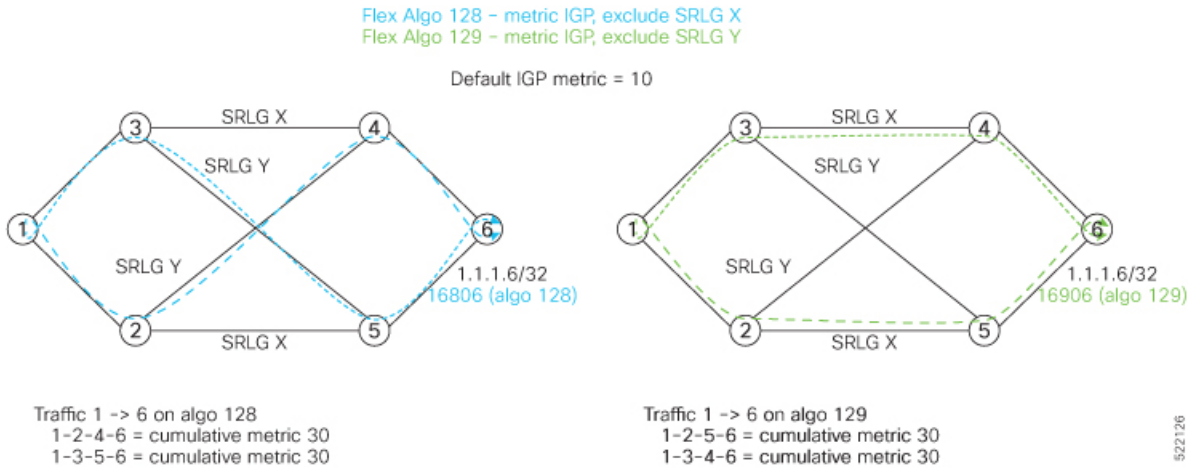
The figure below shows a topology configured with the following flex algos:

- Flex algo 128: metric IGP and exclude SRLG X constraint
- Flex algo 129: metric IGP and exclude SRLG Y constraint

The horizontal links between nodes 3 and 4 and between 2 and 5 are part of SRLG group X. The diagonal links between nodes 3 and 5 and between 2 and 4 are part of SRLG group Y. As a result, traffic from node 1 to node 6's FA 128 prefix SID (16806) avoids interfaces part of SRLG X. While traffic from node 1 to node 6's FA 129 prefix SID (16906) avoids interfaces part of SRLG Y.

Note See Constraints, on page 368 section in the *Configure SR-TE Policies* chapter for information about configuring SR policies with Flex-Algo constraints.

Figure 45: Flex Algo with Exclude SRLG Constraint



Configuration

Use the **router isis** *instance* **address-family ipv4 unicast advertise application flex-algo link-attributes srlg** command to enable the Flexible Algorithm ASLA-specific advertisement of SRLGs.

Use the **router isis** *instance* **flex-algo** *algo* **srlg exclude-any** *srlg-name . . . srlg-name* command to configure the SRLG constraint which is advertised in the Flexible Algorithm definition (FAD) if the FAD advertisement is enabled under the flex-algo sub-mode. You can specify up to 32 SRLG names.

The SRLG configuration (value and port mapping) is performed under the global SRLG sub-mode. Refer to Configuring SRLG Node Protection for more information.

Example

The following example shows how to enable the Flexible Algorithm ASLA-specific advertisement of SRLGs and to exclude SRLG groups from Flexible Algorithm path computation:

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/1/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit
```

```
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/1/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# name groupX value 100
RP/0/RP0/CPU0:router(config-srlg)# name groupY value 200
RP/0/RP0/CPU0:router(config-srlg)# exit

RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-af)# advertise application flex-algo link-attributes srlg
RP/0/RP0/CPU0:router(config-isis-af)# exit
RP/0/RP0/CPU0:router(config-isis)# flex-algo 128
RP/0/RP0/CPU0:router(config-isis-flex-algo)# advertise-definition
RP/0/RP0/CPU0:router(config-isis-flex-algo)# srlg exclude-any groupX
RP/0/RP0/CPU0:router(config-isis-flex-algo)# exit
RP/0/RP0/CPU0:router(config-isis)# flex-algo 129
RP/0/RP0/CPU0:router(config-isis-flex-algo)# advertise-definition
RP/0/RP0/CPU0:router(config-isis-flex-algo)# srlg exclude-any groupY
RP/0/RP0/CPU0:router(config-isis-flex-algo)# commit
RP/0/RP0/CPU0:router(config-isis-flex-algo)# exit
RP/0/RP0/CPU0:router(config-isis)#
```

The following example shows how to enable the Flexible Algorithm ASLA-specific advertisement of SRLGs and to exclude SRLG groups from Flexible Algorithm path computation for OSPF:

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupX
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# interface HunGigE0/0/1/0
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/1/1
RP/0/RP0/CPU0:router(config-srlg-if)# name groupY
RP/0/RP0/CPU0:router(config-srlg-if)# exit

RP/0/RP0/CPU0:router(config-srlg)# name groupX value 100
RP/0/RP0/CPU0:router(config-srlg)# name groupY value 200
RP/0/RP0/CPU0:router(config-srlg)# exit

RP/0/0/CPU0:r1(config)#router ospf 1
RP/0/0/CPU0:r1(config-ospf)#flex-algo 128
RP/0/0/CPU0:r1(config-ospf-flex-algo)#srlg exclude-any
RP/0/0/CPU0:r(config-ospf-flex-algo-srlg-exclude-any)#groupX
RP/0/0/CPU0:r(config-ospf-flex-algo-srlg-exclude-any)#groupY
RP/0/0/CPU0:r(config-ospf-flex-algo-srlg-exclude-any)#commit
```

Verification

The following example shows how to verify the number of SRLGs excluded for OSPF:

```
RP/0/RP0/CPU0:router# show ospf topology summary
 Process ospf-1
 Instance default
```

```
  Router ID       : 192.168.0.1
  Number of Areas : 1
  Number of Algos : 1
  Max Path count  : 16
  Route count     : 10
  SR Global Block : 16000 - 23999

Area 0
    Number of Nodes : 6
     Algo 128
      FAD Advertising Router : 192.168.0.1
      FAD Area ID : 0
       Algo Type   : 0
       Metric Type : 0
          Number of Exlclude SRLGs  : (2)
          [1]:  100        [2]:  200
       FAPM supported : No
```

Flexible Algorithm with Exclude Minimum Bandwidth Constraint

Table 61: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS Flexible Algorithm with Exclude Minimum Bandwidth Constraint	Release 7.11.1	Traffic engineering in networks can be optimized by avoiding low-bandwidth links that may not be capable of handling high volumes of traffic. This feature allows you to use Flexible Algorithm to create topologies in your network that explicitly exclude high bandwidth traffic from utilizing links below a specified capacity. This constraint is achieved by introducing a new bandwidth-based metric type within the Flexible Algorithm framework. Links that do not satisfy the constraint are ignored when computing the associated Flexible Algorithm topology. This feature introduces these changes: **CLI**: • The **router isis** *instance* **flex-algo** *algo* command is modified with the new **minimum-bandwidth** *value* option. **YANG Data Model:** • This feature extends the native `Cisco-IOS-XR-clns-isis-cfg.yang` model (see GitHub, YANG Data Models Navigator)

This feature allows you to configure a minimum bandwidth value for computing a Flexible Algorithm path.

The IS-IS Flex-Algorithm Exclude Minimum Bandwidth sub-TLV (FAEMB) is a way to set a minimum bandwidth requirement for links in the Flex-Algorithm topology.

To determine if a link should be excluded based on this minimum bandwidth requirement, we compare the Minimum Bandwidth specified in the FAEMB sub-TLV with the Maximum Link Bandwidth advertised in the Area Supported by the Link Attribute (ASLA) sub-TLV.

If the Maximum Link Bandwidth is lower than the Minimum bandwidth specified, the link is excluded from the Flex-Algorithm topology. However, if the FAD includes the FAEMB sub-TLV but the Maximum Link Bandwidth is not advertised for the link, it should not be excluded based on the Minimum Bandwidth constraint.

Use the **router isis** *instance* **flex-algo** *algo* **minimum-bandwidth** *value* command to configure the minimum bandwidth value in kbps.

Example: Configuring IS-IS Flexible Algorithm with Minimum Bandwidth Constraint

```
router isis 1
 address-family ipv4 unicast
  segment-routing mpls
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator L1_A129
   !
  !
 !
 flex-algo 129
  advertise-definition
  minimum-bandwidth 10000000
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 100
   prefix-sid algorithm 129 index 300
  !
  address-family ipv6 unicast
 !
!
segment-routing
 srv6
  locators
   locator L1_A129
    micro-segment behavior unode psp-usd
    prefix cafe:0:2100::/48
    algorithm 129
   !
  !
 !
!
```

Flexible Algorithm with Exclude Maximum Delay Constraint

Table 62: Feature History Table

Feature Name	Release Information	Feature Description
IS-IS Flexible Algorithm with Exclude Maximum Delay Constraint	Release 7.11.1	This feature enables you to configure topologies that exclude links that have delays over a specific threshold. This is especially critical for high-frequency trading applications, in satellite networks, or wherever there are fluctuations in link delays. This feature introduces these changes: **CLI**: • The **router isis** *instance* **flex-algo** *algo* command is modified with the new **maximum-delay** *value* option. **YANG Data Model:** • This feature extends the native Cisco-IOS-XR-clns-isis-cfg.yang model (see GitHub, YANG Data Models Navigator)

This feature allows you to configure a maximum delay value for computing a Flexible Algorithm path.

The Flexible Algorithm Exclude Minimum Delay (FAEMD) sub-TLV is used to specify the maximum delay requirement for links in a Flex-Algorithm topology. To ensure proper functioning, the FAEMD sub-TLV must appear only once in the FAD sub-TLV (Flexible Algorithm Definition). If it appears more than once, it should be ignored by the receiver. The maximum link delay advertised in the FAEMD sub-TLV is compared with the minimum unidirectional link delay advertised in the ASLA sub-TLV.

If the minimum unidirectional link delay is higher than the maximum link delay advertised in the FAEMD sub-TLV, the link must be excluded from the Flex-Algorithm topology.

However, if a link does not have the minimum unidirectional link delay advertised but the FAD contains the FAEMD sub-TLV, then based on the maximum delay constraint, that link should not be excluded from the topology.

Use the **router isis** *instance* **flex-algo** *algo* **maximum-delay** *delay* command to configure the maximum delay value in microseconds.

Example: Configuring IS-IS Flexible Algorithm with Maximum Delay Constraint

```
router isis 1
 address-family ipv4 unicast
  segment-routing mpls
 !
 address-family ipv6 unicast
  segment-routing srv6
   locator L1_A128
   !
  !
 !
 flex-algo 128
  advertise-definition
  maximum-delay 300
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 100
   prefix-sid algorithm 128 index 200
  !
  address-family ipv6 unicast
 !
!
segment-routing
 srv6
  locators
   locator L1_A128
    micro-segment behavior unode psp-usd
    prefix cafe:0:1100::/48
    algorithm 128
   !
  !
 !
!
```

Maximum Paths Per IS-IS Flexible Algorithm Per Prefix

Table 63: Feature History Table

Feature Name	Release	Description
Maximum Paths Per IS-IS Flexible Algorithm Per Prefix	Release 7.11.1	Previously, you could configure a maximum number of Equal-Cost Multi-path (ECMP) to be set for individual Flex Algorithms. This feature provides additional granularity to the IS-IS Maximum Paths Per-Algorithm feature by allowing you to specify a set of prefixes for Flexible Algorithm. Now you can achieve a balance between path diversity and computational and memory requirements by controlling the number of paths for each specific algorithm and destination prefix combination. This feature introduces these changes: **CLI** • **maximum-paths route-policy** *name* YANG Data Models: • This feature extends the native `Cisco-IOS-XR-clns-isis-cfg.yang` model See GitHub, Yang Data Models Navigator

Previously, you could set the maximum paths for a Flexible Algorithm per address-family.

With this feature, you can further refine the maximum paths configuration by associating it with specific prefixes for each Flexible Algorithm. The existing **maximum-paths** command is extended to include a **route-policy** qualifier to configure the maximum paths per algorithm per prefix-list.

When installing paths into the Routing Information Base (RIB) for Segment Routing with IPv6 (SRv6) or the Label Switched Database (LSD) for Segment Routing with MPLS (SR-MPLS), the system checks if a maximum paths value has been configured for the algorithm and the associated prefix. If such a configuration exists, it will be used instead of the existing address-family value to determine the number of paths to be installed.

Note Route policies that have the attribute **set maximum-paths** *number* are supported.

Note For information on maximum paths per prefix for IS-IS algo 0 (SPF), refer to the "Implementing IS-IS" chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*.

Usage Guidelines and Limitations

- The **maximum-paths** *maximum-paths* and **maximum-paths route-policy** *name* configurations are mutually exclusive. You can configure either an unqualified number or a route-policy for any given IS-IS instance.
- The maximum paths per-algorithm per-prefix configuration takes precedence over maximum paths per-algorithm configuration. Likewise, the maximum paths per-algorithm configuration takes precedence over maximum paths per-address-family configuration. This hierarchy ensures that the most specific configuration is prioritized when determining the maximum paths for a given algorithm and prefix combination.

Example

The following example shows how to configure the maximum paths for Flex Algo 128:

- **Define a Prefix Set:**

```
prefix-set isis-ipv4-L1
  10.1.0.101/32
end-set
```

- **Create a Route Policy:**

```
route-policy isis-mp-if-L1
  if destination in isis-ipv4-L1 then
    set maximum-paths 2
  endif
end-policy
```

- **Apply Route Policy to Configure Maximum Paths Per-Algo Per-Prefix:**

```
router isis 10
 flex-algo 128
  address-family ipv4 unicast
   maximum-paths route-policy isis-mp-if-L1
```

Verification

```
Router# show isis route flex-algo 128

IS-IS 10 IPv4 Unicast routes Flex-Algo 128

Codes: L1 - level 1, L2 - level 2, ia - interarea (leaked into level 1)
       df - level 1 default (closest attached router), su - summary null
       C - connected, S - static, R - RIP, B - BGP, O - OSPF
       E - EIGRP, A - access/subscriber, M - mobile, a - application
       i - IS-IS (redistributed from another instance)
```

```
Maximum parallel path count: as defined in isis-mp-if-L1

L1 10.1.0.101/32 [121/115]
     via 15.15.15.2, GigabitEthernet0/0/0/5, hare, SRGB Base: 16000, Weight: 0
     via 16.16.16.2, GigabitEthernet0/0/0/6, hare, SRGB Base: 16000, Weight: 0
```

Example: Configuring IS-IS Flexible Algorithm

```
router isis 1
 affinity-map red bit-position 65
 affinity-map blue bit-position 8
 affinity-map green bit-position 201

 flex-algo 128
  advertise-definition
  affinity exclude-any red
  affinity include-any blue
 !
 flex-algo 129
  affinity exclude-any green
 !
!
address family ipv4 unicast
 segment-routing mpls
!
interface Loopback0
 address-family ipv4 unicast
  prefix-sid algorithm 128 index 100
  prefix-sid algorithm 129 index 101
!
!
interface GigabitEthernet0/0/0/0
 affinity flex-algo red
!
interface GigabitEthernet0/0/0/1
 affinity flex-algo blue red
!
interface GigabitEthernet0/0/0/2
 affinity flex-algo blue
!
```

Example: Configuring OSPF Flexible Algorithm

```
router ospf 1
 flex-algo 130
  priority 200
  affinity exclude-any
   red
   blue
  !
  metric-type delay
 !
 flex-algo 140
  affinity include-all
   green
  !
  affinity include-any
   red
```

```
 !
!

 interface Loopback0
  prefix-sid index 10
  prefix-sid strict-spf index 40
  prefix-sid algorithm 128 absolute 16128
  prefix-sid algorithm 129 index 129
  prefix-sid algorithm 200 index 20
  prefix-sid algorithm 210 index 30
 !
!

 interface GigabitEthernet0/0/0/0
  flex-algo affinity
   color red
   color blue
  !
 !

affinity-map
 color red bit-position 10
 color blue bit-position 11
!
```

User-Defined Generic Metric Support for IS-IS Flex Algo

Table 64: Feature History Table

Feature Name	Release Information	Feature Description
User-Defined Generic Metric Support for IS-IS Flex Algo	Release 24.2.11	This feature adds support for user-defined generic metric as a metric type for IS-IS Flexible Algorithm. You can now have more control over traffic flows using user-defined generic metrics. You can define a family of user-defined generic metrics that can advertise different types of administrative metrics such as jitter, reliability, and fiscal cost depending on the traffic class for Flexible Algorithms. You can selectively define and assign semantics of these metrics as per the network requirement. The feature introduces the following changes: **CLI**: • The feature introduces the **generic-metric flex-algo** and **metric-type generic** commands. **YANG Data Models:** • `Cisco-IOS-XR-um-router-isis-cfg.yang`

Control Traffic Flow with User-Defined Generic Metrics

With the addition of different traffic types, the need for alternate types of metrics has evolved. Flexible Algorithm already supports IGP, TE, and delay metrics. However a network operator might want to minimize their operational costs and might want a metric that reflects the actual fiscal costs of using a link. Other traffic may require low jitter, leading to an entirely different set of metrics. With Flexible Algorithm, all these different metrics could be used concurrently on the same network. These improvements are possible as you can now define a family of user-defined generic metrics that can advertise various types of administratively assigned metrics. These metrics are not predefined, which provides network administrators with the flexibility to assign their own meanings and semantics to the metrics. This means you can create metrics tailored to specific operational goals or traffic requirements.

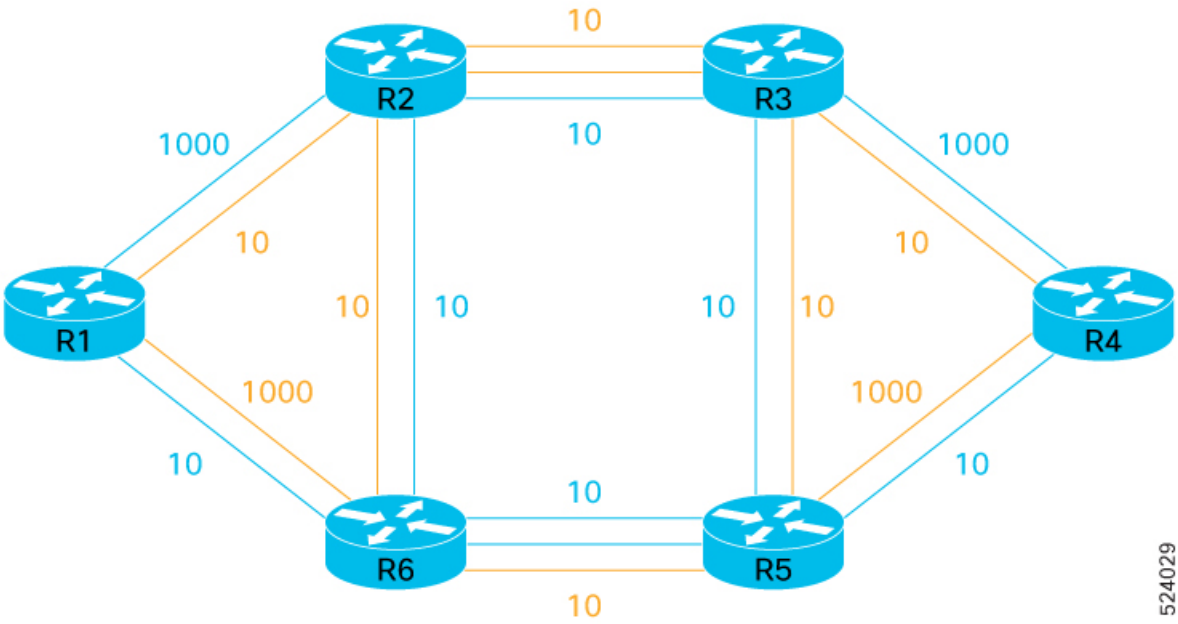
Apply User-Defined Generic Metrics in Flexible Algorithm Definition

You can apply the user-defined generic metrics within the Flexible Algorithm Definition (FAD) similar to how you would use TE or delay metrics. For example, you can assign specific user-defined generic metrics to individual Flex Algos, allowing for a customized path selection criteria. This capability enables different paths to use different metrics, ensuring that each path is optimized for the specific type of traffic it handles.

Split traffic with User-Defined Generic Metrics

Consider a scenario where for some networks, traffic engineering might require splitting east and west traffic for different Flexible Algorithms across multiple hardware data planes. In the following figure, the Flex Algo 128 (Orange) uses the upper primary path, while the Flex Algo 129 (Blue) uses the lower primary path. You can achieve this by defining specific user-defined generic metrics for the Flex Algo to determine the best path for each Flex Algo. The numbers 10 and 100 represent the user-defined generic metrics configured for the two different Flex Algos.

Figure 46: Splitting traffic using user-defined generic metrics for different Flex Algos



Advantages of splitting traffic using user-defined generic metrics

This approach offers several advantages. You can use Equal-Cost Multi-Path (ECMP) routing within the backbone, allowing multiple links of the same color between core nodes. It also ensures efficient backup routes and prevents U-loops with Flex Algo specific Loop-Free Alternates (LFA) and Topology-Independent Loop-Free Alternates (TI-LFA).

In the above example, the vertical core links are used as backup paths for all Flex Algos, and you can activate them only if a core link or node fails. These links are part of all Flex Algos, and you can use user-defined generic metrics to configure and advertise such metrics for each individual Flex Algo to ensure smooth operation.

Benefits of User-Defined Generic Metrics

The key benefits of user-defined generic metrics are:

- You can ensure precise control over routing decisions based on the assigned metrics, as the links that do not advertise the user-defined generic metric are excluded from the Flex Algo topology.
- You can use the metrics to reduce operational expenses by choosing cost-effective paths.
- You can customize metrics to fit unique operational goals and traffic requirements, providing tailored solutions for different scenarios.
- You can simultaneously apply multiple metrics across the same network, enhancing overall network performance and reliability.

Usage Guidelines and Limitations for User-Defined Generic Metrics

The user-defined generic metric is disabled by default. In other words, the metric is not advertised unless it is configured.

Configure User-Defined Generic Metrics

This section includes configuration for user-defined generic metrics for IS-IS Flex Algos.

Procedure

Step 1 Define the metric type and value for an interface using the **generic-metric flex-algo** command in the IS-IS interface address-family submode. In the following example, the user-defined generic metric type is 177 and the metric value as 100.

Example:

```
Router(config)#router isis 1
Router(config-isis)#interface GigabitEthernet 0/2/0/7
Router(config-isis-if)#address-family ipv4 unicast
Router(config-isis-if-af)#generic-metric flex-algo type 128 100
Router(config-isis-if-af)#generic-metric flex-algo type 177 100
Router(config-isis-if-af)#generic-metric flex-algo type 188 1000
```

Step 2 Associate or advertise the configured metric type to a Flexible Algorithm Definition using the **metric-type generic** command. In the following example, the metric type advertised is 177.

```
Router(config)#router isis 1
Router(config-isis)#flex-algo 128
Router(config-isis-flex-algo)#priority 254
Router(config-isis-flex-algo)#metric-type generic 177
Router(config-isis-flex-algo)#advertise-definition
```

Step 3 Verify the running configuration using the **show running-config** command.

Example:

```
router isis 1
 flex-algo 128
  priority 254
  metric-type generic 177
  advertise-definition
 !
 interface GigabitEthernet 0/2/0/7
  address-family ipv4 unicast
```

```
   generic-metric flex-algo type 128 100
   generic-metric flex-algo type 177 100
   generic-metric flex-algo type 188 1000
  !
 !
!
```

Step 4 Verify the Flexible Algorithm Definition configuration using the **show isis flex-algo** command.

Example:

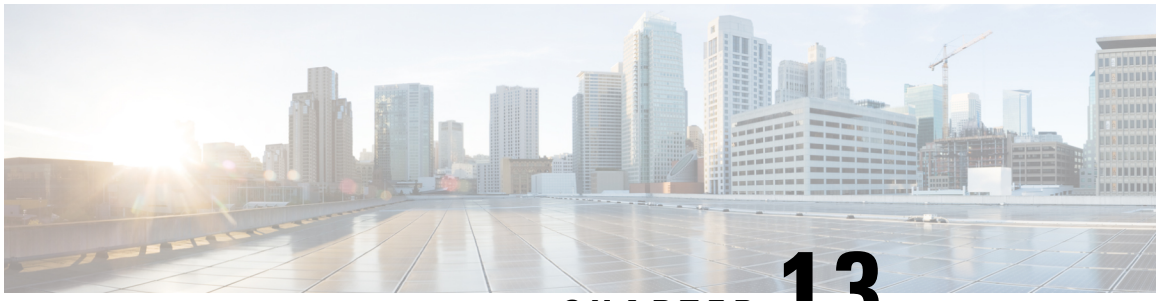
```
Router#show isis flex-algo 128
Thu Dec 7 03:10:56.452 PST
IS-IS 1 Flex-Algo Database
Flex-Algo 128:
Level-2:
Definition Priority: 254
Definition Source: plzen.00, (Local)
Definition Equal to Local: Yes
Definition Metric Type: User-defined: 177.
Definition Flex-Algo Prefix Metric: No
Exclude Any Affinity Bit Positions:
Include Any Affinity Bit Positions:
Include All Affinity Bit Positions:
Reverse Exclude Any Affinity Bit Positions:
Reverse Include Any Affinity Bit Positions:
Reverse Include All Affinity Bit Positions:
Exclude SRLGs:
Minimum Link Bandwidth: 0 kbits/s
Maximum Link Delay: 0 us
Disabled: No
Topologies supported:
IPv4 Unicast
Local Priority: 254
FRR Disabled: No
Microloop Avoidance Disabled: No
UCMP Disabled: No
Data Plane Segment Routing: Yes
Data Plane IP: No
```

Step 5 Optionally, you can also verify the application-specific user-defined generic metric configured for an interface by using the **show isis interface** command.

Example:

```
Router#show isis interface GigabitEthernet 0/2/0/7
Thu Dec 7 03:13:34.140 PST
GigabitEthernet0/2/0/7 Enabled
Adjacency Formation: Enabled
Prefix Advertisement: Enabled
Bandwidth: 1000000
 ...
 ...
IPv4 Unicast Topology: Enabled
Adjacency Formation: Running
Prefix Advertisement: Running
Policy (L1/L2): -/-
Metric (L1/L2): 10/10
Metric fallback:
Bandwidth (L1/L2): Inactive/Inactive
Anomaly (L1/L2): Inactive/Inactive
Weight (L1/L2): 0/0
    L1 Flex-algo Generic-metrics:
       Type: 128            100
       Type: 177            100
```

```
   Type: 188            1000
L2 Flex-algo Generic-metrics:
   Type: 128            100
   Type: 177            100
   Type: 188            1000
```



CHAPTER 13

Configure Segment Routing Path Computation Element

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

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

Table 65: Feature History Table

Feature Name	Release Information	Feature Description
Segment Routing Path Computation Element	Release 7.5.2	You can use a recommended platform to act as the Segment Routing Path Computation Element (SR-PCE) to calculate a suitable network path for transmitting data between a source and destination by applying metrics such as IGP, TE, and latency and restrictions such as the affinity of flexible algorithms for delay or IGP, and disjointness for LSPs. SR-PCE supports up to: • 50000 nodes • 100000 LSPs • 500000 links • 2000 PCEP sessions You can use SR-PCE for: • Disjoint Policy, on page 518 • PCE-initiated SR Policies for Traffic Management, on page 519 • PCC-initiated Policies Delegated to PCE, on page 532 • SR-PCE IPv4 Unnumbered Interface Support, on page 533

- About SR-PCE, on page 515
- Usage Guidelines and Limitations, on page 516
- Configure SR-PCE, on page 516
- Disjoint Policy, on page 518
- PCE-initiated SR Policies for Traffic Management, on page 519
- Exclude Network Resources during Path Computation over SR-TE Policies, on page 523
- Enable Strict Disjointness for SR-TE policies in the PCE, on page 528
- Configure the Shortest Path for Disjoint Candidate Paths, on page 530
- PCC-initiated Policies Delegated to PCE, on page 532
- SR-PCE IPv4 Unnumbered Interface Support, on page 533
- Inter-Domain Path Computation Using Redistributed SID, on page 535

About SR-PCE

Table 66: Feature History Table

Feature Name	Release Information	Feature Description
TCP Authentication Option	Release 7.3.1	This feature introduces support for TCP Authentication Option (TCP-AO), which replaces the TCP Message Digest 5 (MD5) option, which was used for authenticating PCEP (TCP) sessions by using a clear text or encrypted password.

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

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

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

- TE metric—SR-PCE uses the TE metric in its path calculations to optimize cumulative TE metric.
- IGP metric—SR-PCE uses the IGP metric in its path calculations to optimize reachability.
- LSP Disjointness—SR-PCE uses the path computation algorithms to compute a pair of disjoint LSPs. The disjoint paths can originate from the same head-end or different head-ends. Disjoint level refers to the type of resources that should not be shared by the two computed paths.

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

TCP Authentication Option

Transmission Control Protocol (TCP) Message Digest 5 (MD5) authentication is used for authenticating PCEP (TCP) sessions by using clear text or encrypted password. This feature introduces support for TCP Authentication Option (TCP-AO), which replaces the TCP MD5 option.

TCP-AO uses Message Authentication Codes (MACs), which provides the following:

- Protection against replays for long-lived TCP connections
- More details on the security association with TCP connections than TCP MD5
- A larger set of MACs with minimal system and operational changes

TCP-AO is compatible with Primary Key Tuple (PKT) configuration. TCP-AO also protects connections when using the same PKT across repeated instances of a connection. TCP-AO protects the connections by using a traffic key that is derived from the PKT, and then coordinates changes between the endpoints.

Note TCP-AO and TCP MD5 are never permitted to use simultaneously. TCP-AO supports IPv6, and is fully compatible with the proposed requirements for the replacement of TCP MD5.

Usage Guidelines and Limitations

To ensure PCEP compatibility, we recommend that the Cisco IOS XR version on the SR-PCE be the same or later than the Cisco IOS XR version on the PCC or head-end.

These are the unsupported configurations for SR-MPLS TE Path Computation for IPv6:

Configure SR-PCE

This task explains how to configure SR-PCE.

Before you begin

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

SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

	Command or Action	Purpose
Step 2	**pce** **Example:** `RP/0/RP0/CPU0:router(config)# pce`	Enables PCE and enters PCE configuration mode.
Step 3	**address ipv4** *address* **Example:** `RP/0/RP0/CPU0:router(config-pce)# address ipv4 192.168.0.1`	Configures a PCE IPv4 address.
Step 4	**state-sync ipv4** *address* **Example:** `RP/0/RP0/CPU0:router(config-pce)# state-sync ipv4 192.168.0.3`	Configures the remote peer for state synchronization.
Step 5	**tcp-buffer** *size* **Example:** `RP/0/RP0/CPU0:router(config-pce)# tcp-buffer 1024000`	Configures the transmit and receive TCP buffer size for each PCEP session, in bytes. The default buffer size is 256000. The valid range is from 204800 to 1024000.
Step 6	**password** {**clear** | **encrypted**} *password* **Example:** `RP/0/RP0/CPU0:router(config-pce)# password encrypted pwd1`	Enables TCP authentication for all PCEP peers. Any TCP segment coming from the PCC that does not contain a MAC matching the configured password will be rejected. Specify if the password is encrypted or clear text. **Note** TCP-AO and TCP MD5 are never permitted to be used simultaneously.
Step 7	**tcp-ao** *key-chain* [**include-tcp-options**] [**accept-ao-mismatch-connection**] **Example:** `RP/0/RP0/CPU0:router(config-pce)# tcp-ao pce_tcp_ao include-tcp-options`	Enables TCP Authentication Option (TCP-AO) authentication for all PCEP peers. Any TCP segment coming from the PCC that does not contain a MAC matching the configured key chain will be rejected. • **include-tcp-options**—Includes other TCP options in the header for MAC calculation. • **accept-ao-mismatch-connection**—Accepts connection even if there is a mismatch of AO options between peers. **Note** TCP-AO and TCP MD5 are never permitted to be used simultaneously.

	Command or Action	Purpose
Step 8	**segment-routing** {**strict-sid-only** | **te-latency**} **Example:** `RP/0/RP0/CPU0:router(config-pce)# segment-routing strict-sid-only`	Configures the segment routing algorithm to use strict SID or TE latency. **Note** This setting is global and applies to all LSPs that request a path from this controller.
Step 9	**timers** **Example:** `RP/0/RP0/CPU0:router(config-pce)# timers`	Enters timer configuration mode.
Step 10	**keepalive** *time* **Example:** `RP/0/RP0/CPU0:router(config-pce-timers)# keepalive 60`	Configures the timer value for locally generated keep-alive messages. The default time is 30 seconds.
Step 11	**minimum-peer-keepalive** *time* **Example:** `RP/0/RP0/CPU0:router(config-pce-timers)# minimum-peer-keepalive 30`	Configures the minimum acceptable keep-alive timer that the remote peer may propose in the PCEP OPEN message during session establishment. The default time is 20 seconds.
Step 12	**reoptimization** *time* **Example:** `RP/0/RP0/CPU0:router(config-pce-timers)# reoptimization 30`	Configures the re-optimization timer. The default timer is 60 seconds.
Step 13	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pce-timers)# exit`	Exits timer configuration mode and returns to PCE configuration mode.

Disjoint Policy

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

You can configure the disjoint group ID and the preferred level of disjointness (the type of resources that must not be shared by the two paths). SR-PCE supports the following disjoint path computations:

- link—Specifies that links are not shared on the computed paths.
- node—Specifies that nodes are not shared on the computed paths.
- srlg—Specifies that links with the same SRLG value are not shared on the computed paths.
- srlg-node—Specifies that SRLG and nodes are not shared on the computed paths.

If a pair of paths that meet the requested disjointness level cannot be found, then the paths automatically fallback to a lower level in the following way:

- 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.

Configure Disjoint Policy

Perform the following task to configure disjoint policy.

Configuration Example

```
Router)# configure
Router(config)# pce
Router(config-pce)# disjoint-path
Router(config-pce-disjoint)# group-id 1 type node sub-id 1
Router(config-pce-disjoint)# strict
```

strict is an optional parameter. It 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 is stopped and no new path is provided. The existing path is not modified.

```
Router(config-pce-disjoint)# lsp 1 pcc ipv4 192.168.0.1 lsp-name rtrA_t1 shortest-path
```

The above configuration adds LSPs to the disjoint group. The **shortest-path** 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.

```
Router(config-pce-disjoint# commit
```

PCE-initiated SR Policies for Traffic Management

An SR-TE policy can be configured on the path computation element (PCE) to reduce link congestion or to minimize the number of network touch points.

Note The PCE-initiated SR-TE policies are entered in PCE configuration mode. For more information on configuring SR-TE policies, see the SR-TE Policy Overview, on page 350.

The PCE collects network information, such as traffic demand and link utilization. When the PCE determines that a link is congested, it identifies one or more flows that are causing the congestion. The PCE finds a suitable path and deploys an SR-TE policy to divert those flows, without moving the congestion to another part of the network. When there is no more link congestion, the policy is removed.

To minimize the number of network touch points, an application, such as a Network Services Orchestrator (NSO), can request the PCE to create an SR-TE policy. PCE deploys the SR-TE policy using PCC-PCE communication protocol (PCEP).

PCEP defines the communication between PCE and PCE as specified in the following steps:

1. PCE sends a PCInitiate message to the PCC.
2. If the PCInitiate message is valid, the PCC sends a PCRpt message; otherwise, it sends PCErr message.
3. If the PCInitiate message is accepted, the PCE updates the SR-TE policy by sending PCUpd message.

You can achieve high-availability by configuring multiple PCEs with SR-TE policies. If the head-end (PCC) loses connectivity with one PCE, another PCE can assume control of the SR-TE policy.

Configure PCE-initiated SR Policy

To configure a PCE-initiated SR-TE policy, you must complete the following configurations:

1. Enter PCE configuration mode.
2. Create the segment list.
3. Create the policy.

Configure PCE-initiated SR Policy with Explicit SID List

Perform the following task to configure PCE-initiated SR policy with explicit SID list.

Configuration Example

```
/* Enter PCE configuration mode and create the SR-TE segment lists */
Router# configure
Router(config)# pce

/* Create the SR-TE segment lists */
Router(config-pce)# segment-routing
Router(config-pce-sr)# traffic-eng
Router(config-pce-sr-te)# segment-list name addr2a
Router(config-pce-sr-te-sl)# index 1 address ipv4 192.168.0.1
Router(config-pce-sr-te-sl)# exit

/* Create the SR-TE policy */
Router(config-pce-sr-te)# peer ipv4 10.1.1.1
Router(config-pce-sr-te)# policy P1
Router(config-pce-sr-te-policy)# color 2 end-point ipv4 172.16.0.1
Router(config-pce-sr-te-policy)# candidate-paths
Router(config-pce-sr-te-policy-path)# preference 50
Router(config-pce-sr-te-policy-path-preference)# explicit segment-list addr2a
Router(config-pce-sr-te-pp-info)# end
```

Running Configuration

```
pce
 segment-routing
  traffic-eng
   segment-list name addr2a
    index 1 address ipv4 192.168.0.1
   !
   peer ipv4 10.1.1.1
    policy P1
     color 2 end-point ipv4 172.16.0.1
     candidate-paths
      preference 50
       explicit segment-list addr2a
      !
     !
```

Configure PCE-initiated SR Policy with Dynamic SID List

Perform the following task to configure PCE-initiated SR policy with dynamic SID list.

Configuration Example

In this example, the PCE creates the policy and computes the path.

```
/* Enter PCE configuration mode */
Router# configure
Router(config)# pce

/* Create the SR-TE policy */
Router(config-pce)# segment-routing
Router(config-pce-sr)# traffic-eng
Router(config-pce-sr-te)# peer ipv4 10.1.1.1
Router(config-pce-sr-te)# policy P1
Router(config-pce-sr-te-policy)# color 2 end-point ipv4 172.16.0.1
Router(config-pce-sr-te-policy)# binding-sid mpls 10001
Router(config-pce-sr-te-policy)# candidate-paths
Router(config-pce-sr-te-policy-path)# preference 50
Router(config-pce-sr-te-policy-path-preference)# dynamic mpls
Router(config-pce-sr-te-pp-info)# metric type igp
Router(config-pce-sr-te-pp-info)# commit
```

Running Configuration

```
pce
 segment-routing
  traffic-eng
   peer ipv4 10.1.1.1
   policy P1
    binding-sid mpls 10001
    color 2 end-point ipv4 172.16.0.1
    candidate-paths
     preference 50
      dynamic mpls
       metric
        type igp
       !
      !
     !
```

```
        !
      !
    !
  !
!
```

Exclude Network Resources during Path Computation over SR-TE Policies

Table 67: Feature History Table

Feature Name	Release Information	Feature Description
Exclude Network Resources during Path Computation over SR-TE Policies	Release 24.1.1	

Feature Name	Release Information	Feature Description
		You can reduce the risk of attacks due to less secure network resources and IP addresses, improve network performance affected by overloaded or congested paths, and ensure higher levels of network stability and availability that could be impacted by resources experiencing issues and requiring maintenance. These improvements are possible because you can now exclude specific network resources using their IP addresses during path computation for traffic over SR-TE policies. Previously, you could not exclude network resources during path computation. The feature introduces these changes: **CLI:** • The **resources** option is introduced in the **segment-routing traffic-eng policy** and **segment-routing traffic-eng on-demand color** commands. • The **resource-list** keyword is introduced in the **segment-routing traffic-eng** command. **YANG Data Models:** • `Cisco-IOS-XR-infra-xtc-oper.yang` • `Cisco-IOS-XR-infra-xtc-agent-oper.yang` • `Cisco-IOS-XR-infra-xtc-agent-cfg.yang` See (GitHub, Yang Data Models Navigator)

You can configure the SR-TE policies as either single or disjoint candidate paths that do not include certain IP addresses or subnets when traffic is routed through the network.

The key benefits of the feature are:

- **Security** - Excluding certain IP addresses or network resources can help protect sensitive resources from unauthorized access or attacks.
- **Performance** - By excluding certain resources, traffic can be directed away from congested parts of the network.
- **Isolation** - Excluding resources can help isolate different parts of the network, preventing issues in one part of the network from affecting others.

Configuration to exclude network resources during Path Computation

To exclude network resources during path computation for traffic, you must complete these high-level tasks in order:

1. Configure the network resources or IP addresses that you want to exclude from the network list.
2. Associate the excluded network resources to candidate paths for SR-TE or ODN SR-TE policies.

Configure the network resources or IP addresses to exclude from the network list

Perform the following task to configure a list of IPv4 addresses that you want to exclude from the network resource list:

```
Router(config)#segment-routing traffic-eng
Router(config-sr-te)#resource-list node_resc_list
Router(config-sr-te-rl)#index 1 ipv4 10.10.10.1
Router(config-sr-te-rl)#index 2 ipv4 10.10.10.8
```

Running Configuration

```
!
segment-routing
 traffic-eng
  resource-list node_resc_list
   index 1 ipv4 10.10.10.1
   index 2 ipv4 10.10.10.8
  !
 !
!
```

Associate the excluded network resources to candidate paths for SR-TE policies

Perform the following task to associate the excluded IPv4 addresses to one or more candidate paths for SR-TE policies:

```
Router(config)#segment-routing traffic-eng
Router(config-sr-te)#policy dynamic_pcep_policy
Router(config-sr-te-policy)#candidate-paths
Router(config-sr-te-policy-path)#preference 100
Router(config-sr-te-policy-path-pref)#constraints resources exclude resource-list
node_resc_list
```

Running Configuration

```
!
segment-routing
 traffic-eng
  policy dynamic_pcep_policy
   candidate-paths
```

```
    preference 100
     constraints
      resources
       exclude resource-list node_resc_list
      !
     !
    !
   !
  !
 !
!
```

Verification

```
Router#show segment-routing traffic-eng policy endpoint ipv4 100.2.1.1 color 8001

Wed Aug 30 22:21:50.014 UTC
SR-TE policy database
---------------------
Color: 8001, End-point: 100.2.1.1
Name: srte_c_8001_ep_100.2.1.1
Status:
Admin: up Operational: up for 00:00:50 (since Aug 30 22:20:59.341)
Candidate-paths:
Preference: 100 (configuration) (active)
Name: dynamic_pcep_policy
Requested BSID: 8001
PCC info:
Symbolic name: cfg_dynamic_pcep_policy_discr_100
PLSP-ID: 14042
Constraints:
Protection Type: protected-preferred
Maximum SID Depth: 12
Exclude Resources: node_resc_list
10.10.10.1
10.10.10.8
Dynamic (pce 100.7.1.1) (valid)
Metric Type: TE, Path Accumulated Metric: 440
SID[0]: 41111 [Adjacency-SID, 101.1.5.1 - 101.1.5.2]
SID[1]: 21600 [Prefix-SID, 100.6.1.1]
SID[2]: 41600 [Adjacency-SID, 101.2.6.2 - 101.2.6.1]
Attributes:
Binding SID: 8001
Forward Class: Not Configured
Steering labeled-services disabled: yes
Steering BGP disabled: no
IPv6 caps enable: yes
Invalidation drop enabled: no
Max Install Standby Candidate Paths: 0
```

Associate the excluded network resources to candidate paths for ODN SR-TE policies

Perform the following task to associate the excluded IPv4 addresses for ODN SR-TE policies:

```
Router(config)#segment-routing
Router(config-sr)#traffic-eng
Router(config-sr-te)#on-demand color 7001
Routerconfig-sr-te-color)#constraints resources exclude resource-list node_resc_list
```

Running Configuration

```
!
segment-routing
 traffic-eng
  on-demand color 7001
```

```
   constraints
    resources
     exclude resource-list node_resc_list
    !
   !
  !
 !
!
```

Verification

```
Router#show segment-routing traffic-eng policy endpoint ipv4 100.2.1.1 color 7001

Wed Aug 30 22:53:01.079 UTC

SR-TE policy database
---------------------

Color: 7001, End-point: 100.2.1.1
  Name: srte_c_7001_ep_100.2.1.1
  Status:
    Admin: up  Operational: up for 00:31:56 (since Aug 30 22:21:04.869)
  Candidate-paths:
    Preference: 200 (BGP ODN) (inactive) (shutdown)
      Requested BSID: dynamic
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 12
        Exclude Resources: node_resc_list
          10.10.10.1
          10.10.10.8
      Dynamic (inactive)
        Metric Type: IGP,   Path Accumulated Metric: 0
    Preference: 100 (BGP ODN) (active)
      Requested BSID: dynamic
      PCC info:
        Symbolic name: bgp_c_7001_ep_100.2.1.1_discr_100
        PLSP-ID: 14044
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 12
        Exclude Resources: node_resc_list
          10.10.10.1
          10.10.10.8
      Dynamic (pce 100.7.1.1) (valid)
        Metric Type: IGP,   Path Accumulated Metric: 440
          SID[0]: 41111 [Adjacency-SID, 101.1.5.1 - 101.1.5.2]
          SID[1]: 21600 [Prefix-SID, 100.6.1.1]
          SID[2]: 41600 [Adjacency-SID, 101.2.6.2 - 101.2.6.1]
  Attributes:
    Binding SID: 51679
    Forward Class: Not Configured
    Steering labeled-services disabled: yes
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
```

Enable Strict Disjointness for SR-TE policies in the PCE

Table 68: Feature History Table

Feature Name	Release Information	Feature Description
Enable Strict Disjointness for SR-TE Policies in the PCE	Release 24.1.1	You can now enforce strict disjoint constraints for Label Switched Paths (LSPs) and minimize the risk of a single point of failure that affects multiple LSPs. With this feature, if disjoint paths cannot be found for LSPs, the Path Computation Element (PCE) does not return any path. Previously, enforcing strict disjoint constraints for disjoint paths for LSPs was not possible. The feature introduces these changes: **CLI:** The **fallback disable** keyword is introduced in the **segment-routing traffic-eng policy** and **segment-routing traffic-eng on-demand color** commands. **YANG Data Models:** • `Cisco-IOS-XR-infra-xtc-oper.yang` • `Cisco-IOS-XR-infra-xtc-agent-oper.yang` • `Cisco-IOS-XR-infra-xtc-agent-cfg.yang` See (GitHub, Yang Data Models Navigator)

When you enable fallback disable, the Path Computation Element (PCE) does not return any path for LSPs that are not disjoint if there are no disjoint candidate paths. In other words, when the PCE does not find multiple LSPs that satisfy the disjointness requirement, it returns no path. This indicates that there might be a networking issue that prevents the establishment of the desired redundancy, and you can investigate and address the underlying problem.

Without the option, the PCE can relax disjointness by applying an objective function or use a local policy when no objective function is requested.

Enforcing strict disjointness ensures that LSPs do not share any common links or nodes along their paths, minimizing the risk of a single point of failure affecting multiple LSPs. The feature is critical to maintain network reliability and ensures that traffic is properly routed in the event of a network failure.

Configure Strict Disjointness in the PCE

Enable strict disjointness for ODN SR-TE policies

Perform the following steps to enable strict disjointness for ODN SR-TE policies:

```
Router(config)#segment-routing traffic-eng
Router(config-sr-te)#on-demand color 4
Router(config-sr-te-color)#dynamic
Router(config-sr-te-color-dyn)#disjoint-path group-id 1 type node fallback disable
Router(config-sr-te-color-dyn)#commit
```

Running Configuration

```
segment-routing
 traffic-eng
  on-demand color 4
   dynamic
    disjoint-path group-id 1 type node fallback disable
   !
  !
 !
!
```

Enable strict disjointness for SR-TE policies

Perform the following steps to enable strict disjointness for SR-TE policies:

```
Router(config)#segment-routing traffic-eng
Router(config-sr-te)#policy foo
Router(config-sr-te-policy)#color 1 end-point ipv4 10.10.10.1
Router(config-sr-te-policy)#candidate-paths preference 100
Router(config-sr-te-policy-path-pref)#constraints disjoint-path group-id 1 type node fallback
 disable
Router(config-sr-te-policy-path-pref)#commit
```

Running Configuration

```
segment-routing
 traffic-eng
  policy foo
   color 1 end-point ipv4 10.10.10.1
   candidate-paths
    preference 100
     dynamic
      pcep
      !
      metric
       type latency
      !
     !
     constraints
      disjoint-path group-id 1 type node fallback disable
     !
    !
   !
  !
 !
!
```

Verification

```
Router#sh pce association
Wed Mar 13 14:38:27.173 PDT
```

```
PCE's association database:
----------------------
Association: Type Node-Disjoint, Group 1, Strict
 Associated LSPs:
  LSP[0]:
   PCC 10.10.10.1, tunnel name cfg_foo_discr_100,  PLSP ID 4, tunnel ID 8, LSP ID 2,
Configured on PCC
  LSP[1]:
   PCC 10.10.10.1, tunnel name bgp_c_4_ep_10.10.10.2_discr_100,  PLSP ID 1, tunnel ID 5,
LSP ID 3, Configured on PCC
 Status: Satisfied
```

Configure the Shortest Path for Disjoint Candidate Paths

Table 69: Feature History Table

Feature Name	Release Information	Feature Description
Configure the Shortest Path for Disjoint Candidate Paths	Release 24.1.1	You can now configure the available disjoint paths for Label Switched Paths (LSPs) to prefer the shortest path between two points in the network. This configuration ensures that traffic is routed along the most efficient route in the network. Previously, you could not configure the shortest path preference for disjoint LSPs. The feature introduces these changes: **CLI:** The **shortest-path** keyword is introduced in the **policy candidate-paths constraints disjoint-path** command. **YANG Data Models:** • `Cisco-IOS-XR-infra-xtc-oper.yang` • `Cisco-IOS-XR-infra-xtc-agent-oper.yang` • `Cisco-IOS-XR-infra-xtc-agent-cfg.yang` See (GitHub, Yang Data Models Navigator)

The configuration enables the available disjoint paths to indicate preference for the shortest path between two points in the network. This way the traffic is contained within specific network planes as per the affinity

constraints while still provisioning Label Switched Paths (LSPs) that are diverse from each other to meet the path disjointness requirement.

In earlier releases, if path disjointness was configured for LSPs, the Path Computation Element (PCE) prioritized finding disjoint paths for the LSPs over adhering to the affinity constraints.

Configure the shortest path for disjoint candidate paths

Perform the following task to indicate the disjoint path preference for the shortest path in the network:

```
Router(config)#segment-routing traffic-eng
Router(config-sr-te)#policy dynamic_pcep_policy_disjoint
Router(config-sr-te-policy)#candidate-paths
Router(config-sr-te-policy-path)#preference 100
Router(config-sr-te-policy-path-pref)#constraints disjoint-path group-id 1 type link
shortest-path
```

Running Configuration

```
!
segment-routing
 traffic-eng
   policy dynamic_pcep_policy
   candidate-paths
    preference 100
     constraints
      disjoint-path group-id 1 type link shortest-path

     !
    !
   !
  !
 !
!
```

Verification

```
Router#show pce lsp name cfg_dynamic_pcep_policy_disjoint_discr_100 detail
Output received:
Wed Aug 30 18:33:57.807 UTC

PCE's tunnel database:
----------------------
PCC 100.1.1.1:

Tunnel Name: cfg_dynamic_pcep_policy_disjoint_discr_100
Color: 30115
Interface Name: srte_c_30115_ep_100.2.1.1
 LSPs:
  LSP[0]:
   source 100.1.1.1, destination 100.2.1.1, tunnel ID 9031, LSP ID 24
   State: Admin up, Operation active
   Setup type: Segment Routing
   Binding SID: 30115
   Maximum SID Depth: 12
   Preference: 100
   Bandwidth: requested 0 kbps, applied 0 kbps
   Protection type: protected-preferred
   Prefix-SID algorithm: 0 (set: 0)
   PCEP information:
     PLSP-ID 0x36e9, flags: D:1 S:0 R:0 A:1 O:2 C:0
   LSP Role: Exclude LSP
   State-sync PCE: None
```

```
PCC: 100.1.1.1
LSP is subdelegated to: None
Reported path:
  Metric type: TE, Accumulated Metric 30
   SID[0]: Node, Label 21200, Address 100.2.1.1
Computed path: (Local PCE)
  Computed Time: Wed Aug 30 18:31:12 UTC 2023 (00:02:45 ago)
  Metric type: TE, Accumulated Metric 30
   SID[0]: Node, Label 21200, Address 100.2.1.1
Reverse path: (Local PCE)
  None
  Computed Time: Wed Aug 30 18:31:12 UTC 2023 (00:02:45 ago)
Recorded path:
  None
Disjoint Group Information:
  Type Link-Disjoint, Group 1 (SP)
SR Policy Association Group:
  Color: 30115, Endpoint: 100.2.1.1
  Policy Name: srte_c_30115_ep_100.2.1.1
  Preference: 100
  CP Name: dynamic_pcep_policy_disjoint
```

PCC-initiated Policies Delegated to PCE

Policies are created on PCC and a path is requested from PCE. PCEP connection must be up and functional for PCE to perform the path computation.

Configure PCC-initiated Policies Delegated to PCE

Perform the following task to configure the SR policies at PCC and delegate them to PCE after establishing PCEP connection.

Configuration Example

```
Router # configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy local_dynamic_disj_1
Router(config-sr-te-policy)# binding-sid mpls 19002
Router(config-sr-te-policy)# color 19002 end-point ipv4 192.168.0.1
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy-path)# preference 100
Router(config-sr-te-policy-path-preference)# dynamic
Router(config-sr-te-pp-info)# pcep
Router(config-sr-te-path-pcep)# exit
Router(config-sr-te-pp-info)# metric
Router(config-sr-te-path-metric)# type te
Router(config-sr-te-path-metric)# exit
Router(config-sr-te-pp-info)# exit
Router(config-sr-te-policy-path-preference)# constraints
Router(config-sr-te-path-pref-const)# affinity
Router(config-sr-te-path-pref-const-aff)# include-any
Router(config-sr-te-path-pref-const-aff-rule)# name blue
Router(config-sr-te-path-pref-const-aff-rule)# name green
Router(config-sr-te-path-pref-const-aff-rule)# commit
```

Running Configuration

```
Router # show running-config segment-routing traffic-eng policy local_dynamic_disj_1
segment-routing
 traffic-eng
  policy local_dynamic_disj_1
   binding-sid mpls 19002
   color 19002 end-point ipv4 192.168.0.1
   candidate-paths
    preference 100
     dynamic
      pcep
      !
      metric
       type te
      !
     !
     constraints
      affinity
       include-any
        name blue
        name green
       !
      !
     !
    !
   !
  !
 !
!
```

SR-PCE IPv4 Unnumbered Interface Support

This feature allows IPv4 unnumbered interfaces to be part of an SR-PCE topology database.

An unnumbered IPv4 interface is not identified by its own unique IPv4 address. Instead, it is identified by the router ID of the node where this interfaces resides and the local SNMP index assigned for this interface.

This feature provides enhancements to the following components:

- IGPs (IS-IS and OSPF):
  - Support the IPv4 unnumbered interfaces in the SR-TE context by flooding the necessary interface information in the topology
- SR-PCE:

Note SR-PCE and path computation clients (PCCs) need to be running Cisco IOS XR software release 7.5.2 or later.

  - Compute and return paths from a topology containing IPv4 unnumbered interfaces.
  - Process reported SR policies from a head-end router that contain hops with IPv4 unnumbered adjacencies.

    PCEP extensions for IPv4 unnumbered interfaces adhere to IETF RFC8664 “PCEP Extensions for Segment Routing” (https://datatracker.ietf.org/doc/rfc8664/). The unnumbered hops use a Node or

Adjacency Identifier (NAI) of type 5. This indicates that the segment in the explicit routing object (ERO) is an unnumbered adjacency with an IPv4 ID and an interface index.

- SR-TE process at the head-end router:
  - Compute its own local path over a topology, including unnumbered interfaces.
  - Process PCE-computed paths that contain hops with IPv4 unnumbered interfaces.
  - Report a path that contains hops with IPv4 unnumbered interfaces to the PCE.

Configure SR-PCE IPv4 Unnumbered Interface

This section provides the commands to configure the SR-PCE IPv4 unnumbered interface.

Configuration Example

The following example shows how to configure an IPv4 unnumbered interface:

```
Router # configure
Router(config)# interface GigabitEthernet0/0/0/0
Router(config-if)# ipv4 point-to-point
Router(config-if)# ipv4 unnumbered Loopback0
```

To bring up the IPv4 unnumbered adjacency under the IGP, configure the link as point-to-point under the IGP configuration. The following example shows how to configure the link as point-to-point under the IGP configuration:

```
Router # configure
Router(config)# router ospf one
Router(config-ospf)# area 0
Router(config-ospf-ar)# interface GigabitEthernet0/0/0/0
Router(config-ospf-ar-if)# network point-to-point
```

Verification

Use the **show ipv4 interface** command to display information about the interface:

```
Router # show ipv4 interface GigabitEthernet0/0/0/0 brief
Tue Apr  2 12:59:53.140 EDT
Interface                      IP-Address      Status                Protocol
GigabitEthernet0/0/0/0         192.168.0.1     Up                    Up
```

This interface shows the IPv4 address of Loopback0.

Use the **show snmp interface** command to find the SNMP index for this interface:

```
Router# show snmp interface
Tue Apr  2 13:02:49.190 EDT
ifName : Null0                 ifIndex : 3
ifName : Loopback0             ifIndex : 10
ifName : GigabitEthernet0/0/0/0  ifIndex : 6
```

The interface is identified with the pair (IPv4:192.168.0.1, index:6).

Use the **show ospf neighbor** command to display the adjacency:

```
Router# show ospf neighbor gigabitEthernet 0/0/0/0 detail
…
```

```
Neighbor 192.168.0.4, interface address 192.168.0.4
    In the area 0 via interface GigabitEthernet0/0/0/0
    Neighbor priority is 1, State is FULL, 6 state changes
    …
    Adjacency SIDs:
        Label: 24001,    Dynamic, Unprotected
    Neighbor Interface ID: 4
```

The output of the **show pce ipv4 topology** command is enhanced to display the interface index instead of the IP address for unnumbered interfaces:

```
Router# show pce ipv4 topology
…
  Link[2]: unnumbered local index 6, remote index 4
    Local node:
      OSPF router ID: 192.168.0.1 area ID: 0 ASN: 0
    Remote node:
      TE router ID: 192.168.0.4
      OSPF router ID: 192.168.0.4 area ID: 0 ASN: 0
    Metric: IGP 1, TE 1, Latency 1 microseconds
    Bandwidth: Total 125000000 Bps, Reservable 0 Bps
    Admin-groups: 0x00000000
    Adj SID: 24001 (unprotected)
```

The output of **show pce lsp detail** command includes unnumbered hops:

```
Router# show pce lsp detail
…
   Reported path:
     Metric type: TE, Accumulated Metric 3
       SID[0]: Adj unnumbered, Label 24001, local 192.168.0.1(6), remote 192.168.0.4(4)
       SID[1]: Adj unnumbered, Label 24002, local 192.168.0.4(7), remote 192.168.0.3(7)
       SID[2]: Adj unnumbered, Label 24000, local 192.168.0.3(5), remote 192.168.0.2(5)
   Computed path: (Local PCE)
     Computed Time: Wed Apr 03 11:01:46 EDT 2019 (00:01:06 ago)
     Metric type: TE, Accumulated Metric 3
       SID[0]: Adj unnumbered, Label 24001, local 192.168.0.1(6), remote 192.168.0.4(4)
       SID[1]: Adj unnumbered, Label 24002, local 192.168.0.4(7), remote 192.168.0.3(7)
       SID[2]: Adj unnumbered, Label 24000, local 192.168.0.3(5), remote 192.168.0.2(5)
```

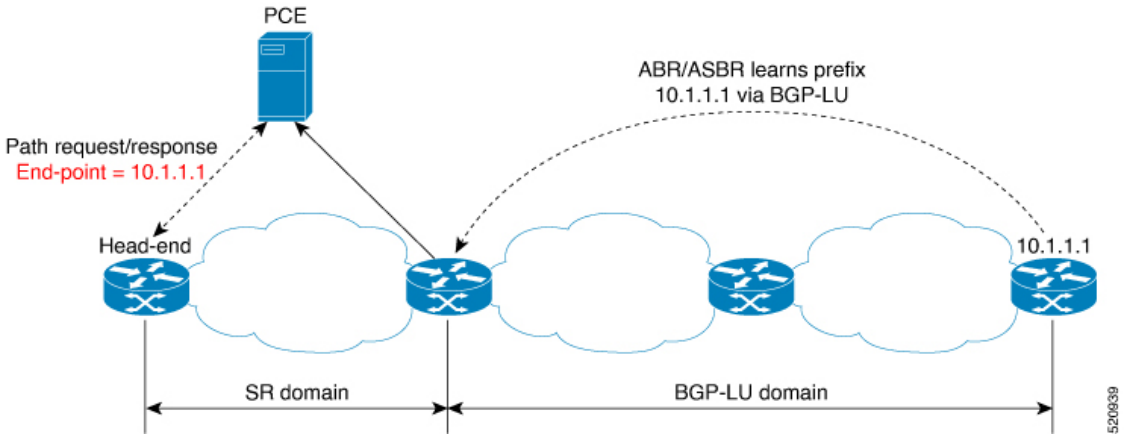
Inter-Domain Path Computation Using Redistributed SID

A Path Computation Element (PCE) computes SR-TE paths based on SR topology database that stores connectivity, state, and TE attributes of SR network nodes and links. BGP Labeled Unicast (BGP-LU) provides MPLS transport across IGP boundaries by advertising loopbacks and label binding of impact edge and border routers across IGP boundaries.

This feature adds new functionality to the SR-PCE that enables it to compute a path for remote non-SR end-point device distributed by BGP-LU.

The remote end-point device in the BGP-LU domain is unknown to the SR-PCE. For the SR-PCE to know about the end-point device, the gateway ABR/ASBR learns the end-point prefix via BGP-LU. The prefix is then redistributed to SR-PCE topology database from the gateway ABR/ASBR. SR-PCE then can compute the best path from the head-end device to the selected gateway router.

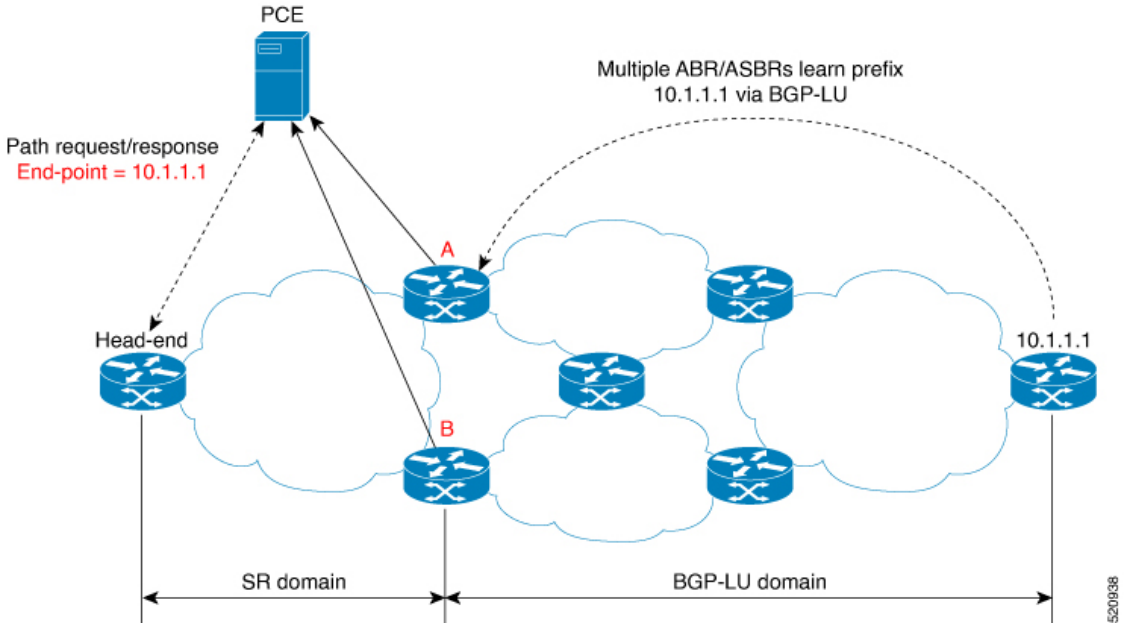
The following topology shows an SR domain and a BGP-LU domain, with a gateway ABR/ASBR between the two domains.



1. The gateway ABR/ASBR is configured with BGP/IGP helper to learn the remote prefix through BGP-LU and redistribute the remote prefix to the IGP helper, then to SR-PCE.
2. The SR-PCE selects the best gateway node to BGP-LU domain and computes the path to reach the remote prefix through the gateway node.
3. The head-end device in the SR domain requests a path to the remote destination and signals the SR profile interworking with the BGP-LU domain.

The BGP-LU prefix advertisement to SR-PCE Traffic Engineer Database (TED) is done by creating an IGP helper on the ABR/ASBR to redistribute BGP-LU prefix information to IGP. IGP then sends the prefix information to the SR-PCE via BGP-LS.

If there are multiple ABR/ASBRs advertising the same remote BGP-LU prefix, the SR-PCE selects the best gateway node to the BGP-LU domain using the accumulative metric from the head-end device to the gateway and the advertised metric from the gateway to the destination.



Example: Inter-Domain Path Computation Using Redistributed SID

The following examples show the configurations for the IGP helper, BGP-LU, and proxy BGP-SR:

Configuration on the End-Point Device

Configure the end-point device to allocate a label for the BGP-LU prefix on the end-point device:

```
router bgp 3107
bgp router-id 1.0.0.8
address-family ipv4 unicast
  network 1.0.0.8/32 route-policy bgplu-com
  allocate-label all

route-policy bgplu-com
  set community (65002:999)
end-policy
```

Configuration on the Gateway ABR/ASBR

1. Configure the remote prefix set and create the route policy for the BGP-LU domain:

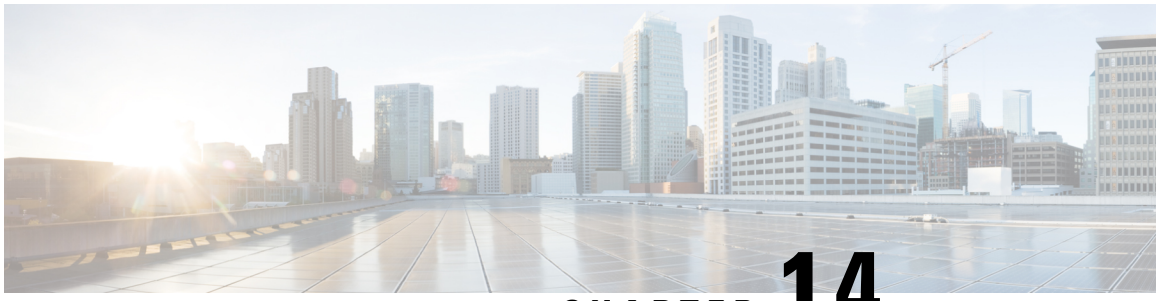
```
prefix-set bgplu
  1.0.0.7/32,
  1.0.0.8/32,
  1.0.0.101/32,
  1.0.0.102/32
end-set
!

route-policy bgp2isis
  if destination in bgplu then
    pass
  else
    drop
  endif
end-policy
!
end
```

2. Configure the helper IGP instance on the Loopback interface:

```
router isis 101
 is-type level-2-only
 net 49.0001.0000.1010.1010.00
 distribute link-state instance-id 9999
 nsf cisco
 nsf lifetime 120
 address-family ipv4 unicast
  metric-style wide
  maximum-paths 64
  router-id Loopback10
  redistribute bgp 3107 metric 200 route-policy bgp2isis
  segment-routing mpls sr-prefer
!
interface Loopback10 >>> this loopback is for gateway SR-TE node-id
  passive
  address-family ipv4 unicast
   prefix-sid index 2001 explicit-null
```

3. Configure the gateway proxy BGP-SR and SR Mapping Server to allocate SR labels:

```
router bgp 3107
 address-family ipv4 unicast
  segment-routing prefix-sid-map
  allocate-label all

segment-routing
global-block 16000 23999
mapping-server
  prefix-sid-map
   address-family ipv4
    1.0.0.7/32 2007
    1.0.0.8/32 2008
    1.0.0.101/32 2101
    1.0.0.102/32 2102
```



CHAPTER 14

Configure Performance Measurement

Network performance metrics is a critical measure for traffic engineering (TE) in service provider networks. Network performance metrics include the following:

- Packet loss
- Delay
- Delay variation
- Bandwidth utilization

These network performance metrics provide network operators information about the performance characteristics of their networks for performance evaluation and help to ensure compliance with service level agreements. The service-level agreements (SLAs) of service providers depend on the ability to measure and monitor these network performance metrics. Network operators can use Segment Routing Performance Measurement (SR-PM) feature to monitor the network metrics for links and end-to-end TE label switched paths (LSPs).

The following table explains the functionalities supported by performance measurement feature for measuring delay for links or SR policies.

Table 70: Performance Measurement Functionalities

Functionality	Details
Profiles	You can configure different profiles for different types of delay measurements. Delay profile type interfaces is used for link-delay measurement. Delay profile type sr-policy is used for SR policy delay measurements. Delay profile allows you to schedule probe and configure metric advertisement parameters for delay measurement.
Protocols	Two-Way Active Measurement Protocol (TWAMP) Light (using RFC 5357 with IP/UDP encap).
Probe and burst scheduling	Schedule probes and configure metric advertisement parameters for delay measurement.
Metric advertisements	Advertise measured metrics periodically using configured thresholds. Also supports accelerated advertisements using configured thresholds.
Measurement history and counters	Maintain packet delay and loss measurement history and also session counters and packet advertisement counters.

- Liveness Monitoring, on page 540
- Delay Measurement, on page 563
- Path Tracing in SRv6 Network, on page 591
- Two-Way Active Measurement Protocol Light Source Address Filtering, on page 601
- Synthetic Loss Measurement, on page 605

Liveness Monitoring

Liveness refers to the ability of the network to confirm that a specific path, segment, or a node is operational and capable of forwarding packets. Liveness checks are essential for maintaining network availability and reliability.

Benefits

- **Fault Detection**: You can quickly identify if a device is down, which allows for immediate response and troubleshooting.
- **Load Balancing**: You can identify if the devices in a network are live, so work can be distributed more evenly across the network, preventing overloading of specific components and improving overall performance.
- **System Health**: You can provide an ongoing snapshot of a system's health, helping to identify potential issues before they become significant problems.
- **Maintenance Planning**: Liveness information can also help with maintenance planning, as system administrators can understand which components are live or down and plan maintenance and downtime accordingly without significant disruption to services.
- **Security**: Regular liveness checks can also play a role in maintaining network security. Administrators can take proactive steps to mitigate the damage and prevent future incidents by identifying unusual activity that might indicate a security breach or attack.

You can determine liveness for SR Policy and IP Endpoint.

IP Endpoint Liveness Monitoring

Table 71: Feature History Table

Feature Name	Release Information	Feature Description
Liveness Monitoring for IP Endpoint over SRv6 Network	Release 24.2.11	In Segment Routing over an IPv6 network (SRv6), you can keep track of the operational status of both the forward and reverse paths of a particular node or IP endpoint. You can use this information for troubleshooting, network maintenance, and optimizing network performance. Additionally, you can use flow labels to verify the liveness of each subsequent hop path toward the IP endpoint of that path. So that, when network traffic is distributed across multiple available paths towards an IP endpoint, liveness detection tracks the operational status of each of these paths towards the IP endpoint. The feature introduces these changes: **CLI:** • The **reverse-path** and **segment-list name** keywords are introduced in the **segment-routing traffic-eng explicit** command. • The **source-address ipv6** is introduced in the **performance-measurement endpoint** command. **YANG Data Model:** • Cisco-IOS-XR-um-performance-measurement-cfg • Cisco-IOS-XR-perf-meas-oper.yang (see GitHub, YANG Data Models Navigator)

Feature Name	Release Information	Feature Description
IP Endpoint Liveness Monitoring	Release 7.4.1	This feature measures the end-to-end delay and monitors liveness of a specified IP endpoint node, including VRF-aware (awareness of multiple customers belonging to different VRFs). This feature is supported on IPv4, IPv6, and MPLS data planes.

The Segment Routing Performance Measurement (SR-PM) for IP endpoint liveness is a type of node liveness that involves testing whether an IP endpoint or a device identified by an IP address is available to send and receive data.

IP endpoint liveness is verified by sending a request to the IP address of the endpoint and waiting for a response. The probe could be an ICMP echo request (Ping), a TCP packet, a UDP packet, or any other type of packet that the endpoint would respond to.

- If a response is received, the endpoint is considered *live*.
- If no response is received within a certain time frame, the endpoint is considered *down* or *unreachable*.

IP endpoint dynamically measures the liveness towards a specified IP endpoint. IP endpoints can be located in a default or nondefault VRFs. IP endpoint is any device in the network a device identified by an IP address.

Liveness of an IP endpoint is verified by sending a request to the IP address of the endpoint and waiting for a response, which is referred to as a probe.

The endpoint of a probe is defined by an IP address, which can be either IPv4 or IPv6. This IP address can be any address that the sender can reach, such as a local interface or a remote node or host, either within an operator's network or accessible via a VRF.

The endpoint of a probe can be any IP address reachable by the sender. For example, a local interface or a remote node or host located within an operator's network or reachable through a VRF.

The IP address of the endpoint can be reached through an IP path, MPLS, LSP SRV6, or IP tunnel (GRE).

- When the endpoint is reachable using an MPLS LSP (for example, SR, LDP, RSVP-TE, SR Policy), the forwarding stage imposes the corresponding MPLS transport labels.
- When the endpoint is reachable via a GRE tunnel, the forwarding stage imposes the corresponding GRE header.
- When the endpoint is reachable via a VRF in an MPLS network, the forwarding stage imposes the corresponding MPLS service labels. In the forward path, the sender node uses the configured VRF for the endpoint address. In the return path, the reflector node derives the VRF based on which incoming VRF label the probe packet is received with.
- When the endpoint is reachable using SRv6, the forwarding stage imposes the SRv6 encapsulation.

You can configure the following parameters in the **performance-measurement** command:

- **Endpoint**: The endpoint of a probe is defined by an IP address, which can be either IPv4 or IPv6. This IP address can be any address that the sender can reach, such as a local interface or a remote node or

host, either within an operator's network or accessible via a VRF. The endpoint's IP address can be located in the global routing table or under a user-specified VRF routing table.

The endpoint of a probe can be any IP address reachable by the sender. For example, a local interface or a remote node or host located within an operator's network or reachable through a VRF.

Use the **performance-measurement endpoint** command to configure a probe endpoint source and destination addresses on a sender node.

- **VRF**: You can define the endpoint point IP address belonging to a specific VRF. Use the **performance-measurement endpoint {ipv4 | ipv6} ip_addr [vrf WORD]** command to configure an endpoint to define the VRF. Endpoint segment list configuration is not supported under nondefault VRF.
    - VRF-awareness allows operators to deploy probes in the following scenarios:
        - Managed Customer Equipment (CE) scenarios:
            - PE to CE probes
            - CE to CE probes
        - Unmanaged Customer Equipment (CE) scenarios:
            - PE to PE probes
            - PE to PE (source from PE-CE interface) probes

- **Source address**: You can define the source of the endpoint using the endpoint specific source address and the global source address.

  Global source address configuration is applied to all the endpoints when the endpoint specific source address configuration isn’t specified. endpoint specific configuration overrides all the global source address configuration for those specific endpoints for which source addresses are configured.

  For Micro-SID configuration for IPv4 endpoint sessions, if IPv6 global source address is configured, then it applies the configured global IPv6 source address for the IPv6 header in the SRv6 packet. If IPv6 global address is not configured, then It does not form a valid SRv6 packet.

  You can use the **source-address** keyword under the **performance-measurement** command to define the global source address or use the keyword under **performance-measurement endpoint** to define endpoint specific source address.

- Reverse Path: To detect the liveness of the reverse of the segment, you can configure the reverse path using the **reverse-path** command.

  The default reverse path configured under the endpoint submode is only used for sessions with segment list. The endpoint session without a segment list does not support reverse path configuration and will not use this reverse path.

  The **reverse-path** under the **performance-measurement endpoint** is used as the default reverse path if there are no reverse paths configured under the segment list.

  Use the **reverse-path** under the **performance-measurement endpoint segment-routing traffic-eng explicit segment-list name** to configure the reverse path under segment list.

  The reverse type must be the same as the forward path. Using different types for forward and reverse paths is not supported. For example, uSID forward path and uSID reverse path; MPLS forward path and MPLS reverse path.

User-configured segment-list can also represent the reverse path (reflector to sender) when probe is configured in liveness detection mode. Up to 128 segment-lists can be configured under a probe. An additional PM session is created for each segment-list. Segment-lists are configured under **segment-routing traffic-eng segment-list** submode. See SR-TE Policy with Explicit Path for details about configuring segment lists.

- Flow Label: The flow label field in the IPv6 header is used to carry information that helps distribute traffic across multiple network paths. The flow label is a 20-bit field in the IPv6 header designed to carry information about the flow of packets, which routers can use to identify and differentiate between different traffic flows. Flow label sweeping uses a flow label to distribute the traffic load across multiple paths to the endpoint.

  Use the **flow-label** keyword to configure flow label.

Supported and Unsupported Features

The table lists supported and unsupoted features for Liveness Monitoring for IP Endpoint over SRv6 Network.

Table 72: Supported and Unsupported Features for Liveness Monitoring for IP Endpoint over SRv6 Network

Supported Features	Unsupported Features
SRv6 Endpoint Liveness in Default VRF Example: *endpoint ipv6 fccc:2:: liveness-detection* In this example, the endpoint with the IPv6 address fccc:2:: is the SRv6 uSID format.	IPv6 Endpoint Liveness in Default VRF (over SRv6) Example: *endpoint ipv6 10::2 liveness-detection* In this example, the endpoint with the IPv6 address 2::2 is part of an underlay network using SRv6.
IPv6 Endpoint Liveness in VRF (static uDT6) Example: *endpoint ipv6 fccc:2:fe02:: liveness-detection* In this example, the endpoint with the IPv6 address fccc:2:fe02:: is the static uDT6 uSID carrier	IPv6 Endpoint Liveness in VRF (dynamic DT6 encap) Example: *endpoint ipv6 10::1 vrf purple source-address ipv6 10::2 liveness-detection*
IPv4 Endpoint Liveness in VRF or GRT (static uDT4) Example: *endpoint ipv4 10.5.56.1 source-address ipv4 10.1.17.1 segment-routing traffic-eng explicit segment-list name vrf-2-3-5-udt4 liveness-detection* The segment-list *vrf-2-3-5-udt4* here has static uDT4 SID.	IPv4 Endpoint Liveness in VRF or GRT (dynamic uDT4 encap Example: *endpoint ipv4 10.0.0.1 vrf purple source-address ipv4 10.0.0.2 liveness-detection*
	IPv6 address over SRV6 underlay Example: *endpoint ipv6 10::1 source-address ipv6 10::2 liveness-detection*

Usage Guidelines and Limitations

- For liveness detection, the session fails to come up when the endpoint address is a regular IPv4 address in a default VRF and that is a normal loopback IP address that uses IGP path. Packets get dropped with the following message. However, this issue does not apply if a segment list is configured.

  ```
  GRE IPv4 decap qualification failed
  ```

  To mitigate this issue, you must configure the GRE tunnel on responder. The following example shows how to configure GRE tunnel:

  ```
  /*Tunnel config on responder*\
  interface tunnel-ip1
  tunnel model ipv4 decap
  tunnel source 10.3.1.1
  tunnel destination 10.1.1.1
  ```

- Liveness session without segment list for an endpoint in a non-default VRF is not supported.
- SR Performance Measurement endpoint session over BVI interface is not supported.
- SRv6 locator prefix and VRF SRv6 locator/function (uDT4/uDT6) as IPv6 endpoint of a probe is not supported.
- IPv6 Endpoint Liveness in Default VRF is not supported over SRv6.
- PM probe over GREv4 is supported.

IP Endpoint Liveness Detection in an SR MPLS Network

IP endpoint liveness detection leverages the loopback measurement-mode. The following workflow describes the sequence of events.

1. The sender creates and transmits the PM probe packets.

   The IP destination address (DA) on the probe packets is set to the loopback value of the sender itself.

   The transmit timestamp (T1) is added to the payload.

   The probe packet is encapsulated with the label corresponding to the endpoint.

2. The network delivers the PM probe packets following the LSP toward the endpoint.
3. The end-point receives the PM probe packets.

   Packets are forwarded back to the sender based on the forwarding entry associated with the IP DA of the PM probe packet. If an LSP exists, the probe packet is encapsulated with the label of the sender.

4. The sender node receives the PM probe packets.

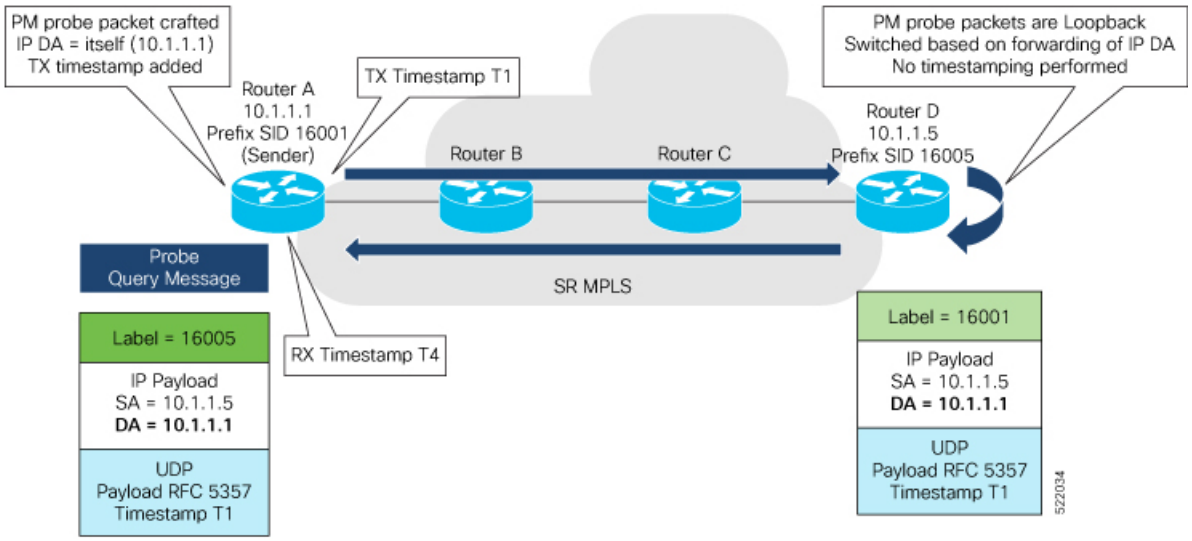
   The received timestamp (T4) stored.

   If the sender node doesn't receive the specified number of probe packets (based on the configured multiplier), the sender node declares the PM session as down.

The following figure illustrates a liveness detection probe toward an IP endpoint learned by the IGP. The network interconnecting the sender and reflector provides MPLS connectivity with Segment Routing.

The liveness detection multiplier is set to 5 to specify the number of consecutive missed probe packets before the PM session is declared as down.

Figure 47: IP Endpoint Liveness Detection



Configuration Example

```
RouterA(config)# performance-measurement
RouterA(config-perf-meas)# endpoint ipv4 1.1.1.5
RouterA(config-pm-ep)# source-address ipv4 1.1.1.1
RouterA(config-pm-ep)# liveness-detection
RouterA(config-pm-ep-ld)# exit
RouterA(config-pm-ep)# exit
RouterA(config-perf-meas)# liveness-profile endpoint default
RouterA(config-pm-ld-ep)# liveness-detection
RouterA(config-pm-ld-ep-ld)# multiplier 5
RouterA(config-pm-ld-ep-ld)# exit
RouterA(config-pm-ld-ep)# probe
RouterA(config-pm-ld-ep-probe)# measurement-mode loopback
```

Running Configuration

```
performance-measurement
 endpoint ipv4 1.1.1.5
  source-address ipv4 1.1.1.1
  liveness-detection
  !
 !
 liveness-profile endpoint default
  liveness-detection
   multiplier 5
  !
  probe
   measurement-mode loopback
  !
 !
!
end
```

Verification

```
RouterA# show performance-measurement endpoint ipv4 1.1.1.5
```

```
------------------------------------------------------------------------------
0/RSP0/CPU0
------------------------------------------------------------------------------

Endpoint name: IPv4-1.1.1.5-vrf-default
  Source address              : 1.1.1.1
  VRF name                    : default
  Liveness Detection          : Enabled
  Profile Keys:
    Profile name              : default
    Profile type              : Endpoint Liveness Detection

  Segment-list                : None
  Session State: Down
  Missed count: 0
```

IP Endpoint Liveness in an SRv6 Network

IP endpoint liveness detection leverages the loopback measurement-mode. The following workflow describes the sequence of events.

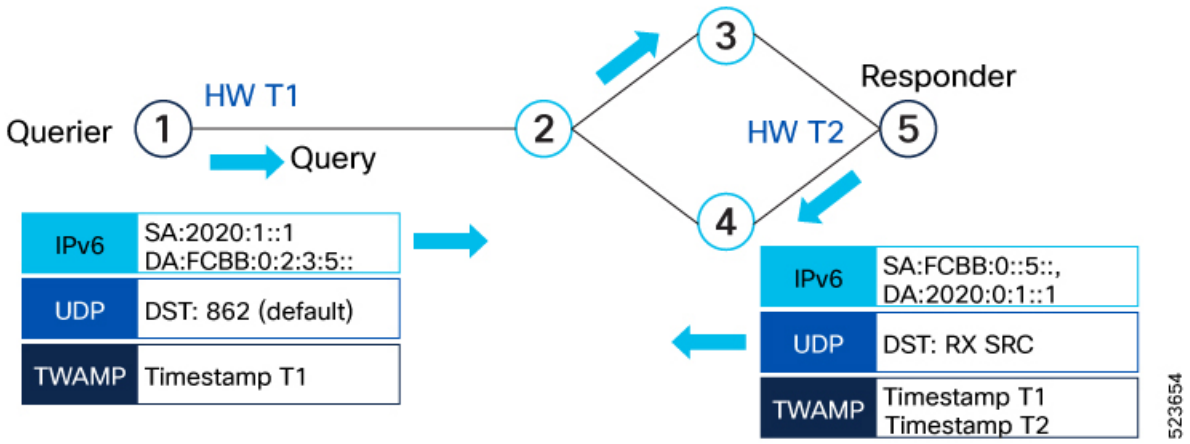
1. The querier creates and transmits the PM TWAMP probe packets based on the endpoint configuration.

   The packet is formed so it could reach the responder and return to the querier node over the SRv6 transport network.

2. The querier node declares the session up once it receives the probe packet back.

   If the sender node doesn't receive the specified number of probe packets consecutively based on the configured multiplier, it declares the PM session down.

Figure 48: IP Endpoint Liveness In an SRv6 Network



Note Liveness is not supported for non-default VRF.

Configuration Example

```
Router(config)#performance-measurement
Router(config-perf-meas)#source-address ipv6 2020:1::1
```

```
Router(config-perf-meas)#endpoint ipv6 FCBB:0::5::
Router(config-pm-ep)#exit
Router(config-perf-meas)#liveness-profile endpoint default
Router(config-pm-ld-ep)#probe
Router(config-pm-ld-ep-probe)#exit
Router(config-pm-ld-ep)#liveness-detection
Router(config-pm-ld-ep-ld)#multiplier 3
Router(config-pm-ld-ep-ld)#
```

The following example shows how to configure liveness with segment list and reverse path.

```
Router(config-sr)#traffic-eng
Router(config-sr-te)#segment-lists
Router(config-sr-te-segment-lists)#srv6
Router(config-sr-te-sl-global-srv6)#sid-format usid-f3216
Router(config-sr-te-sl-global-srv6)#exit
Router(config-sr-te-sl-global)#segment-list test
Router(config-sr-te-sl)#srv6
Router(config-sr-te-sl-srv6)#index 10 sid ff::2
Router(config-sr-te-sl-srv6)#index 20 sid ff::3
```

The following example shows how to configure liveness reverse path under segment list and under endpoint:

```
Router(config)#performance-measurement
Router(config-perf-meas)#endpoint ipv6 ff::2

/* Configure reverse path under segment list name *\
Router(config-pm-ep)#segment-routing traffic-eng explicit segment-list name fwd-path
Router(config-pm-ep-sl)#reverse-path segment-list name rev-path
Router(config-pm-ep-sl)#exit

/* Configure reverse path under performance measurement endpoint *\
Router(config-pm-ep)# segment-routing traffic-eng explicit reverse-path segment-list name
rev-path-name
```

The following example shows how to configure liveness with flow label:

```
Router(config-perf-meas)#liveness-profile endpoint default
Router(config-pm-ld-ep)#probe
Router(config-pm-ld-ep-probe)#flow-label from 1000 to 20000 increment 16
Router(config-pm-ld-ep-probe)#liveness-detection
Router(config-pm-ld-ep-ld)#multiplier 3
```

The following example shows how to configure liveness with flow label sweeping:

```
Router#configure
Router(config)#performance-measurement
Router(config-perf-meas)#liveness-profile name profile-sweeping
Router(config-pm-ld-profile)# flow-label from 1000 to 20000 increment 16
Routerconfig-pm-ld-profile)#commit
```

Verification

```
Router# show performance-measurement endpoint detail
Endpoint name: IPv6-FCBB:0::5:::-vrf-default
  Source address              : 2020:1::1
  VRF name                    : default
  Liveness Detection          : Enabled
  Profile Keys:
    Profile name              : default
    Profile type              : Endpoint Liveness Detection
  Segment-list                : None
  Liveness Detection session:
    Session ID                : 4109
    Flow-label                : 1000
```

```
  Session State: Up
  Last State Change Timestamp: Jan 23 2024 16:06:01.214
  Missed count: 0

Liveness Detection session:
  Session ID               : 4110
  Flow-label               : 2000
  Session State: Up
  Last State Change Timestamp: Jan 23 2024 16:06:01.214
  Missed count: 0

Segment-list               : test-dm-two-carrier-sl2
  FCBB:0::5:2:e004::/64
    Format: f3216
  FCBB:0::5:3:e000::/64
    Format: f3216
  FCBB:0::5:2:e004::/64
    Format: f3216
  FCBB:0::5:2:e000::/64
    Format: f3216
  FCBB:0::5:1:e000::/64
    Format: f3216
  FCBB:0::5:1:e004::/64
    Format: f3216
  FCBB:0::5:4:e000::/64
    Format: f3216
  FCBB:0::5:4::/48
    Format: f3216
Liveness Detection session:
  Session ID               : 4111
  Flow-label               : 1000
  Session State: Up
  Last State Change Timestamp: Jan 23 2024 16:06:01.217
  Missed count: 0

Liveness Detection session:
  Session ID               : 4112
  Flow-label               : 2000
  Session State: Up
  Last State Change Timestamp: Jan 23 2024 16:06:01.217
  Missed count: 0
```

SR Policy Liveness Monitoring

Table 73: Feature History Table

Feature Name	Release Information	Feature Description
SR Policy Liveness Monitoring on Segment Routing over IPv6 (SRv6)	Release 7.11.1	In segment routing over IPv6 (SRv6), you can now verify end-to-end traffic forwarding over an SR policy candidate path by periodically sending probe messages. Performance monitoring on an SRv6 network enables you to track and monitor traffic flows at a granular level. Earlier releases supported SR policy liveness monitoring over an SR policy candidate path on MPLS.
SR Policy Liveness Monitoring	Release 7.5.2	This feature allows you to verify end-to-end traffic forwarding over an SR Policy candidate path by periodically sending performance monitoring packets.

SR Policy liveness monitoring allows you to verify end-to-end traffic forwarding over an SR Policy candidate path by periodically sending probe messages. The head-end router sends PM packets to the SR policy's endpoint router, which sends them back to the head-end without any control-plane dependency on the endpoint router.

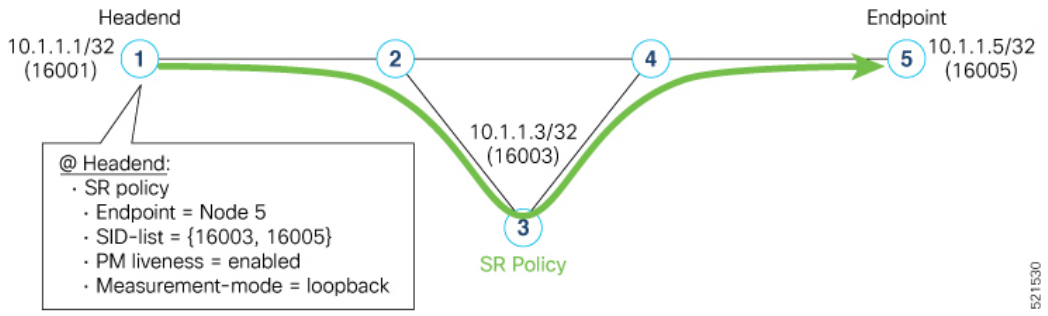
For more information about the segment routing over IPv6, see Segment Routing over IPv6 Overview topic.

The following are benefits to using SR-PM liveness monitoring:

- Allows both liveness monitoring and delay measurement using a single-set of PM packets as opposed to running separate monitoring sessions for each purpose. This improves the overall scale by reducing the number of PM sessions required.
- Eliminates network and device complexity by reducing the number of monitoring protocols on the network (for example, no need for Bidirectional Failure Detection [BFD]). It also simplifies the network and device operations by not requiring any signaling to bootstrap the performance monitoring session.
- Improves interoperability with third-party nodes because signaling protocols aren't required. In addition, it leverages the commonly supported TWAMP protocol for packet encoding.
- Improves liveness detection time because PM packets aren't punted on remote nodes
- Provides a common solution that applies to data-planes besides MPLS, including IPv4, IPv6, and SRv6.

The workflow associated with liveness detection over SR policy is described in the following sequence.

Consider an SR policy programmed at head-end node router 1 towards end-point node router 5. This SR policy is enabled for liveness detection using the loopback measurement-mode.



- **A**: The head-end node creates and transmits the PM probe packets.

  The IP destination address (DA) on the probe packets is set to the loopback value of the head-end node itself.

  A transmit (Tx) timestamp is added to the payload.

  Optionally, the head-end node may also insert extra encapsulation (labels) to enforce the reverse path at the endpoint node.

  Finally, the packet is injected into the data-plane using the same encapsulation (label stack) of that of the SR policy being monitored.

- **B**: The network delivers the PM probe packets as it would user packet for the SR policy.

- **C**: The end-point node receives the PM probe packets.

  Packets are switched back based on the forwarding entry associated with the IP DA of the packet. This would typically translate to the end-point node pushing the prefix SID label associated with the head-end node.

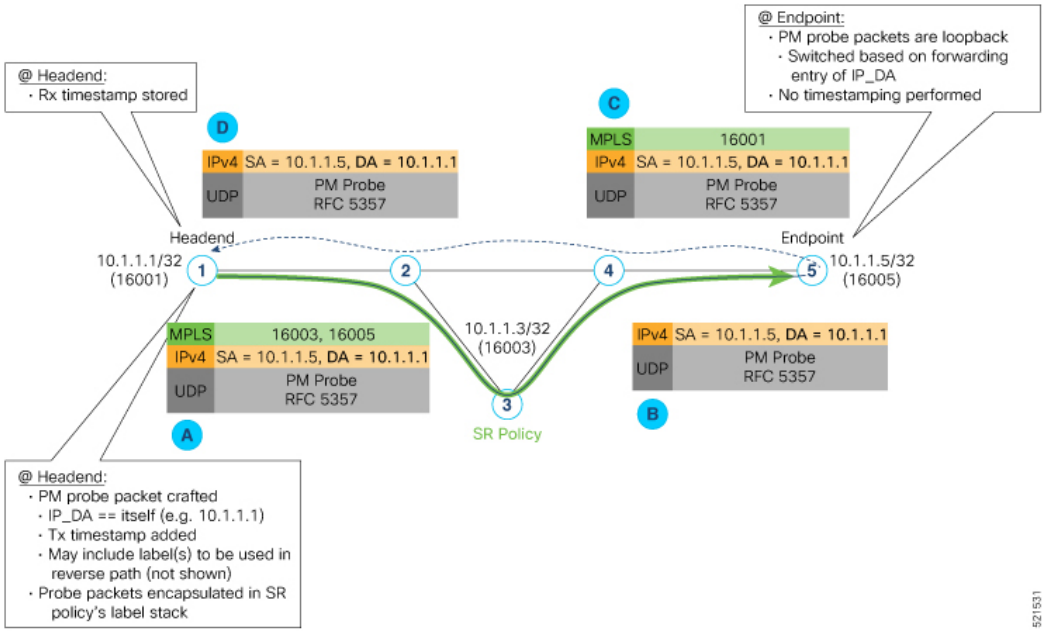
  If the head-end node inserted label(s) for the reverse path, then the packets are switched back at the end-point node based on the forwarding entry associated with the top-most reverse path label.

- **D**: Headend node receives the PM probe packets.

  A received (Rx) timestamp stored.

  If the head-end node receives the PM probe packets, the head-end node assume that the SR policy active candidate path is up and working.

  If the head-end node doesn't receive the specified number of consecutive probe packets (based on configured multiplier), the head-end node assumes the candidate path is down and a configured action is trigerred.



Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- Liveness-detection and delay-measurement aren't supported together
- When liveness-profile isn't configured, SR Policies use the default values for the liveness-detection profile parameters.

Configuring SR Policy Liveness Monitoring

Configuring SR Policy liveness monitoring involves the following steps:

- Configuring a performance measurement liveness profile to customize generic probe parameters
- Enabling liveness monitoring under SR Policy by associating a liveness profile, and customizing SR policy-specific probe parameters

Configuring Performance Measurement Liveness Profile

Liveness monitoring parameters are configured under sub-mode. The following parameters are configurable:

- **liveness-profile** {**sr-policy default** | **name** *name*}
- **probe**: Configure the probe parameters.
- **measurement-mode**: Liveness detection must use loopback mode (see Measurement Mode topic within this guide).
- **tx-interval**: Interval for sending probe packet. The default value is 3000000 microseconds and the range is from 3300 to 15000000 microseconds.
- **tos dscp** *value*: The default value is 48 and the range is from 0 to 63. You can modify the DSCP value of the probe packets, and use this value to priortize the probe packets from headend to tailend.

- **sweep destination ipv4 127.*x.x.x* range** *range*: Configure SR Policy ECMP IP-hashing mode. Specifiy the number of IP addresses to sweep. The range is from 0 (default, no sweeping) to 128. The option is applicable to IPv4 packets.

Note The destination IPv4 headendaddress 127.x.x.x – 127.y.y.y is used in the Probe messages to take advantages of 3-tuple IP hashing (source-address, destination-address, and local router ID) for ECMP paths of SR-MPLS Policy.

The destination IPv4 address must be 127/8 range (loopback), otherwise it will be rejected.

Note One PM session is always created for the actual endpoint address of the SR Policy.

- **liveness-detection**: Configure the liveness-detection parameters:
- **multiplier**: Number of consecutive missed probe packets before the PM session is declared as down. The range is from 2 to 10, and the default is 3.

Note The detection-interval is equal to (tx-interval * multiplier).

Enabling Liveness Monitoring under SR Policy

Enable liveness monitoring under SR Policy, associate a liveness-profile, and configure SR Policy-specific probe parameters under the **segment-routing traffic-eng policy performance-measurement** sub-mode. The following parameters are configurable:

- **liveness-detection**: Enables end-to-end SR Policy Liveness Detection for all segment-lists of the active and standby candidate-path that are in the forwarding table.
- **liveness-profile name** *name*: Specifies the profile name for named profiles.
- **invalidation-action {down | none}**:
  - **Down (default)**: When the PM liveness session goes down, the candidate path is immediately operationally brought down.
  - **None**: When the PM liveness session goes down, no action is taken. If logging is enabled, the failure is logged but the SR Policy operational state is not modified.
- **logging session-state-change**: Enables Syslog messages when the session state changes.
- **reverse-path label** {*BSID-value* | *NODE-SID-value* | *ADJACENCY-SID-value*}: Specifies the MPLS label to be used for the reverse path for the reply. If you configured liveness detection with ECMP hashing, you must specify the reverse path. The default reverse path uses IP Reply.
  - *BSID-value*: The Binding SID (BSID) label for the reverse SR Policy. (This is practical for manual SR policies with a manual BSID.)

- *NODE-SID-value*: The Node SID is a segment type that represents the ECMP-aware shortest path to reach a particular IP prefix from any IGP topology location
- *ADJACENCY-SID-value*: The absolute SID label of the (local) Sender Node to be used for the reverse path for the reply.

Configuration Examples

Configure a Default SR-Policy PM Liveness-Profile

The following example shows a default sr-policy liveness-profile:

```
RP/0/RSP0/CPU0:ios(config)# performance-measurement
RP/0/RSP0/CPU0:ios(config-perf-meas)# liveness-profile sr-policy default
RP/0/RSP0/CPU0:ios(config-pm-ld-srpolicy)# probe

RP/0/RSP0/CPU0:ios(config-pm-ld-srpolicy-probe)# tx-interval 150000
RP/0/RSP0/CPU0:ios(config-pm-ld-srpolicy-probe)# tos dscp 52
RP/0/RSP0/CPU0:ios(config-pm-ld-srpolicy-probe)# exit
RP/0/RSP0/CPU0:ios(config-pm-ld-srpolicy)# liveness-detection
RP/0/RSP0/CPU0:ios(config-pm-ld-srpolicy-ld)# multiplier 5
```

Running Configuration:

```
performance-measurement
 liveness-profile sr-policy default
  liveness-detection
   multiplier 5
  !
  probe
   tos dscp 52
   tx-interval 150000
  !
 !
!
end
```

Configure a Named (Non-Default) SR-Policy PM Liveness-Profile

The following example shows a named sr-policy liveness-profile:

```
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-profile name sample-profile
Router(config-pm-ld-profile)# probe
Router(config-pm-ld-probe)# tx-interval 150000
Router(config-pm-ld-probe)# tos dscp 52
Router(config-pm-ld-probe)# exit
Router(config-pm-ld-profile)# liveness-detection
Router(config-pm-ld-profile-ld)# multiplier 5
Router(config-pm-ld-profile-ld)#commit
```

Running Configuration:

```
performance-measurement
 liveness-profile name sample-profile
  liveness-detection
   multiplier 5
  !
  probe
   tos dscp 52
   tx-interval 150000
```

```
  !
 !
!
end
```

Configure a SR-Policy PM Liveness-Profile with Sweep Parameters

The following example shows a named sr-policy liveness-profile with sweep parameters:

```
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-profile name sample-profile
Router(config-pm-ld-profile)# probe
Router(config-pm-ld-probe)# tx-interval 150000
Router(config-pm-ld-probe)# tos dscp 52
Router(config-pm-ld-probe)# sweep
Router(config-pm-ld-probe-sweep)# destination ipv4 127.0.0.1 range 25
Router(config-pm-ld-probe-sweep)# exit
Router(config-pm-ld-probe)# exit

Router(config-pm-ld-profile)# liveness-detection
Router(config-pm-ld-profile-ld)# multiplier 5
Router(config-pm-ld-profile-ld)#commit
```

Running Configuration

```
performance-measurement
 liveness-profile name sample-profile
  liveness-detection
   multiplier 5
  !
  probe
   tos dscp 52
   sweep
    destination ipv4 127.0.0.1 range 25
   !
   tx-interval 150000
  !
 !
!
end
```

Enable Liveness Monitoring Under SR Policy

The following example shows how to enable liveness monitoring under SR Policy, associate a liveness-profile, and configure the invalidation action:

```
RP/0/RSP0/CPU0:ios(config)# segment-routing traffic-eng
RP/0/RSP0/CPU0:ios(config-sr-te)# policy FOO
RP/0/RSP0/CPU0:ios(config-sr-te-policy)# performance-measurement
RP/0/RSP0/CPU0:ios(config-sr-te-policy-perf-meas)# liveness-detection
RP/0/RSP0/CPU0:ios(config-sr-te-policy-live-detect)# liveness-profile name sample-profile
RP/0/RSP0/CPU0:ios(config-sr-te-policy-live-detect)# invalidation-action none
```

Running Config

```
segment-routing
 traffic-eng
  policy FOO
   performance-measurement
    liveness-detection
     liveness-profile name sample-profile
     invalidation-action none
    !
   !
  !
 !
```

```
!
end
```

Enable Liveness Monitoring under SR Policy with Optional Parameters

The following example shows how to enable liveness monitoring under SR Policy, associate a liveness-profile, and configure reverse path label and session logging:

```
RP/0/RSP0/CPU0:ios(config)# segment-routing traffic-eng
RP/0/RSP0/CPU0:ios(config-sr-te)# policy BAA
RP/0/RSP0/CPU0:ios(config-sr-te-policy)# performance-measurement
RP/0/RSP0/CPU0:ios(config-sr-te-policy-perf-meas)# liveness-detection
RP/0/RSP0/CPU0:ios(config-sr-te-policy-live-detect)# liveness-profile name sample-profile
RP/0/RSP0/CPU0:ios(config-sr-te-policy-live-detect)# invalidation-action down
RP/0/RSP0/CPU0:ios(config-sr-te-policy-live-detect)# logging session-state-change
RP/0/RSP0/CPU0:ios(config-sr-te-policy-live-detect)# exit
RP/0/RSP0/CPU0:ios(config-sr-te-policy-perf-meas)# reverse-path label 16001
```

Running Config

```
segment-routing
 traffic-eng
  policy BAA
   performance-measurement
    liveness-detection
     logging
      session-state-change
     !
     liveness-profile name sample-profile
     invalidation-action down
    !
    reverse-path
     label 16001
    !
   !
  !
 !
!
end
```

Configure Segment Lists to Activate Candidate Paths in SRv6 for PM Liveness

Table 74: Feature History Table

Feature Name	Release Information	Feature Description
Configure Segment Lists to Activate Candidate Paths in SRv6 for PM Liveness	Release 7.11.1	You can now enable a candidate path to be up by configuring the minimum number of active segment lists associated with the candidate path. The head-end router determines that a candidate path is up based on the minimum number of active segment lists configured. In earlier releases, the head-end router identified a candidate path as up only when all the segment lists associated with the path were active. The feature introduces these changes: **CLI:** • The **validation-cp minimum-active segment-lists** option is introduced in the **performance-measurement liveness-detection** command. **YANG Data Models:** • `Cisco-IOS-XR-infra-xtc-agent-cfg.yang` See (GitHub, Yang Data Models Navigator)

The state of the segment lists in a candidate path determines whether a candidate path is up or down. You can now configure the minimum number of active segment lists associated with a candidate path. The head-end router identifies a candidate path as up when one or more segment lists are active.

Note If the configured minimum number of active segment lists is greater than the number of available segment lists in a candidate path, the head-end router determines the candidate path as up only when all the segment lists are active.

In earlier releases, the router identified a candidate path as up only when all the segment lists associated with the path were active.

Configure the minimum number of segment lists in SRv6

Perform this task to activate three segment lists to have the PM liveness session up:

```
Router(config)#segment-routing
Router(config-sr)#traffic-eng
Router(config-sr-te)#policy po-103
Router(config-sr-te-policy)#performance-measurement
Router(config-sr-te-policy-perf-meas)#liveness-detection
Router(config-sr-te-policy-live-detect)#validation-cp minimum-active segment-lists 3
```

Running Configuration

```
segment-routing
 traffic-eng
  policy po-103
   performance-measurement
    liveness-detection
     validation-cp minimum-active segment-lists 3
    !
   !
  !
 !
!
```

Verification

The following example shows three active segment-lists to have the PM liveness session up:

```
Router#show performance-measurement sr-policy liveness color 103 detail verbose private
Mon Oct 30 15:10:51.863 EDT

------------------------------------------------------------------------------------------------------------------------
0/1/CPU0
------------------------------------------------------------------------------------------------------------------------

SR Policy name: srte_c_103_ep_3::1
  Color                       : 103
  SRv6 Encap Source Address   : 1::1
  Endpoint                    : 3::1
  Handle                      : 0x00000000
  Policy to be deleted        : False
  Number of candidate-paths   : 1

  Candidate-Path:
    Instance                  : 5
    Preference                : 300
    Protocol-origin           : Configured
    Discriminator             : 300
    Profile Keys:
      Profile name            : default
      Profile type            : SR Policy Liveness Detection
    Candidate path to be deleted: False
    Source address            : 1::1
    Local label               : Not set
    Fast notification for session down: Disabled
      No fast notifications have been sent
    Number of segment-lists   : 3
    Liveness Detection: Enabled
      Minumum SL Up Required: 1
      Session State: Up
      Last State Change Timestamp: Oct 30 2023 15:10:16.322
      Missed count: 0

    Segment-List              : sl-1041
      fccc:cc00:1:fe10:: (Local Adjacency SID)
      fccc:cc00:2:fe41::/64
```

```
    Format: f3216
  Segment List ID: 0
  Reverse path segment-List: Not configured
  Segment-list to be deleted: False
  Number of atomic paths  : 1
  Liveness Detection: Enabled
    Session State: Up
    Last State Change Timestamp: Oct 30 2023 15:10:16.322
    Missed count: 0

  Atomic path:
    Flow Label            : 0
    Session ID            : 4198
    Trace ID              : 738913600
    Atomic path to be deleted: False
    NPU Offloaded session : False
    Timestamping Enabled  : True
    Liveness Detection: Enabled
      Session State: Up
      Last State Change Timestamp: Oct 30 2023 15:10:16.322
      Missed count: 0
    Responder IP          : 1::1
    Number of Hops        : 3

Segment-List              : sl-1042
  fccc:cc00:1:fe10:: (Local Adjacency SID)
  fccc:cc00:2:fe42::/64
    Format: f3216
  Segment List ID: 0
  Reverse path segment-List: Not configured
  Segment-list to be deleted: False
  Number of atomic paths  : 1
  Liveness Detection: Enabled
    Session State: Up
    Last State Change Timestamp: Oct 30 2023 15:10:16.322
    Missed count: 0

  Atomic path:
    Flow Label            : 0
    Session ID            : 4199
    Trace ID              : 954039677
    Atomic path to be deleted: False
    NPU Offloaded session : False
    Timestamping Enabled  : True
    Liveness Detection: Enabled
      Session State: Up
      Last State Change Timestamp: Oct 30 2023 15:10:16.322
      Missed count: 0
    Responder IP          : 1::1
    Number of Hops        : 3

Segment-List              : sl-1043
  fccc:cc00:1:fe10:: (Local Adjacency SID)
  fccc:cc00:2:fe43::/64
    Format: f3216
  Segment List ID: 0
  Reverse path segment-List: Not configured
  Segment-list to be deleted: False
  Number of atomic paths  : 1
  Liveness Detection: Enabled
    Session State: Up
    Last State Change Timestamp: Oct 30 2023 15:10:16.322
    Missed count: 0
```

```
Atomic path:
  Flow Label            : 0
  Session ID            : 4200
  Trace ID              : 1119107116
  Atomic path to be deleted: False
  NPU Offloaded session : False
  Timestamping Enabled  : True
  Liveness Detection: Enabled
    Session State: Up
    Last State Change Timestamp: Oct 30 2023 15:10:16.322
    Missed count: 0
  Responder IP          : 1::1
  Number of Hops        : 3

--------------------------------------------------------------------------------

0/RSP0/CPU0

--------------------------------------------------------------------------------
```

Configure Flow Labels in SRv6 Header for PM Liveness

Table 75: Feature History Table

Feature Name	Release Information	Feature Description
Configure Flow Labels in SRv6 Header for PM Liveness	Release 7.11.1	You can now monitor the activeness of multiple paths for a given segment list using flow labels in the SRv6 header. In earlier releases, the SRv6 header didn't include flow labels. The feature introduces these changes: **CLI:** • The **flow-label** keyword is introduced in the **performance-measurement liveness-profile** command. **YANG Data Models:** • ~~Cisco-IOS-XR-unperformance-measurement-cfg.yang~~ • Cisco-IOS-XR-perf-meas-oper.yang See (GitHub, Yang Data Models Navigator)

To monitor the activeness of multiple paths for a given a segment list, you can configure the SRv6 header to include flow labels as the packet travels in the network. When there are multiple paths, different traffic flows may use different paths. A flow label is a flow identifier and you can use different flow labels to monitor different ECMP paths. It's only used for IPv6 probe packets. Flow labels are 20-bit fields in the SRv6 header.

Configure flow labels in the SRv6 header

Perform the following task in the global configuration mode to configure flow labels in the SRv6 header:

```
Router#configure
Router(config)#performance-measurement
Router(config-perf-meas)#liveness-profile name name1
Router(config-pm-ld-profile)#probe flow-label from 0 to 1000000 increment 10
```

Running Configuration

```
performance-measurement
 liveness-profile name name1
  probe
   flow-label from 0 to 1000000 increment 10
  !
 !
!
```

Verification

The following example shows an SR-policy configured with flow labels:

```
Router#show performance-measurement sr-policy liveness color 1001 detail verbose private

Mon Oct 30 15:25:55.241 EDT

------------------------------------------------------------------------------------------------------------------------
0/1/CPU0
------------------------------------------------------------------------------------------------------------------------

SR Policy name: srte_c_1001_ep_3::1
  Color                       : 1001
  SRv6 Encap Source Address   : 1::1
  Endpoint                    : 3::1
  Handle                      : 0x00000000
  Policy to be deleted        : False
  Number of candidate-paths   : 1

  Candidate-Path:
    Instance                  : 3
    Preference                : 300
    Protocol-origin           : Configured
    Discriminator             : 300
    Profile Keys:
      Profile name            : profile-scale
      Profile type            : Generic Liveness Detection
    Candidate path to be deleted: False
    Source address            : 1::1
    Local label               : Not set
    Fast notification for session down: Disabled
      No fast notifications have been sent
    Number of segment-lists   : 2
    Liveness Detection: Enabled
      Minumum SL Up Required: 2
      Session State: Up
      Last State Change Timestamp: Oct 26 2023 15:31:43.478
      Missed count: 0

    Segment-List              : sl-1041
      fccc:cc00:1:fe10:: (Local Adjacency SID)
      fccc:cc00:2:fe41::/64
        Format: f3216
      Segment List ID: 0
      Reverse path segment-List: Not configured
      Segment-list to be deleted: False
      Number of atomic paths  : 2
```

```
  Liveness Detection: Enabled
    Session State: Up
    Last State Change Timestamp: Oct 26 2023 15:31:43.478
    Missed count: 0

  Atomic path:
    Flow Label            : 0
    Session ID            : 4178
    Trace ID              : 280178832
    Atomic path to be deleted: False
    NPU Offloaded session : False
    Timestamping Enabled  : True
    Liveness Detection: Enabled
      Session State: Up
      Last State Change Timestamp: Oct 26 2023 15:31:43.478
      Missed count: 0
    Responder IP          : 1::1
    Number of Hops        : 3

  Atomic path:
    Flow Label            : 10
    Session ID            : 4179
    Trace ID              : 1866227171
    Atomic path to be deleted: False
    NPU Offloaded session : False
    Timestamping Enabled  : True
    Liveness Detection: Enabled
      Session State: Up
      Last State Change Timestamp: Oct 26 2023 15:31:43.478
      Missed count: 0
    Responder IP          : 1::1
    Number of Hops        : 3

Segment-List              : sl-scale
  fccc:cc00:1:fe10:: (Local Adjacency SID)
  fccc:cc00:2:fed1::/64
    Format: f3216
  Segment List ID: 0
  Reverse path segment-List: Not configured
  Segment-list to be deleted: False
  Number of atomic paths  : 2
  Liveness Detection: Enabled
    Session State: Up
    Last State Change Timestamp: Oct 26 2023 15:31:43.478
    Missed count: 0

  Atomic path:
   Flow Label            : 0
    Session ID            : 4180
    Trace ID              : 2609815826
    Atomic path to be deleted: False
    NPU Offloaded session : False
    Timestamping Enabled  : True
    Liveness Detection: Enabled
      Session State: Up
      Last State Change Timestamp: Oct 26 2023 15:31:43.478
      Missed count: 0
    Responder IP          : 1::1
    Number of Hops        : 3

  Atomic path:
    Flow Label            : 10
    Session ID            : 4181
    Trace ID              : 170501506
```

```
Atomic path to be deleted: False
NPU Offloaded session : False
Timestamping Enabled  : True
Liveness Detection: Enabled
  Session State: Up
  Last State Change Timestamp: Oct 26 2023 15:31:43.478
  Missed count: 0
Responder IP          : 1::1
Number of Hops        : 3
```

```
0/RSP0/CPU0
```

Delay Measurement

Delay measuremnt is a mechanism used to measure the latency or delay experienced by data packets when they traverse a network.

The PM for delay measuremnt uses the IP/UDP packet format defined in RFC 5357 (TWAMP-Light) for probes. Two-Way Active Measurement Protocol (TWAMP) adds two-way or round-trip measurement capabilities. TWAMP employs time stamps applied at the echo destination (reflector) to enable greater accuracy. In the case of TWAMP Light, the Session-Reflector doesn't necessarily know about the session state. The Session-Reflector simply copies the Sequence Number of the received packet to the Sequence Number field of the reflected packet. The controller receives the reflected test packets and collects two-way metrics. This architecture allows for collection of two-way metrics.

Benefits

- Network Troubleshooting: You can quickly and easily identify areas in your network with high delay and resolve network problems using delay measurement.
- Network Planning and Optimization: You can easily understand the performance of your network under various conditions and design a network that can handle expected traffic loads.
- Quality of Service (QoS): You can ensure quality of service standards are being met by continuously monitoring the delay in your network.

Supported Delay Measurement Methods

You can measure delay using the following methods:

- Use to monitor delay experienced by data packets in a single link or path between two nodes in a network.
- : Use to monitor the amount of time it takes for a data packet to travel from a source device to a specific IP endpoint within a network.
- : Use to to monitor the end-to-end delay experienced by the traffic sent over an SR policy.

Measurement Modes

The following table compares the different hardware and timing requirements for the measurement modes supported in SR PM.

Feature Name	Release	Description
SR Performance Measurement: Loopback Measurement Mode	Release 7.5.2	Loopback measurement mode provides two-way and one-way measurements. PTP-capable hardware and hardware timestamping are required on the Sender but are not required on the Reflector. Liveness monitoring uses "self-addressed" PM IP packets crafted by the sender (where the destination address is the sender's own IP address); this mode of operation is referred as "loopback mode".

Table 76: Measurement Mode Requirements

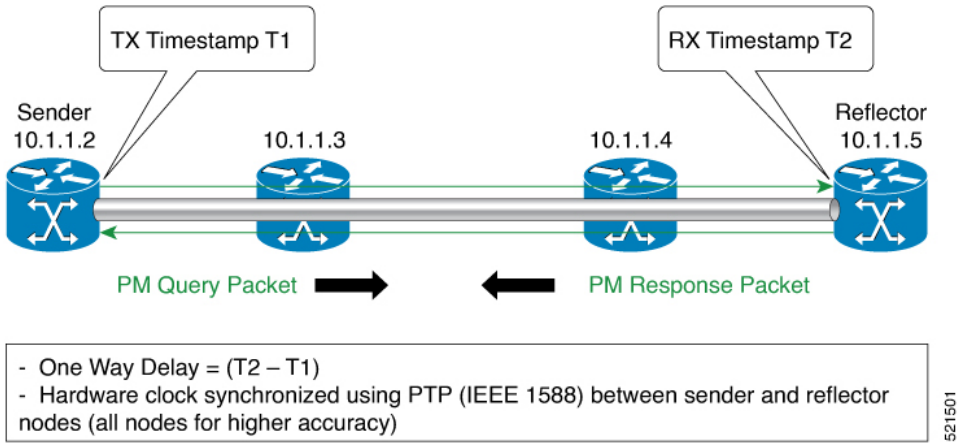
Measurement Mode	Sender: PTP-Capable HW and HW Timestamping	Reflector: PTP-Capable HW and HW Timestamping	PTP Clock Synchronization between Sender and Reflector
One-way	Required	Required	Required
Two-way	Required	Required	Not Required
Loopback	Required	Not Required	Not Required

One-Way Measurement Mode

One-way measurement mode provides the most precise form of one-way delay measurement. PTP-capable hardware and hardware timestamping are required on both Sender and Reflector, with PTP Clock Synchronization between Sender and Reflector.

Delay measurement in one-way mode is calculated as (T2 – T1).

Figure 49: One-Way



The PM query and response for one-way delay measurement can be described in the following steps:

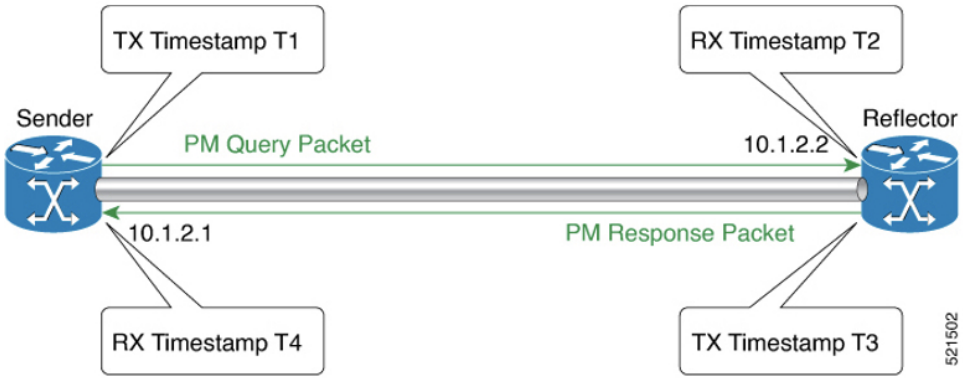
1. The local-end router sends PM query packets periodically to the remote side once the egress line card on the router applies timestamps on packets.
2. The ingress line card on the remote-end router applies time-stamps on packets as soon as they are received.
3. The remote-end router sends the PM packets containing time-stamps back to the local-end router.
4. One-way delay is measured using the time-stamp values in the PM packet.

Two-Way Measurement Mode

Two-way meaurement mode provides two-way measurements. PTP-capable hardware and hardware timestamping are required on both Sender and Reflector, but PTP clock synchronization between Sender and Reflector is not required.

Delay measurement in two-way mode is calculated as ((T4 – T1) – (T3 – T2))/2.

Figure 50: Two-Way



The PM query and response for two-way delay measurement can be described in the following steps:

1. The local-end router sends PM query packets periodically to the remote side once the egress line card on the router applies timestamps on packets.

2. Ingress line card on the remote-end router applies time-stamps on packets as soon as they are received.
3. The remote-end router sends the PM packets containing time-stamps back to the local-end router. The remote-end router time-stamps the packet just before sending it for two-way measurement.
4. The local-end router time-stamps the packet as soon as the packet is received for two-way measurement.
5. Delay is measured using the time-stamp values in the PM packet.

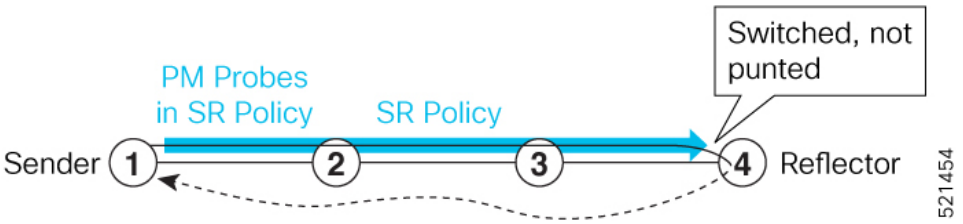
Loopback Measurement Mode

Loopback meaurement mode provides two-way and one-way measurements. PTP-capable hardware and hardware timestamping are required on the Sender, but are not required on the Reflector.

Delay measurements in Loopback mode are calculated as follows:

- Round-Trip Delay = (T4 – T1)
- One-Way Delay = Round-Trip Delay/2

Figure 51: Loopback



The PM query and response for Loopback delay measurement can be described in the following steps:

1. The local-end router sends PM probe packets periodically on the SR Policy.
2. The probe packets are loopback on the endpoint node (not punted), with no timestamping on endpoint node.
3. Round-trip Delay = T4 – T1.

Link Delay Measurement

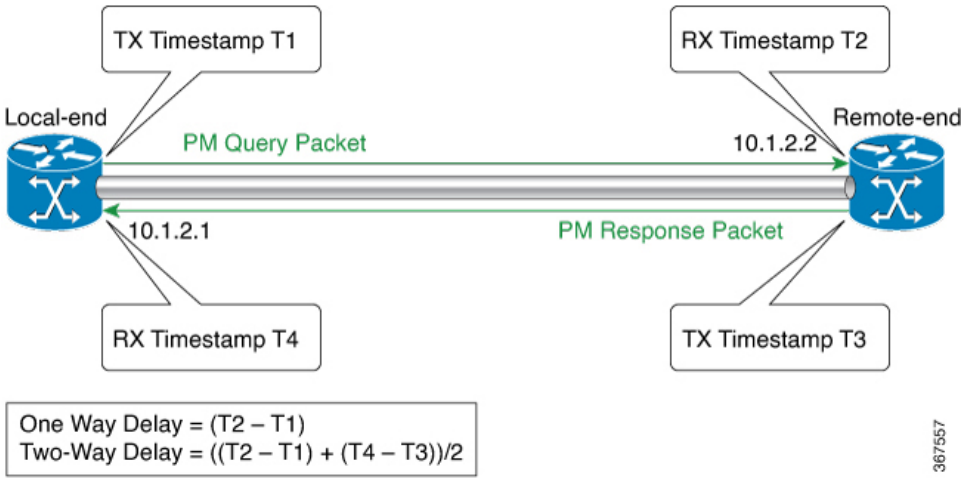
Table 77: Feature History Table

Feature Name	Release Information	Feature Description
Link Delay Measurement using TWAMP Light Encoding	Release 7.3.1	The PM for link delay uses the IP/UDP packet format defined in RFC 5357 (TWAMP-Light) for probes. Two-Way Active Measurement Protocol (TWAMP) adds two-way or round-trip measurement capabilities. TWAMP employs time stamps applied at the echo destination (reflector) to enable greater accuracy.

The PM for link delay uses the IP/UDP packet format defined in RFC 5357 (TWAMP-Light) for probes. Two-Way Active Measurement Protocol (TWAMP) adds two-way or round-trip measurement capabilities. TWAMP employs time stamps applied at the echo destination (reflector) to enable greater accuracy. In the case of TWAMP Light, the Session-Reflector doesn't necessarily know about the session state. The Session-Reflector simply copies the Sequence Number of the received packet to the Sequence Number field of the reflected packet. The controller receives the reflected test packets and collects two-way metrics. This architecture allows for collection of two-way metrics.

The following figure explains the PM query and response for link delay.

Figure 52: Performance Measurement for Link Delay



The PM query and response for link delay can be described in the following steps:

1. The local-end router sends PM query packets periodically to the remote side once the egress line card on the router applies timestamps on packets.
2. Ingress line card on the remote-end router applies time-stamps on packets as soon as they are received.
3. The remote-end router sends the PM packets containing time-stamps back to the local-end router. The remote-end router time-stamps the packet just before sending it for two-way measurement.

4. The local-end router time-stamps the packet as soon as the packet is received for two-way measurement.
5. One-way delay and optionally two-way delay is measured using the time-stamp values in the PM packet.

Restrictions and Usage Guidelines for PM for Link Delay

The following restrictions and guidelines apply for the PM for link delay feature for different links.

- For LSPs, remote-end line card needs to be MPLS and multicast MAC address capable.
- For broadcast links, only point-to-point (P2P) links are supported. P2P configuration on IGP is required for flooding the value.
- For link bundles, the hashing function may select a member link for forwarding but the reply may come from the remote line card on a different member link of the bundle.
- For one-way delay measurement, clocks should be synchronized on two end-point nodes of the link using PTP.
- Link delay measurement is supported on IPv4 unnumbered interfaces. An IPv4 unnumbered interface is identified by a node ID (a loopback address) and the local SNMP index assigned to the interface. Note that the reply messages could be received on any interface, since the packets are routed at the responder based on the loopback address used to identify the link.

Configuration Example: PM for Link Delay

This example shows how to configure performance-measurement functionalities for link delay as a global default profile. The default values for the different parameters in the PM for link delay is given as follows:

- **probe measurement mode**: The default measurement mode for probe is two-way delay measurement. If you are configuring one-way delay measurement, hardware clocks must be synchronized between the local-end and remote-end routers using precision time protocol (PTP).
- **protocol**: Interface delay measurement uses TWAMP-Light (RFC5357) with IP/UDP encap.
- **tx-interval**: Interval for sending probe packet. The default value is 3000000 microseconds and the range is from 3300 to 15000000 microseconds.
- **computation interval**: Interval for metric computation. Default is 30 seconds; range is 1 to 3600 seconds.
- **periodic advertisement**: Periodic advertisement is enabled by default.
- **periodic-advertisement interval**: The default value is 120 seconds and the interval range is from 30 to 3600 seconds.
- **periodic-advertisement threshold**: The default value of periodic advertisement threshold is 10 percent.
- **periodic-advertisement minimum change**: The default value is 1000 microseconds (usec) and the range is from 0 to 10000 microseconds.
- **accelerated advertisement**: Accelerated advertisement is disabled by default.
- **accelerated-advertisement threshold**: The default value is 20 percent and the range is from 0 to 100 percent.
- **accelerated-advertisement minimum change**: The default value is 1000 microseconds and the range is from 1 to 100000 microseconds.

```
RP/0/0/CPU0:router(config)# performance-measurement delay-profile interfacesdefault
RP/0/0/CPU0:router(config-pm-dm-intf)# probe
RP/0/0/CPU0:router(config-pm-dm-intf-probe)# measurement-mode one-way
RP/0/0/CPU0:router(config-pm-dm-intf-probe)# tx-interval 30000
RP/0/0/CPU0:router(config-pm-dm-intf-probe)# exit

RP/0/0/CPU0:router(config-pm-dm-intf)# advertisement periodic
RP/0/0/CPU0:router(config-pm-dm-intf-adv-per)# interval 120
RP/0/0/CPU0:router(config-pm-dm-intf-adv-per)# threshold 20
RP/0/0/CPU0:router(config-pm-dm-intf-adv-per)# minimum-change 1000
RP/0/0/CPU0:router(config-pm-dm-intf-adv-per)# exit

RP/0/0/CPU0:router(config-pm-dm-intf)# advertisement accelerated
RP/0/0/CPU0:router(config-pm-dm-intf-adv-acc)# threshold 30
RP/0/0/CPU0:router(config-pm-dm-intf-adv-acc)# minimum-change 1000
RP/0/0/CPU0:router(config-pm-dm-intf-adv-per)# exit
```

Configure the UDP Destination Port

Configuring the UDP port for TWAMP-Light protocol is optional. By default, PM uses port 862 as the TWAMP-reserved UDP destination port.

The UDP port is configured for each PM measurement probe type (delay, loss, protocol, authentication mode, etc.) on querier and responder nodes. If you configure a different UDP port, the UDP port for each PM measurement probe type must match on the querier and the responder nodes.

Note The same UDP destination port is used for delay measurement for links and SR Policy.

This example shows how to configure the UDP destination port.

```
Router(config)# performance-measurement
Router(config-perf-meas)# protocol twamp-light
Router(config-pm-protocol)# measurement delay unauthenticated
Router(config-pm-proto-mode)# querier-dst-port 12000
```

Enable PM for Link Delay Over an Interface

This example shows how to enable PM for link delay over an interface.

```
RP/0/0/CPU0:router(config)# performance-measurement
RP/0/0/CPU0:router(config-perf-meas)# interface TenGigE0/0/0/0
RP/0/0/CPU0:router(config-pm-intf)# delay-measurement
RP/0/0/CPU0:router(config-pm-dm-intf)# exit
```

Verification

```
RP/0/0/CPU0:router# show performance-measurement profile default interface
Thu Dec 12 14:13:16.029 PST

------------------------------------------------------------------------------
0/0/CPU0
------------------------------------------------------------------------------
Interface Delay-Measurement:
```

```
  Profile configuration:
    Measurement Type                          : Two-Way
    Probe computation interval                : 30 (effective: 30) seconds
    Type of services                          : Traffic Class: 6, DSCP: 48
    Burst interval                            : 3000 (effective: 3000) mSec
    Burst count                               : 10 packets
    Encap mode                                : UDP
    Payload Type                              : TWAMP-light
    Destination sweeping mode                 : Disabled
    Periodic advertisement                    : Enabled
      Interval                                : 120 (effective: 120) sec
      Threshold                               : 10%
      Minimum-Change                          : 500 uSec
    Advertisement accelerated                 : Disabled
    Threshold crossing check                  : Minimum-delay

RP/0/0/CPU0:router# show performance-measurement summary detail location 0/2/CPU0

Thu Dec 12 14:09:59.162 PST

------------------------------------------------------------------------------
0/2/CPU0
------------------------------------------------------------------------------
Total interfaces                              : 1
Total SR Policies                             : 0
Total RSVP-TE tunnels                         : 0
Total Maximum PPS                             : 2000 pkts/sec
Total Interfaces PPS                          : 0 pkts/sec
Maximum Allowed Multi-hop PPS                 : 2000 pkts/sec
Multi Hop Requested PPS                       : 0 pkts/sec (0% of max allowed)
Dampened Multi Hop Requested PPS              : 0% of max allowed
Inuse Burst Interval Adjustment Factor        : 100% of configuration


Interface Delay-Measurement:
  Total active sessions                       : 1
  Counters:
    Packets:
      Total sent                              : 26
      Total received                          : 26
    Errors:
        TX:
          Reason interface down               : 0
          Reason no MPLS caps                 : 0
          Reason no IP address                : 0
          Reason other                        : 0
        RX:
          Reason negative delay               : 0
          Reason delay threshold exceeded     : 0
          Reason missing TX timestamp         : 0
          Reason missing RX timestamp         : 0
          Reason probe full                   : 0
          Reason probe not started            : 0
          Reason control code error           : 0
          Reason control code notif           : 0
    Probes:
      Total started                           : 3
      Total completed                         : 2
      Total incomplete                        : 0
      Total advertisements                    : 0

SR Policy Delay-Measurement:
  Total active sessions                       : 0
  Counters:
```

```
    Packets:
      Total sent                            : 0
      Total received                        : 0
    Errors:
        TX:
          Reason interface down             : 0
          Reason no MPLS caps               : 0
          Reason no IP address              : 0
          Reason other                      : 0
        RX:
          Reason negative delay             : 0
          Reason delay threshold exceeded   : 0
          Reason missing TX timestamp       : 0
          Reason missing RX timestamp       : 0
          Reason probe full                 : 0
          Reason probe not started          : 0
          Reason control code error         : 0
          Reason control code notif         : 0
    Probes:
      Total started                         : 0
      Total completed                       : 0
      Total incomplete                      : 0
      Total advertisements                  : 0

RSVP-TE Delay-Measurement:
  Total active sessions                     : 0
  Counters:
    Packets:
      Total sent                            : 0
      Total received                        : 0
    Errors:
        TX:
          Reason interface down             : 0
          Reason no MPLS caps               : 0
          Reason no IP address              : 0
          Reason other                      : 0
        RX:
          Reason negative delay             : 0
          Reason delay threshold exceeded   : 0
          Reason missing TX timestamp       : 0
          Reason missing RX timestamp       : 0
          Reason probe full                 : 0
          Reason probe not started          : 0
          Reason control code error         : 0
          Reason control code notif         : 0
    Probes:
      Total started                         : 0
      Total completed                       : 0
      Total incomplete                      : 0
      Total advertisements                  : 0

Global Delay Counters:
  Total packets sent                        : 26
  Total query packets received              : 26
  Total invalid session id                  : 0
  Total missing session                     : 0

RP/0/0/CPU0:router# show performance-measurement interfaces detail
Thu Dec 12 14:16:09.692 PST

------------------------------------------------------------------------------
0/0/CPU0
------------------------------------------------------------------------------

------------------------------------------------------------------------------
```

```
0/2/CPU0
------------------------------------------------------------------------------

Interface Name: GigabitEthernet0/2/0/0 (ifh: 0x1004060)
  Delay-Measurement                 : Enabled
  Loss-Measurement                  : Disabled
  Configured IPv4 Address           : 10.10.10.2
  Configured IPv6 Address           : 10:10:10::2
  Link Local IPv6 Address           : fe80::3a:6fff:fec9:cd6b
  Configured Next-hop Address       : Unknown
  Local MAC Address                 : 023a.6fc9.cd6b
  Next-hop MAC Address              : 0291.e460.6707
  Primary VLAN Tag                  : None
  Secondary VLAN Tag                : None
  State                             : Up

  Delay Measurement session:
    Session ID          : 1

    Last advertisement:
      Advertised at: Dec 12 2019 14:10:43.138 (326.782 seconds ago)
      Advertised reason: First advertisement
      Advertised delays (uSec): avg: 839, min: 587, max: 8209, variance: 297

    Next advertisement:
      Threshold check scheduled in 1 more probe (roughly every 120 seconds)
      Aggregated delays (uSec): avg: 751, min: 589, max: 905, variance: 112
      Rolling average (uSec): 756

    Current Probe:
      Started at Dec 12 2019 14:15:43.154 (26.766 seconds ago)
      Packets sent: 9, received: 9, lost: 0
      Measured delays (uSec): avg: 795, min: 631, max: 1199, variance: 164
      Next probe scheduled at Dec 12 2019 14:16:13.132 (in 3.212 seconds)
      Next burst packet will be sent in 0.212 seconds
      Burst packet sent every 3.0 seconds
      Probe samples:
        Packet Rx Timestamp       Measured Delay (nsec)
        Dec 12 2019 14:15:43.156          689223
        Dec 12 2019 14:15:46.156          876561
        Dec 12 2019 14:15:49.156          913548
        Dec 12 2019 14:15:52.157         1199620
        Dec 12 2019 14:15:55.156          794008
        Dec 12 2019 14:15:58.156          631437
        Dec 12 2019 14:16:01.157          656440
        Dec 12 2019 14:16:04.157          658267
        Dec 12 2019 14:16:07.157          736880
```

You can also use the following commands for verifying the PM for link delay on the local-end router.

Command	Description
show performance-measurement history probe interfaces [*interface*]	Displays the PM link-delay probe history for interfaces.
show performance-measurement history aggregated interfaces [*interface*]	Displays the PM link-delay aggregated history for interfaces.
show performance-measurement history advertisement interfaces [*interface*]	Displays the PM link-delay advertisement history for interfaces.

Command	Description
show performance-measurement counters [interface *interface*] [location *location-name*]	Displays the PM link-delay session counters.

You can also use the following commands for verifying the PM for link-delay configuration on the remote-end router.

Command	Description
show performance-measurement responder summary [**location** *location-name*]	Displays the PM for link-delay summary on the remote-end router (responder).
show performance-measurement responder interfaces [*interface*]	Displays PM for link-delay for interfaces on the remote-end router.
show performance-measurement responder counters [**interface** *interface*] [**location** *location-name*]	Displays the PM link-delay session counters on the remote-end router.

SR Performance Measurement Named Profiles

Feature	Release Information	Feature Description
SR Performance Measurement Named Profiles	Release 7.5.2	You can use this feature to create specific performance measurement delay and liveness profiles, and associate it with an SR policy. You can use the delay or liveness profile to be associated with an interface or network area, where the performance management probes are enabled, and performance measurement is precise and enhanced.

You can create a named performance measurement profile for delay or liveness.

Delay Profile

This example shows how to create a named SR performance measurement delay profile.

```
Router(config)# performance-measurement delay-profile name profile2
Router(config-pm-dm-profile)# probe
Router(config-pm-dm-probe)# tx-interval 60000
Router(config-pm-dm-probe)# computation-interval 60
Router(config-pm-dm-probe)# protocol twamp-light
Router(config-pm-dm-probe)# tos dscp 63
Router(config-pm-dm-probe)# exit

Router(config-pm-dm-profile)# advertisement
Router(config-pm-dm-adv)# periodic
Router(config-pm-dm-adv-per)# interval 60
Router(config-pm-dm-adv-per)# minimum-change 1000
```

```
Router(config-pm-dm-adv-per)# threshold 20
Router(config-pm-dm-adv-per)# commit
```

Apply the delay profile for an SR Policy.

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# policy TEST
Router(config-sr-te-policy)# color 4 end-point ipv4 10.10.10.10
Router(config-sr-te-policy)# performance-measurement
Router(config-sr-te-policy-perf-meas)# delay-measurement delay-profile name profile2

Router(config-sr-te-policy)#candidate-paths
Router(config-sr-te-policy-path)#preference 100
Router(config-sr-te-policy-path-pref)#explicit segment-list LIST1
Router(config-sr-te-pp-info)#weight 2

Router(config-sr-te-policy-path-pref)#explicit segment-list LIST2
Router(config-sr-te-pp-info)#weight 3
```

Running Configuration

```
Router# show run segment-routing traffic-eng policy TEST

segment-routing
 traffic-eng
  policy TEST
   color 4 end-point ipv4 10.10.10.10
   candidate-paths
    preference 100
     explicit segment-list LIST1
      weight 2
     !
     explicit segment-list LIST2
      weight 3
     !
    !
   !
   performance-measurement
    delay-measurement
     delay-profile name profile2
```

Verification

```
Router# show performance-measurement profile named-profile delay

----------------------------
0/RSP0/CPU0
----------------------------
SR Policy Delay Measurement Profile Name: profile2
  Profile configuration:
    Measurement mode                          : One-way
    Protocol type                             : TWAMP-light
    Encap mode                                : UDP
    Type of service:
      PM-MPLS traffic class                   : 6
      TWAMP-light DSCP                        : 63
    Probe computation interval                : 60 (effective: 60) seconds
    Burst interval                            : 60 (effective: 60) mSec
    Packets per computation interval          : 1000
    Periodic advertisement                    : Enabled
      Interval                                : 60 (effective: 60) sec
      Threshold                               : 20%
      Minimum-change                          : 1000 uSec
    Advertisement accelerated                 : Disabled
    Advertisement logging:
      Delay exceeded                          : Disabled (default)
```

```
    Threshold crossing check                    : Maximum-delay
    Router alert                                : Disabled (default)
    Destination sweeping mode                   : Disabled
    Liveness detection parameters:
      Multiplier                                : 3
      Logging state change                      : Disabled
```

On-Demand SR Policy

```
Router(config-sr-te)# on-demand color 20
Router(config-sr-te-color)# performance-measurement delay-measurement
Router(config-sr-te-color-delay-meas)# delay-profile name profile2
Router(config-sr-te-color-delay-meas)# commit
```

Running Configuration

```
Router# show run segment-routing traffic-eng on-demand color 20

segment-routing
 traffic-eng
  on-demand color 20
   performance-measurement
    delay-measurement
     delay-profile name profile2
```

Liveness Profile

This example shows how to create a *named* SR performance measurement liveness profile.

```
Router(config)# performance-measurement liveness-profile name profile3
Router(config-pm-ld-profile)# probe
Router(config-pm-ld-probe)# tx-interval 60000
Router(config-pm-ld-profile)# probe
Router(config-pm-ld-probe)# tx-interval 60000
Router(config-pm-ld-probe)# tos dscp 10
Router(config-pm-ld-probe)# exit

Router(config-pm-ld-profile)# liveness-detection
Router(config-pm-ld-profile-ld)# multiplier 5
Router(config-pm-ld-profile-ld)# commit
```

Apply the liveness profile for the SR policy

This example shows how to enable PM for SR policy liveness for a specific policy.

For the same policy, you cannot enable delay-measurement (delay-profile) and liveness-detection (liveness-profile) at the same time. For example, if delay measurement is enabled, use the **no delay-measurement** command to disable it, and then enable the following command for enabling liveness detection.

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# policy TRST2
Router(config-sr-te-policy)# color 40 end-point ipv4 20.20.20.20
Router(config-sr-te-policy)#candidate-paths
Router(config-sr-te-policy-path)#preference 50
Router(config-sr-te-policy-path-pref)#explicit segment-list LIST3
Router(config-sr-te-pp-info)#weight 2

Router(config-sr-te-policy-path-pref)#explicit segment-list LIST4
Router(config-sr-te-pp-info)#weight 3

Router(config-sr-te-policy)# performance-measurement
Router(config-sr-te-policy-perf-meas)# liveness-detection liveness-profile name profile3
```

Running Configuration

```
Router# show run segment-routing traffic-eng policy TRST2

segment-routing
 traffic-eng
  policy TRST2
   color 40 end-point ipv4 20.20.20.20
   candidate-paths
    preference 50
     explicit segment-list LIST3
      weight 2
     !
     explicit segment-list LIST4
      weight 3
     !
    !
   !
   performance-measurement
    liveness-detection
     liveness-profile name profile3
    !
```

Verification

```
Router# show performance-measurement profile named-profile delay sr-policy

-----------
0/RSP0/CPU0
-----------

SR Policy Liveness Detection Profile Name: profile1
  Profile configuration:
    Measurement mode                          : Loopback
    Protocol type                             : TWAMP-light
    Type of service:
      TWAMP-light DSCP                        : 10
    Burst interval                            : 60 (effective: 60) mSec
    Destination sweeping mode                 : Disabled
    Liveness detection parameters:
      Multiplier                              : 3
      Logging state change                    : Disabled

SR Policy Liveness Detection Profile Name: profile3
  Profile configuration:
    Measurement mode                          : Loopback
    Protocol type                             : TWAMP-light
    Type of service:
      TWAMP-light DSCP                        : 10
    Burst interval                            : 60 (effective: 60) mSec
    Destination sweeping mode                 : Disabled
    Liveness detection parameters:
      Multiplier                              : 3
      Logging state change                    : Disabled
```

On-Demand SR Policy

For the same policy, you cannot enable delay-measurement (delay-profile) and liveness-detection (liveness-profile) at the same time. For example, to disable delay measurement, use the **no delay-measurement** command, and then enable the following command for enabling liveness detection.

```
Router(config-sr-te)#on-demand color 30
Router(config-sr-te-color)#performance-measurement
Router(config-sr-te-color-pm)# liveness-detection liveness-profile name profile1
Router(config-sr-te-color-delay-meas)# commit
```

Running Configuration

```
Router# show run segment-routing traffic-eng on-demand color 30

segment-routing
 traffic-eng
  on-demand color 30
   performance-measurement
    liveness-detection
     liveness-profile name profile1
    !
```

Verification

```
Router# show performance-measurement profile named-profile liveness

--------------------
0/RSP0/CPU0
--------------------
SR Policy Liveness Detection Profile Name: profile1
  Profile configuration:
    Measurement mode                          : Loopback
    Protocol type                             : TWAMP-light
    Type of service:
      TWAMP-light DSCP                        : 10
    Burst interval                            : 60 (effective: 60) mSec
    Destination sweeping mode                 : Disabled
    Liveness detection parameters:
      Multiplier                              : 3
      Logging state change                    : Disabled
```

Delay Normalization

Performance measurement (PM) measures various link characteristics like packet loss and delay. Such characteristics can be used by IS-IS as a metric for Flexible Algorithm computation. Low latency routing using dynamic delay measurement is one of the primary use cases for Flexible Algorithm technology.

Delay is measured in microseconds. If delay values are taken as measured and used as link metrics during the IS-IS topology computation, some valid ECMP paths might be unused because of the negligible difference in the link delay.

The Delay Normalization feature computes a normalized delay value and uses the normalized value instead. This value is advertised and used as a metric during the Flexible Algorithm computation.

The normalization is performed when the delay is received from the delay measurement component. When the next value is received, it is normalized and compared to the previous saved normalized value. If the values are different, then the LSP generation is triggered.

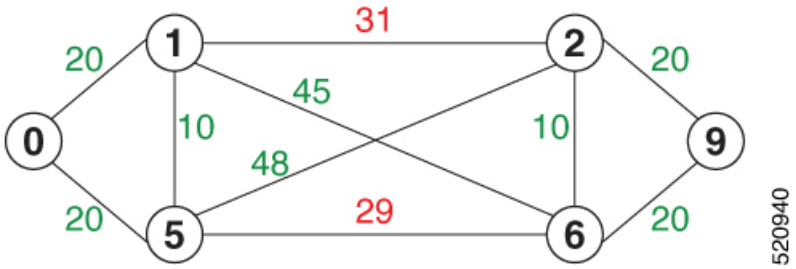
The following formula is used to calculate the normalized value:

- **Dm** – measured Delay
- **Int** – configured normalized Interval
- **Off** – configured normalized Offset (must be less than the normalized interval Int)
- **Dn** – normalized Delay
- **a** = Dm / Int (rounded down)
- **b** = a * Int + Off

If the measured delay (Dm) is less than or equal to **b**, then the normalized delay (Dn) is equal to **b**. Otherwise, Dn is **b** + **Int**.

Example

The following example shows a low-latency service. The intent is to avoid high-latency links (1-6, 5-2). Links 1-2 and 5-6 are both low-latency links. The measured latency is not equal, but the difference is insignificant.



We can normalize the measured latency before it is advertised and used by IS-IS. Consider a scenario with the following:

- Interval = 10
- Offset = 3

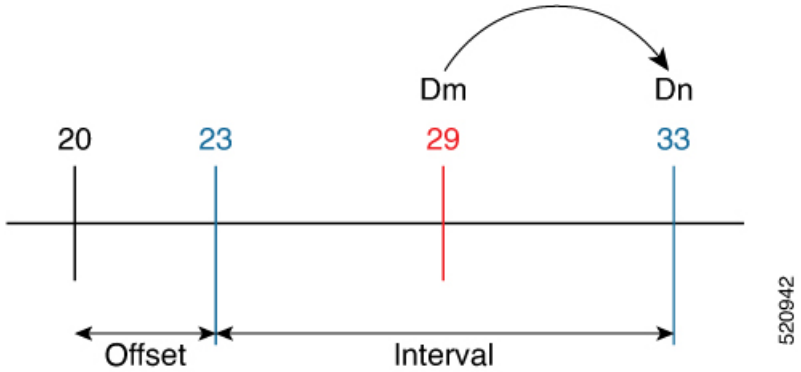
The measured delays will be normalized as follows:

- **Dm** = 29

  a = 29 / 10 = 2 (2.9, rounded down to 2)

  b = 2 * 10 + 3 = 23

  In this case, **Dm** (29) is greater than **b** (23); so **Dn** is equal to **b**+**I** (23 + 10) = **33**

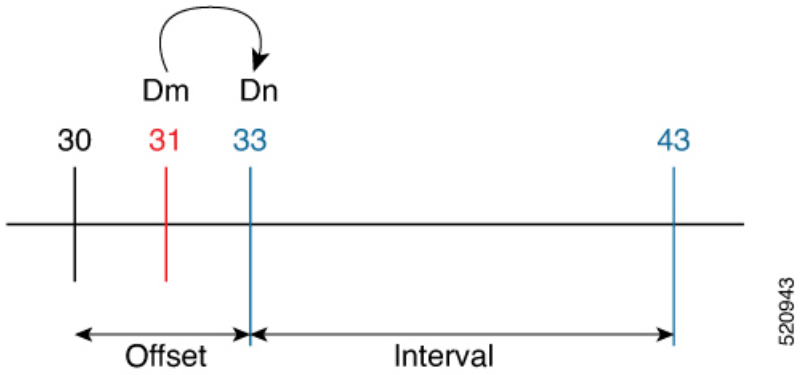


- **Dm** = 31

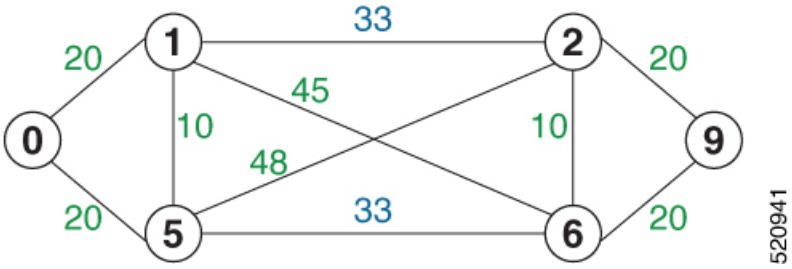
  a = 31 / 10 = 3 (3.1, rounded down to 3)

  b = 3 * 10 + 3 = 33

  In this case, **Dm** (31) is less than **b** (33); so **Dn** is **b** = **33**



The link delay between 1-2 and 5-6 is normalized to 33.



Configuration

Delay normalization is disabled by default. To enable and configure delay normalization, use the **delay normalize interval** *interval* [**offset** *offset*] command.

- *interval* – The value of the normalize interval in microseconds.
- *offset* – The value of the normalized offset in microseconds. This valude must be smaller than the value of normalized interval.

IS-IS Configuration

```
router isis 1
 interface GigEth 0/0/0/0
  delay normalize interval 10 offset 3
  address-family ipv4 unicast
   metric 77
```

OSPF Configuration

```
router ospf 1
 area 0
  interface GigabitEthernet0/0/0/0
   delay normalize interval 10 offset 3
  !
 !
!
```

Link Anomaly Detection with IGP Penalty

Table 78: Feature History Table

Feature Name	Release Information	Feature Description
Link Anomaly Detection with IGP Penalty	Release 7.4.1	This feature allows you to define thresholds above the measured delay that is considered "anomalous" or unusual. When this threshold is exceeded, an anomaly (A) bit/flag is set along with link delay attribute that is sent to clients.

Customers might experience performance degradation issues, such as increased latency or packet loss on a link. Degraded links might be difficult to troubleshoot and can affect applications, especially in cases where traffic is sent over multiple ECMP paths where one of those paths is degraded.

The Anomaly Detection feature allows you to define a delay anomaly threshold to identify unacceptable link delays. Nodes monitor link performance using link delay monitoring probes. The measured value is compared against the delay anomaly threshold values. When the upper bound threshold is exceeded, the link is declared "abnormal", and performance measurement sets an anomaly bit (A-bit). When IGP receives the A-bit, IGP can automatically increase the IGP metric of the link by a user-defined amount to make this link undesirable or unusable. When the link recovers (lower bound threshold), PM resets the A-bit.

Usage Guidelines and Limitations

This feature is not active when narrow metrics are configured because the performance measurement advertisement requires the "wide" metric type length values.

Configuration Example

The following example shows how to configure the upper and lower anomoly thresholds. The range for *upper_bound* and *lower_bound* is from 1 to 200,000 microseconds. The *lower_bound* value must be less than the *upper_bound* value.

```
RP/0/0/CPU0:router(config)# performance-measurement delay-profile interfaces default
RP/0/0/CPU0:router(config-pm-dm-intf)# advertisement
RP/0/0/CPU0:router(config-pm-dm-intf-adv)# anomaly-check upper-bound 5000 lower-bound 1000
RP/0/0/CPU0:router(config-pm-dm-intf-adv)# commit
```

Running Configuration

```
performance-measurement
 delay-profile interfaces default
  advertisement
   anomaly-check
    upper-bound 5000 lower-bound 1000
   !
  !
 !
!
end
```

Delay Measurement for IP Endpoint

Table 79: Feature History Table

Feature Name	Release Information	Feature Description
Delay Measurement for IP Endpoint over SRv6 Network	Release 24.2.11	In Segment Routing over an IPv6 network (SRv6), you can measure packet delay from the source to a specific IP endpoint. You can use this information for troubleshooting, network maintenance, and optimizing network performance. Additionally, you can use flow labels to verify the delay of each subsequent hop path towards the IP endpoint of that path. So that, when network traffic is distributed across multiple available paths towards an IP endpoint, delay measurement tracks the delay of each of these paths towards the IP endpoint. The feature introduces these changes: **CLI:** • The **source-address ipv6** keyword is introduced in the **performance-measurement endpoint** command. • The **segment-list name** keyword is introduced in the **segment-routing traffic-eng explicit** command. • The **flow-label** keyword is introduced in the **performance-measurement delay-profile name** command. **YANG Data Model:** • Cisco-IOS-XR-um-performance-measurement-cfg • Cisco-IOS-XR-perf-meas-oper.yang (See GitHub, YANG Data Models Navigator)

Feature Name	Release Information	Feature Description
IP Endpoint Delay Measurement Monitoring	Release 7.4.1	This feature measures the end-to-end delay and monitors liveness of a specified IP endpoint node, including VRF-aware (awareness of multiple customers belonging to different VRFs). This feature is supported on IPv4, IPv6, and MPLS data planes.

Delay for an IP endpoint is the amount of time it takes for a data packet to travel from a source device to a specific IP endpoint within a network.

To measure a delay for a packet, also called a probe, is sent from a source device to the target IP endpoint.

The time from when the packet leaves the source to when it arrives at the endpoint is measured and recorded as the delay.

You can measure one-way delay, Two-way delay, and Roundtrip delay or delay in loop-back mode. For more information on Delay measurement, see Link Delay Measurement and Measurement Modes.

Collecting IP Endpoint Probe Statistics

- Statistics associated with the probe for delay metrics are available via Histogram and Streaming Telemetry.
- Model Driven Telemetry (MDT) is supported for the following data:
  - Summary, endpoint, session, and counter show command bags.
  - History buffers data
- Model Driven Telemetry (MDT) and Event Driven Telemetry (EDT) are supported for the following data:
  - Delay metrics computed in the last probe computation-interval (event: probe-completed)
  - Delay metrics computed in the last aggregation-interval; that is, end of the periodic advertisement-interval (event: advertisement-interval expired)
  - Delay metrics last notified (event: notification-triggered)
- The following xpaths for MDT/EDT is supported:
  - `Cisco-IOS-XR-perf-meas-oper:performance-measurement/nodes/node/endpoints/endpoint-delay/endpoint-last-probes`
  - `Cisco-IOS-XR-perf-meas-oper:performance-measurement/nodes/node/endpoints/endpoint-delay/endpoint-last-aggregations`
  - `Cisco-IOS-XR-perf-meas-oper:performance-measurement/nodes/node/endpoints/endpoint-delay/endpoint-last-advertisements`

Supported Features

- IPv6 Endpoint Delay in Default VRF (over SRv6)
- SRv6 Endpoint Delay in Default VRF (Endpoint can be Node SID, Flex-Algo SID, Packed uSID carrier)
- IPv6 Endpoint Delay in VRF (static uDT6)
- IPv6 Endpoint Delay in VRF (dynamic uDT6 encap)
- IPv4 Endpoint Delay in VRF or GRT (static uDT4)
- IPv4 Endpoint Delay in VRF or GRT (dynamic uDT4 encap)

Guidelines and Limitations

You can specify a custom labeled path through one or more user-configured segment-lists. User-configured segment-list represents the forwarding path from sender to reflector when the probe is configured in delay-measurement mode.

- SR PM is supported on hardware that supports Precision Time Protocol (PTP). This requirement applies to both one-way and two-way delay measurement.

  See the "**Configuring Precision Time Protocol**" chapter in the *System Management Configuration Guide for Cisco 8000 Series Routers* for Restrictions for PTP and the Timing Hardware Support Matrix.

- Examples of the custom segment-list include:
  - Probe in delay-measurement mode with a segment-list that includes Flex-Algo prefix SID of the endpoint
  - Probe in delay-measurement mode with a segment-list that includes a SID-list with labels to reach the endpoint or the sender (forward direction)
  - Probe in delay-measurement mode with a segment-list that includes BSID associated with SR policy to reach the end point.
- Endpoint segment list configuration is not supported under nondefault VRF.
- Liveness session without segment list for an endpoint in a non-default VRF is not supported.
- SR Performance Measurement endpoint session over BVI interface is not supported.

IP Endpoint Liveness Detection in an SR MPLS Network

IP endpoint liveness detection leverages the loopback measurement-mode. The following workflow describes the sequence of events.

1. The sender creates and transmits the PM probe packets.

   The IP destination address (DA) on the probe packets is set to the loopback value of the sender itself.

   The transmit timestamp (T1) is added to the payload.

   The probe packet is encapsulated with the label corresponding to the endpoint.
2. The network delivers the PM probe packets following the LSP toward the endpoint.
3. The end-point receives the PM probe packets.

Packets are forwarded back to the sender based on the forwarding entry associated with the IP DA of the PM probe packet. If an LSP exists, the probe packet is encapsulated with the label of the sender.

4. The sender node receives the PM probe packets.

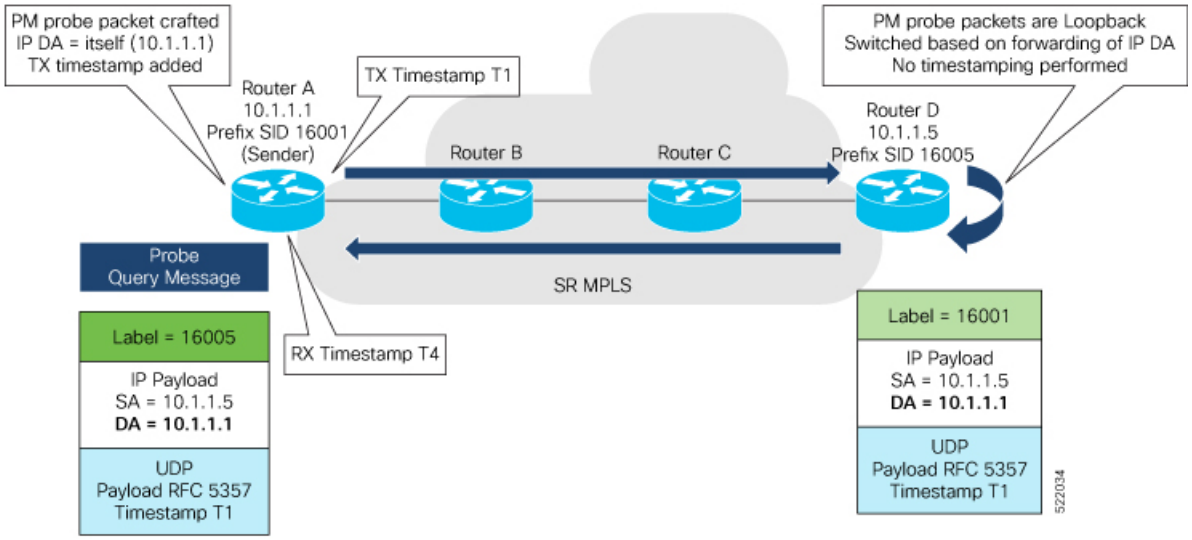
   The received timestamp (T4) stored.

   If the sender node doesn't receive the specified number of probe packets (based on the configured multiplier), the sender node declares the PM session as down.

The following figure illustrates a liveness detection probe toward an IP endpoint learned by the IGP. The network interconnecting the sender and reflector provides MPLS connectivity with Segment Routing.

The liveness detection multiplier is set to 5 to specify the number of consecutive missed probe packets before the PM session is declared as down.

Figure 53: IP Endpoint Liveness Detection



Configuration Example

```
RouterA(config)# performance-measurement
RouterA(config-perf-meas)# endpoint ipv4 1.1.1.5
RouterA(config-pm-ep)# source-address ipv4 1.1.1.1
RouterA(config-pm-ep)# liveness-detection
RouterA(config-pm-ep-ld)# exit
RouterA(config-pm-ep)# exit
RouterA(config-perf-meas)# liveness-profile endpoint default
RouterA(config-pm-ld-ep)# liveness-detection
RouterA(config-pm-ld-ep-ld)# multiplier 5
RouterA(config-pm-ld-ep-ld)# exit
RouterA(config-pm-ld-ep)# probe
RouterA(config-pm-ld-ep-probe)# measurement-mode loopback
```

Running Configuration

```
performance-measurement
 endpoint ipv4 1.1.1.5
  source-address ipv4 1.1.1.1
  liveness-detection
```

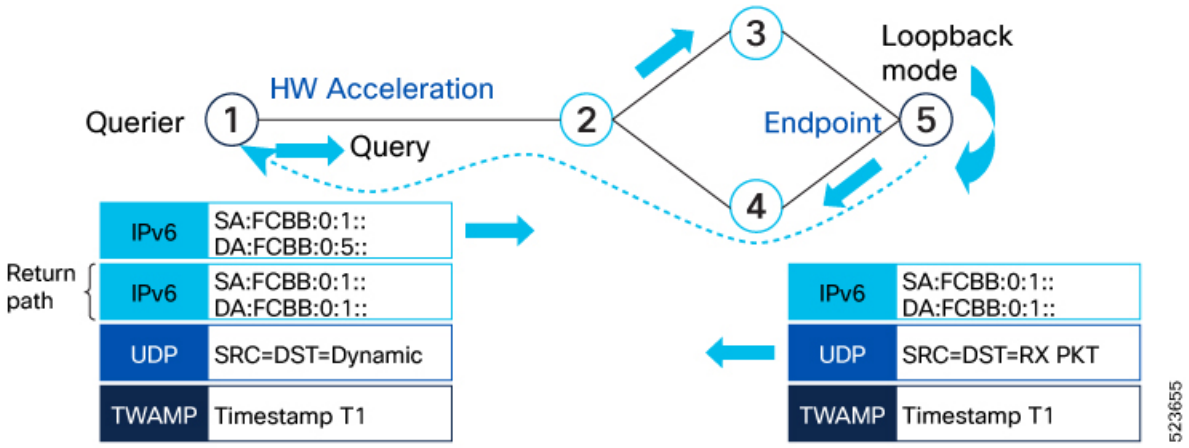
```
  !
 !
 liveness-profile endpoint default
  liveness-detection
   multiplier 5
  !
  probe
   measurement-mode loopback
  !
 !
!
end
```

Verification

```
RouterA# show performance-measurement endpoint ipv4 1.1.1.5

--------------------------------------------------------------------------------
0/RSP0/CPU0
--------------------------------------------------------------------------------

Endpoint name: IPv4-1.1.1.5-vrf-default
  Source address              : 1.1.1.1
  VRF name                    : default
  Liveness Detection          : Enabled
  Profile Keys:
    Profile name              : default
    Profile type              : Endpoint Liveness Detection

  Segment-list                : None
  Session State: Down
  Missed count: 0
```

IP Endpoint Delay Measurement over SRv6 Network Usecase

The following figure illustrates a delay measurement probe toward an IP endpoint over the SRv6 network. The network interconnecting the sender and the reflector provides plain IP connectivity.




```
RP/0/RSP0/CPU0:ios#configure
RP/0/RSP0/CPU0:ios(config)#performance-measurement
RP/0/RSP0/CPU0:ios(config-perf-meas)#endpoint ipv6 FCBB:0:1::
RP/0/RSP0/CPU0:ios(config-pm-ep)#delay-measurement
RP/0/RSP0/CPU0:ios(config-pm-ep-dm)#delay-profile name test
```

```
RP/0/RSP0/CPU0:ios(config-pm-ep-dm)#exit
RP/0/RSP0/CPU0:ios(config-pm-ep)#exit
RP/0/RSP0/CPU0:ios(config-perf-meas)#liveness-profile name test
RP/0/RSP0/CPU0:ios(config-pm-ld-profile)#probe
RP/0/RSP0/CPU0:ios(config-pm-ld-probe)#flow-label explicit 100 200 300
RP/0/RSP0/CPU0:ios(config-pm-ld-probe)#
```

The following example shows how to use flow label for dealy profile for a default endpoint:

```
RP/0/RSP0/CPU0:ios#configure
RP/0/RSP0/CPU0:ios(config)#performance-measurement
RP/0/RSP0/CPU0:ios(config-perf-meas)#delay-profile endpoint default
RP/0/RSP0/CPU0:ios(config-pm-dm-ep)#probe
RP/0/RSP0/CPU0:ios(config-pm-dm-ep-probe)#flow-label explicit 100 200 300
```

Show Running Configuration

```
performance-measurement
 endpoint ipv6 FCBB:0:1::
  delay-measurement
   delay-profile name test
  !
 !
 liveness-profile name test
  probe
   flow-label explicit 100 200 300
  !
 !
!
```

Verification

The show output displays the delay information for the endpoint.

```
Router# show performance-measurement endpoint detail
Endpoint name: IPv6-FCBB:0:1::-vrf-default
  Source address              : 192::2
  VRF name                    : default
  Liveness Detection          : Enabled
  Profile Keys:
    Profile name              : default
    Profile type              : Endpoint Liveness Detection
  Segment-list                : None
  Liveness Detection session:
    Session ID                : 4109
    Flow-label                : 1000
    Session State: Up
    Last State Change Timestamp: Jan 23 2024 16:06:01.214
    Missed count: 0

  Liveness Detection session:
    Session ID                : 4110
    Flow-label                : 2000
    Session State: Up
    Last State Change Timestamp: Jan 23 2024 16:06:01.214
    Missed count: 0

  Segment-list                : test-dm-two-carrier-sl2
    FCBB:0:1:2:e004::/64
      Format: f3216
    FCBB:0:1:3:e000::/64
      Format: f3216
    FCBB:0:1:2:e004::/64
```

```
    Format: f3216
  FCBB:0:1:2:e000::/64
    Format: f3216
  FCBB:0:1:1:e000::/64
    Format: f3216
  FCBB:0:1:1:e004::/64
    Format: f3216
  FCBB:0:1:4:e000::/64
    Format: f3216
  FCBB:0:1:4::/48
    Format: f3216
Liveness Detection session:
  Session ID               : 4111
  Flow-label               : 1000
  Session State: Up
  Last State Change Timestamp: Jan 23 2024 16:06:01.217
  Missed count: 0

Liveness Detection session:
  Session ID               : 4112
  Flow-label               : 2000
  Session State: Up
  Last State Change Timestamp: Jan 23 2024 16:06:01.217
  Missed count: 0
```

SR Policy End-to-End Delay Measurement

Feature Name	Release	Description
SR Policy End-to-End Delay Measurement	Release 7.5.2	This feature allows you to monitor the end-to-end delay experienced by the traffic sent over an SR policy to ensure that the delay does not exceed the requested "upper-bound" and violate SLAs.

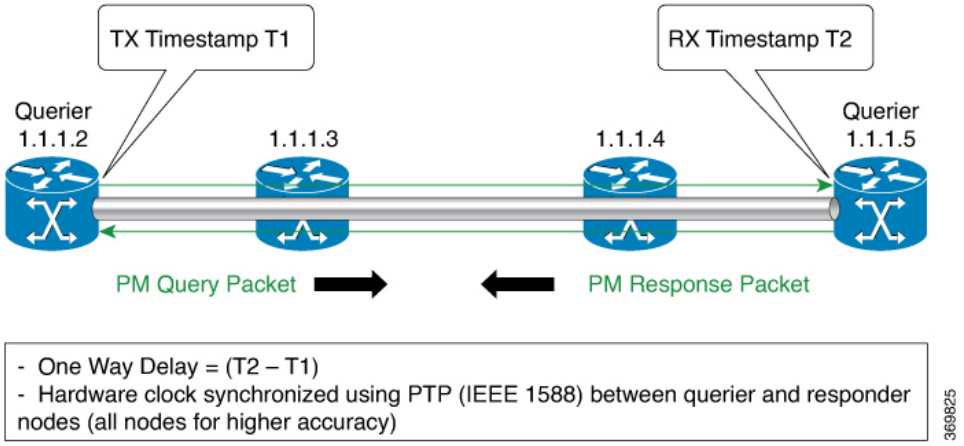
The PM for SR Policy uses IP/UDP packet format defined in RFC 5357 (TWAMP-Light) for probes.

The extended TE link delay metric (minimum-delay value) can be used to compute paths for SR policies as an optimization metric or as an accumulated delay bound.

There is a need to monitor the end-to-end delay experienced by the traffic sent over an SR policy to ensure that the delay does not exceed the requested "upper-bound" and violate SLAs. You can verify the end-to-end delay values before activating the candidate-path or the segment lists of the SR policy in forwarding table, or to deactivate the active candidate-path or the segment lists of the SR policy in forwarding table.

Note The end-to-end delay value of an SR policy will be different than the path computation result (for example, the sum of TE link delay metrics) due to several factors, such as queuing delay within the routers.

Figure 54: Performance Measurement for SR Policy End-to-End Delay



The PM query and response for end-to-end SR Policy delay can be described in the following steps:

1. The local-end router sends PM query packets periodically to the remote side once the egress line card on the router applies timestamps on packets.
2. The ingress line card on the remote-end router applies time-stamps on packets as soon as they are received.
3. The remote-end router sends the PM packets containing time-stamps back to the local-end router.
4. One-way delay is measured using the time-stamp values in the PM packet.

Restrictions and Usage Guidelines for PM for SR Policy Delay

Hardware clocks must be synchronized between the querier and the responder nodes of the link using PTP for one-way delay measurement.

Enable One-Way Delay Mode

This example shows how to enable one-way delay mode.

When one-way delay mode is enabled, an IP/UDP TLV (defined in RFC 7876) is added in the query packet to receive the PM reply via IP/UDP. Hardware clocks must be synchronized between querier and responder nodes (using PTP).

```
Router(config)# performance-measurement delay-profile sr-policy default
Router(config-pm-dm-intf)# probe measurement-mode one-way
Router(config-perf-meas)# exit
```

Configuring Performance Measurement Parameters

This example shows how to configure performance-measurement parameters for SR policy delay as a global default profile. The default values for the different parameters in the PM for SR policy delay is given as follows:

- **probe**: The default mode for probe is one-way delay measurement.
- **tx-interval**: Interval for sending probe packet. The default value is 3000000 microseconds and the range is from 3300 to 15000000 microseconds.
- **computation interval**: Interval for metric computation. Default is 30 seconds; range is 1 to 3600 seconds.

- **protocol**:
  - **twamp-light**: SR Policy delay measurement using RFC 5357 with IP/UDP encap. This is the default protocol.
- **tos**: Type of Service
  - **dscp** *value*: The default value is 0 and the range is from 0 to 63.
  - **traffic-class** *value*: The default value is 0 and the range is from 0 to 7.
- **advertisement threshold-check**: The advertisement threshold-check has three types:
  - **average-delay**: Enable average-delay threshold-check.
  - **maximum-delay**: Enable maximum-delay threshold-check.
  - **minimum-delay**: Enable minimum-delay threshold-check.

Note The default value of periodic **advertisement threshold-check** is **maximum-delay**.

- **periodic advertisement**: Periodic advertisement is enabled by default.
- **periodic-advertisement interval**: The default value is 120 seconds and the interval range is from 30 to 3600 seconds.
- **periodic-advertisement threshold**: The default value of periodic advertisement threshold is 10 percent and the range is from 0 to 100 percent.
- **periodic-advertisement minimum-change**: The default value is 500 microseconds (usec) and the range is from 0 to 100000 microseconds.
- **accelerated advertisement**: Accelerated advertisement is disabled by default.
- **accelerated-advertisement threshold**: The default value is 20 percent and the range is from 0 to 100 percent.
- **accelerated-advertisement minimum**: The default value is 500 microseconds and the range is from 1 to 100000 microseconds.

```
Router(config)# performance-measurement delay-profile sr-policy default
Router(config-pm-dm-srpolicy)# probe
Router(config-pm-dm-srpolicy-probe)# tx-interval 60000
Router(config-pm-dm-srpolicy-probe)# computation-interval 60
Router(config-pm-dm-srpolicy-probe)# protocol twamp-light
Router(config-pm-dm-srpolicy-probe)# tos dscp
Router(config-pm-dm-srpolicy-probe)# exit

Router(config-pm-dm-srpolicy)# advertisement
Router(config-pm-dm-srpolicy-adv)# periodic
Router(config-pm-dm-srpolicy-adv-per)# interval 60
Router(config-pm-dm-srpolicy-adv-per)# minimum-change 1000
Router(config-pm-dm-srpolicy-adv-per)# threshold 20
Router(config-pm-dm-srpolicy-adv-per)# exit

Router(config-pm-dm-srpolicy-adv)# accelerated
```

```
Router(config-pm-dm-srpolicy-adv-acc)# minimum-change 1000
Router(config-pm-dm-srpolicy-adv-acc)# threshold 10
Router(config-pm-dm-srpolicy-adv-acc)# exit

Router(config-pm-dm-srpolicy-adv)# threshold-check minimum-delay
Router(config-pm-dm-srpolicy-adv)# exit
Router(config-pm-dm-srpolicy)#
```

Configure the UDP Destination Port

Configuring the UDP port for TWAMP-Light protocol is optional. By default, PM uses port 862 as the TWAMP-reserved UDP destination port.

The UDP port is configured for each PM measurement probe type (delay, loss, protocol, authentication mode, etc.) on querier and responder nodes. If you configure a different UDP port, the UDP port for each PM measurement probe type must match on the querier and the responder nodes.

Note The same UDP destination port is used for delay measurement for links and SR Policy.

This example shows how to configure the UDP destination port.

```
Router(config)# performance-measurement
Router(config-perf-meas)# protocol twamp-light
Router(config-pm-protocol)# measurement delay unauthenticated
Router(config-pm-proto-mode)# querier-dst-port 12000
```

Enable Performance Measurement for SR Policy

This example shows how to enable PM for SR policy delay for a specific policy.

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# policy foo
Router(config-sr-te-policy)# performance-measurement
Router(config-sr-te-policy-perf-meas)# delay-measurement
```

SR Policy Probe IP/UDP ECMP Hashing Configuration

This example shows how to configure SR Policy ECMP IP-hashing mode.

- The destination IPv4 address 127.x.x.x – 127.y.y.y is used in the Probe messages to take advantages of 3-tuple IP hashing (source-address, destination-address, and local router ID) for ECMP paths of SR-MPLS Policy.

Note The destination IPv4 address must be 127/8 range (loopback), otherwise it will be rejected.

- One PM session is always created for the actual endpoint address of the SR Policy.
- You can specify the number of IP addresses to sweep. The range is from 0 (default, no sweeping) to 128.
- Platforms may have a limitation for large label stack size to not check IP address for hashing.

```
Router(config)# performance-measurement delay-profile sr-policy default
Router(config-pm-dm-srpolicy)# probe
Router(config-pm-dm-srpolicy-probe)# sweep
Router(config-pm-dm-srpolicy-probe-sweep)# destination ipv4 127.0.0.1 range 28
```

Path Tracing in SRv6 Network

Table 80: Feature History Table

Feature Name	Release	Description
Path Tracing Source and Sink Nodes	Release 24.1.1	You can now view the Path Tracing Source and Sink nodes, which are responsible for handling IPv6 transit traffic. This feature also provides full characterization of the packet delivery path which includes real-time information to check the current status of the network, such as whether packets are being diverted due to a breach. It also allows for the pinpointing of the exact location of problems between routers, and ensures that traffic flows according to specified priorities for Quality of Service (QoS) enforcement. This feature introduces a new behavior keyword **utef** under the **route (static** command.
Path Tracing Midpoint Node	Release 7.8.1	Path Tracing (PT) provides a log or record of the packet path as a sequence of interface IDs along with its time stamp. In Path Tracing, a node can behave as a source, midpoint, or sink node. The Path Tracing Midpoint feature is implemented in this release which measures the hop-by-hop delay, traces the path in the network and collects egress interface load information and interface Id, and stores them in the Midpoint Compressed Data (MCD) section of Hop-by-Hop Path Tracing (HbH-PT) header. This feature provides visibility to the Path Tracing Midpoint node that handles IPv6 transit in Path Tracing and full characterization of the packet delivery path. It provides real time information and the current status of the network. This feature introduces the following command: • **performance-measurement interface**

Operators do not know the actual path that the packets take within their network. This makes operations, such as troubleshooting routing problems, or verifying Equal-Cost Multipath (ECMP), a complex problem. Also, operators want to characterize the network in terms of delay and load on a per-hop basis.

Knowledge of the Path Tracing Midpoint helps the operators to troubleshoot the routing problems faster.

This feature allows the operators to:

- Detect the actual path the packet takes between any two nodes in network (A and Z).
- Measure the end-to-end delay from A to Z.
- Measure the per-hop delay at each node on the path from A to Z.
- Detects the load on each router that forwards the packet from A to Z

Path Tracing (PT) provides a log or record of the packet path as a sequence of interface IDs along with its time-stamp. In addition, it provides a record of end-to-end delay, per-hop delay, and load on each egress interface along the packet delivery path.

In Path Tracing, a node can behave as a source, midpoint, or a sink node.

The source node generates and injects probe packets toward a destination node to trace the time-stamp and interface ID along the path of the probe packet. The Interface ID value of 0 means that Path Tracing (PT) is disabled on the interface.

Path Tracing (PT) Midpoint: It is a transit node that performs IPv6 routing. In addition, it records the PT information (MCD) in the HbH-PT.

Note There is no support for Path Tracing Midpoint on transit nodes that perform SRH operations or SRv6 endpoint operations.Support starts from Cisco IOS XR Release 24.1.1.

- Midpoint Compressed Data (MCD): The PT Midpoint along the packet delivery path from the Source to Sink node, stores its PT information into the HbH-PT header. This PT information is called Midpoint Compressed Data (MCD).
- Hop-by-Hop Path Tracing (HbH-PT): In IPv6 The HbH PT Options header is used to carry optional information that is examined and processed by every node along a packet's delivery path. It contains a stack of MCDs.
- PT-Aware Midpoint: A midpoint node that performs plain IPv6 routing or SR Endpoint processing and in addition stores the Path Tracing information in HbH-PT.
- PT-Unaware Midpoint: A midpoint node that performs plain IPv6 routing or SR Endpoint processing and is not capable of performing Path Tracing.
- PT-Legacy Midpoint: A midpoint node that performs plain IPv6 routing or SR Endpoint processing and is not capable of recording Path Tracing information in the HBH-PT. However, it is capable of exporting Path Tracing information directly to the collector, using the node telemetry system.
- PT Source: A Source node is the one that starts a PT session and generates PT probes. Support starts from Cisco IOS XR Release 24.1.1.
- PT Sink: A node that receives the PT probes sent from the Source node containing the information recorded by every PT Midpoint along the path and forwards them to the collector after recording its Path Tracing information. Support starts from Cisco IOS XR Release 24.1.1.
- RC: Regional collector that receives PT probes, parses, and stores them in Timeseries DB

The destination or sink node that receives the PT probes generated by the PT source node, stores PT related info into PT-TLV and forwards them to a Regional Collector (RC). This Regional Collector (RC) parses and stores them in the TimeSeries Database. It uses the information in the Hop-by-Hop Path Tracing (HbH-PT) to construct the packet delivery path and the timestamps at each node.

Limitations and Guidelines

This section lists the limitations of this feature.

- PT Source and Sink nodes are not supported yet. The system can still work as PT midpoint for other devices acting as Source or Sink in the PT network path.
- No support for interface load calculation and recording on IPv6 Path Tracing MidPoint Node. MCD contains interface load value of 0.
- SRv6 Segment Endpoint Midpoint PT (Update DA from SRH.SL and PT MCD update) at midpoint node is not supported. SRv6 endpoint function will not execute properly.
- IPv6 and SRv6 Path Tracing Midpoint Node are supported. SRv6 PT midpoint support Micro-SID (uSID) Shift and Forward action with MCD update.
- Path tracing on Bundled Interfaces and subinterfaces is supported by configuring path-tracing interface-id on physical ports.
- PT unaware IPv6 and SRv6 midpoint forwards transparently without PT update or may punt the packet locally and the control-plane drops the packet.
- PT unaware SRv6 Segment Endpoint Midpoint Node will not execute SRv6 endpoint function. PT packet is forwarded transparently without PT update or punted locally and the control-plane drops the packet.

Configuration Steps

Note These configurations must be done on the Source, Midpoint and Sink routers as shown in the following configuration examples.

- Configuration example of Source node:

  Configure the endpoint with the probe profile name on the source node:

```
Router(config)# performance-measurement
Router(config-perf-meas)# endpoint ipv6 fccc:cc00:9000:fef1::
Router(config-pm-ep)# path-tracing
Router(config-pm-ep-ptrace)# session-id 1011
Router(config-pm-ep-ptrace-sid)# segment-routing traffic-eng seg$
Router(config-pm-ep-ptrace-sid)# probe-profile name PP_12_1
Router(config-pm-ep-ptrace-sid)# source-address ipv6 1::1
Router(config-pm-ep)# path-assurance
Router(config-pm-ep-passurance)# session-id 1111
Router(config-pm-ep-passurance-sid)# segment-routing traffic-eng$
Router(config-pm-ep-passurance-sid)# probe-profile name PP_12_1
```

  Configure probe profile parameters:

```
Router(config)# performance-measurement
Router(config-perf-meas)# path-tracing
Router(config-pm-ptrace)# probe-profile name PP_12_1
Router(config-pm-pr-profile)# tx-interval 3000
Router(config-pm-pr-profile)# flow-label explicit 1000 2000 4000 8$
Router(config-pm-pr-profile)# traffic-class from 16 to 128 increme$
```

Configure Interface ID under Path-tracing for the Source node and for it to participate in the MCD updates inside the probe packets:

```
Router(config)# performance-measurement
Router(config-pm)# interface FourHundredGigE0/0/0/1
Router(config-pm-interf)# path-tracing
Router(config-pm-interf-interf-id)# interface-id 200
Router(config-pm-interf-time)# exit
```

- Configuration example of Midpoint node:

  Configure the Interface ID under Path-tracing for the Midpoint node and for it to participate in the MCD updates inside the probe packets:

```
Router(config)# performance-measurement
Router(config-pm)# interface FourHundredGigE0/0/0/1
Router(config-pm-interf)# path-tracing
Router(config-pm-interf-interf-id)# interface-id 200
Router(config-pm-interf-time)# exit
```

- Configuration example of Sink node.

  Configure Router Static:

```
Router(config)# router static
Router(config-static)# address-family ipv6 unicast
Router(config-static-afi)# fccc:cc00:9000:fef1::/64 segment-routing srv6 endpoint
behavior utef controller-address fccc:cc00:7::
!
```

  Configure Interface ID under Path-tracing for the Sink node and for it to participate in the MCD updates inside the probe packets:

```
Router(config)# performance-measurement
Router(config-pm)# interface FourHundredGigE0/0/0/1
Router(config-pm-interf)# path-tracing
Router(config-pm-interf-interf-id)# interface-id 200
Router(config-pm-interf-time)# exit
```

Running Configuration

- Running configuration example of Source node:

  Configure the endpoint with the probe profile name on the source node:

```
performance-measurement
 endpoint ipv6 fccc:cc00:9000:fef1::
  path-tracing
   session-id 1011
    segment-routing traffic-eng seg$
```

```
    probe-profile name PP_12_1
    source-address ipv6 1::1
  path-assurance
   session-id 1111
    segment-routing traffic-eng$
    probe-profile name PP_12_1
    !
   !
  !
!
```

Configure probe profile parameters:

```
performance-measurement
 path-tracing
  probe-profile name PP_12_1
   tx-interval 3000
   flow-label explicit 1000 2000 4000 8$
   traffic-class from 16 to 128 increme$
   !
  !
 !
```

Configure Interface ID under Path-tracing for the Source node and for it to participate in the MCD updates inside the probe packets:

```
performance-measurement
 interface FourHundredGigE0/0/0/1
  path-tracing
   interface-id 200
   exit
   !
  !
 !
!
```

- Running configuration example of Midpoint node:

  Configure the Interface ID under Path-tracing for the Midpoint node and for it to participate in the MCD updates inside the probe packets:

```
performance-measurement
 interface FourHundredGigE0/0/0/1
  path-tracing
   interface-id 200
   exit
   !
  !
 !
!
```

- Running configuration example of Sink node.

  Configure Router Static:

```
router static
 address-family ipv6 unicast
  fccc:cc00:9000:fef1::/64 segment-routing srv6 endpoint behavior utef controller-address
 fccc:cc00:7::
  !
 !
!
```

Configure Interface ID under Path-tracing for the Sink node and for it to participate in the MCD updates inside the probe packets:

```
performance-measurement
 interface FourHundredGigE0/0/0/1
  path-tracing
   interface-id 200
   exit
   !
  !
 !
!
```

Verification

It is good to check the target interface configuration and performance-measurement configuration for that interface.

Verify using the show commands listed below to check if the PT configuration is applied to the interface properly.

Source Node Verification

```
Router# sh run performance-measurement
performance-measurement
probe-profile name foo
  tx-interval 6000
  flow-label from 100 to 300 increment 10
!
!

Router# sh performance-measurement profile named-profile
Endpoint Probe Measurement Profile Name: foo
  Profile configuration:
    Measurement mode                          : One-way
    Protocol type                             : TWAMP-light
    Type of service:
      TWAMP-light DSCP                        : 48
    TX interval                               : 6000000 (effective: 6000000) uSec
    Destination sweeping mode                 : Disabled
    Liveness detection parameters:
      Multiplier                              : 3
      Logging state change                    : Disabled

    Hop Limit                                 : 255
    Flow Label Count                          : 21
      Flow Labels: 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230,
240,
                   250, 260, 270, 280, 290, 300
    Packet Size Count                         : 0
    Traffic Class Count                       : 0


Router#sh run performance-measurement
performance-measurement
endpoint ipv6 bbbb::
  path-assurance
   session-id 11
   !
  !
!
source-address ipv6 aaaa::
```

```
!

Router# sh performance-measurement endpoint
Endpoint name: IPv6-bbbb::-vrf-default
  Source address              : Unknown
  VRF name                    : default
  Probe Measurement           : Enabled
  Profile Keys:
    Profile name              : default
    Profile type              : Endpoint Probe Measurement

Run this show command to verify the probe sessions:
Router# show performance-measurement probe-sessions
Transport type              : Endpoint
Measurement type            : Probe
Endpoint name               : IPv6-bbbb:bbbb:2::-vrf-default
endpoint                    : bbbb:bbbb:2::
source                      : bbbb:bbbb:1::
vrf                         : default
Segment-list                :
Path Tracing session:
  Session ID          : 10
  Profile Keys:
    Profile name        : pt1
    Profile type        : Probe
  Current status:
    Packet sent every 0.30000 seconds (value stretched for rate-limiting)
    Next packet will be sent in 0.20 seconds


Transport type              : Endpoint
Measurement type            : Probe
Endpoint name               : IPv6-bbbb:bbbb:2::-vrf-default
endpoint                    : bbbb:bbbb:2::
source                      : bbbb:bbbb:1::
vrf                         : default
Segment-list                :
Path Tracing session:
  Session ID          : 11
  Profile Keys:
    Profile name        : pt2 (Profile not found)
    Profile type        : N/A
  Current status:
    Not running: Profile is not configured

Transport type              : Endpoint
Measurement type            : Probe
Endpoint name               : IPv6-bbbb:bbbb:2::-vrf-default
endpoint                    : bbbb:bbbb:2::
source                      : bbbb:bbbb:1::
vrf                         : default
Segment-list                :
Path Assurance session:
  Session ID          : 20
  Profile Keys:
    Profile name        : pa1
    Profile type        : Probe
  Current status:
    Packet sent every 0.30000 seconds (value stretched for rate-limiting)
    Next packet will be sent in 0.24 seconds

Run this show command to view the summary of all the probe sessions:
Router# show performance-measurement summary
Measurement Information:
```

```
  Total interfaces with PM sessions            : 0
  Total SR Policies with PM sessions           : 0
  Total Endpoints with PM sessions             : 1
  Total RSVP-TE tunnels with PM sessions       : 0

    Global Counters:
      Total packets sent                       : 0
      Total query packets received             : 0
      Total invalid session id                 : 0
      Total missing session                    : 0

    Probe sessions:
      Total sessions                           : 3
       Path-tracing sessions:
          Total running sessions               : 1
          Total running error sessions         : 0
        Path-assurance sessions:
          Total running PA sessions            : 1
          Total running error PA sessions      : 0
      Counters:
        Path-tracing packets:
          Total sent                           : 3063
          Total sent errors                    : 0
        Path-assurance packets:
          Total sent                           : 470
          Total sent errors                    : 0
```

```
Router# show cef interface fourHundredGigE 0/0/0/1
FourHundredGigE0/0/0/1 is up if_handle 0x0f000208 if_type IFT_FOURHUNDREDGE(0xcd)
  idb info 0x94dfbf88 flags 0x30001 ext 0x0
  Vrf Local Info (0x0)
  Interface last modified, create
  Reference count 1       Next-Hop Count 0
  PT (path tracing) is enabled: id:0xC8 load_in:0x0 load_out:0x0 tts:0x3
  Protocol Reference count 0
  Protocol ipv4 not configured or enabled on this card
  Primary IPV4 local address NOT PRESENT
```

This is an example of Show CLI with Interface ID:

```
Router# show run performance-measurement
performance-measurement
probe-profile name foo
  tx-interval 6000
  flow-label from 100 to 300 increment 10
!
!
Router# sh performance-measurement profile named-profile
Endpoint Probe Measurement Profile Name: foo
  Profile configuration:
    Measurement mode                           : One-way
    Protocol type                              : TWAMP-light
    Type of service:
      TWAMP-light DSCP                         : 48
    TX interval                                : 6000000 (effective: 6000000) uSec
    Destination sweeping mode                  : Disabled
    Liveness detection parameters:
      Multiplier                               : 3
      Logging state change                     : Disabled

    Hop Limit                                  : 255
    Flow Label Count                           : 21
      Flow Labels: 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230,
```

```
240,
                    250, 260, 270, 280, 290, 300
    Packet Size Count                          : 0
    Traffic Class Count                        : 0
```

```
Router# show cef interface GigabitEthernet 0/2/0/0
GigabitEthernet0/2/0/0 is up if_handle 0x01000020 if_type IFT_GETHERNET(0xf)
  idb info 0x619f16f0 flags 0x30101 ext 0x627ef180 flags 0x30050
  Vrf Local Info (0x626510f0)
  Interface last modified Mar  4, 2022 13:34:43, modify
  Reference count 1       Next-Hop Count 3
  PT (path tracing) is enabled: id:0x40 load_in:0x0 load_out:0x0 tts:0x1
  Forwarding is enabled
  ICMP redirects are never sent
  ICMP unreachables are enabled
  Protocol MTU 1500, TableId 0xe0000000(0x61ccf768)
  Protocol Reference count 4
  Primary IPV4 local address 10.10.10.1
```

```
Router# show performance-measurement interfaces
Interface Name: GigabitEthernet0/2/0/0 (ifh: 0x1000020)
  Delay-Measurement                : Disabled
  Loss-Measurement                 : Disabled
  Path-Tracing                     : Enabled
  Configured IPv4 Address          : 10.10.10.1
  Configured IPv6 Address          : 10:10:10::1
  Link Local IPv6 Address          : fe80::91:e4ff:fe60:6707
  Configured Next-hop Address      : Unknown
  Local MAC Address                : 0291.e460.6707
  Next-hop MAC Address             : 023a.6fc9.cd6b
  In-use Source Address            : 10.10.10.1
  In-use Destination Address       : 10.10.10.2
  Primary VLAN Tag                 : None
  Secondary VLAN Tag               : None
  State                            : Up

  Path-Tracing:
    Interface ID                     : 64
    Load IN                          : 0
    Load OUT                         : 0
    Load Interval                    : 60
    Last FIB Update:
      Updated at: Mar 04 2022 13:34:43.112 (0.392 seconds ago)
      Update reason: Path tracing config
      Update status: Done
```

This is an example of Show CLI without InterfaceID, which means PT is disabled on the target interface. So, you can configure timestamp template:

```
Router# show cef interface GigabitEthernet 0/2/0/0
GigabitEthernet0/2/0/0 is up if_handle 0x01000020 if_type IFT_GETHERNET(0xf)
  idb info 0x619f16f0 flags 0x30101 ext 0x627ef180 flags 0x30050
  Vrf Local Info (0x626510f0)
  Interface last modified Mar  4, 2022 13:49:37, modify
  Reference count 1       Next-Hop Count 3
  Forwarding is enabled
  ICMP redirects are never sent
  ICMP unreachables are enabled
  Protocol MTU 1500, TableId 0xe0000000(0x61ccf768)
  Protocol Reference count 4
  Primary IPV4 local address 10.10.10.1
```

```
Router# sh performance-measurement interfaces
Interface Name: GigabitEthernet0/2/0/0 (ifh: 0x1000020)
  Delay-Measurement                : Disabled
  Loss-Measurement                 : Disabled
  Path-Tracing                     : Enabled
  Configured IPv4 Address          : 10.10.10.1
  Configured IPv6 Address          : 10:10:10::1
  Link Local IPv6 Address          : fe80::91:e4ff:fe60:6707
  Configured Next-hop Address      : Unknown
  Local MAC Address                : 0291.e460.6707
  Next-hop MAC Address             : 023a.6fc9.cd6b
  In-use Source Address            : 10.10.10.1
  In-use Destination Address       : 10.10.10.2
  Primary VLAN Tag                 : None
  Secondary VLAN Tag               : None
  State                            : Up

  Path-Tracing:
    Interface ID                     : 0
    Timestamp Template               : 3
    Load IN                          : 0
    Load OUT                         : 0
    Load Interval                    : 60
    Last FIB Update:
      Updated at: Mar 04 2022 13:49:37.492 (176.418 seconds ago)
      Update reason: Path tracing config
      Update status: Done
```

Sink Node Verification

```
Router# sh segment-routing srv6 sid fccc:cc00:1:fef1:: detail

SID                         Behavior          Context
 Owner               State  RW
fccc:cc00:1:fef1::          uTEF              [fccc:cc01:7::, default]:fccc:cc00:1:fef1::
 ip_static_srv6      InUse  Y
SID Function: 0xfef1
SID context: { controller=fccc:cc01:7::, table-id=0xe0800000 ('default':IPv6/Unicast),
differentiator=fccc:cc00:1:fef1:: }
Locator: 'locator0'
Allocation type: Explicit

Router# sh segment-routing srv6 sid fccc:cc00:1:fef3:: detail
SID                         Behavior          Context
 Owner               State  RW
fccc:cc00:1:fef3::          uTEF              [fccc:cc00:7::, default]:fccc:cc00:1:fef3::
 ip_static_srv6      InUse  Y
SID Function: 0xfef3
SID context: { controller=fccc:cc00:7::, table-id=0xe0800000 ('default':IPv6/Unicast),
differentiator=fccc:cc00:1:fef3:: }
Locator: 'locator0'
Allocation type: Explicit


Router# sh segment-routing srv6 sid fccc:cc01:1:fef2:: detail
SID                         Behavior          Context                           Owner
          State  RW
fccc:cc01:1:fef2::          uTEF              [fccc:cc00:7::, default]:fccc:cc01:1:fef2::
 ip_static_srv6      InUse  Y
SID Function: 0xfef2
SID context: { controller=fccc:cc00:7::, table-id=0xe0800000 ('default':IPv6/Unicast),
differentiator=fccc:cc01:1:fef2:: }
Locator: 'locator1'
```

```
Allocation type: Explicit
Created: Feb 27 11:06:54.999 (00:10:05 ago)
```

Two-Way Active Measurement Protocol Light Source Address Filtering

Table 81: Feature History Table

Feature Name	Release Information	Feature Description
Two-Way Active Measurement Protocol Light Source Address Filtering	Release 7.11.1	You can now restrict unauthorized users from sending packets to the network and prevent compromising the network security and reliability. For a destination UDP port, you can configure the list of IP addresses that can send Two-Way Active Measurement Protocol (TWAMP)-light packets to responder or querier nodes. In earlier releases, the responder or querier node accepted TWAMP-light packets from all IP addresses. The feature introduces these changes: **CLI:** • The **querier** and **responder** keywords are introduced in the **performance-measurement protocol twamp-light measurement delay** command. **YANG Data Models:** • Cisco-IOS-XR-performance-measurement-cfg.yang • Cisco-IOS-XR-perf-meas-oper.yang See (GitHub, Yang Data Models Navigator)

Earlier, the responder node scanned all IP addresses of a querier on the destination UDP port. In other words, the responder node accepted packets from any IP address. See Measurement Modes for more information about querier and responder nodes.

Note The responder is also called the reflector, and the querier is also called the sender.

With this configuration, you can specify the source IP addresses on both the responder and querier nodes. The responder or querier nodes accept packets only from the IP addresses configured in the TWAMP-light protocol, and reject the packets from an IP address that isn't included in the configured list.

All the configured addresses are available for use on all interfaces in the Local Packet Transport Services (LPTS). The configured address filter applies to both default and nondefault VRFs. The TWAMP delay measurement sessions use the configured addresses.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- When you configure the prefix entries on the responder or querier nodes, the PM adds the responder or querier node source IP address to the LPTS. For each prefix, a new LPTS entry is added or created.
- For TWAMP liveness sessions, the PM automatically adds the source IP addresses to the LPTS for if you have configured the prefix entries on the responder or querier nodes.
- As the maximum number of LPTS hardware entries are limited, ensure that enough LPTS entries are allocated for the IP addresses on a line card. You can scale the LPTS configuration to maximum LPTS entries for the PM flow-type. For more details on configuring the LTPS entries for PM flow-type, refer to the *IP Addresses and Services Configuration Guide*.

Note The PM UDP port accepts all incoming IPv4 or IPv6 packets when there are no IPv4 or IPv6 prefix entries configured.

Configure IP address on querier and responder nodes

- The length of the IPv4 and IPv6 prefixes must be less than 32 and 128 respectively.
- The length or mask of the source IP address must be:
  - For IPv4: 0–31
  - For IPv6: 0–127

Configure the IP address on a responder

Perform this task to configure the IP address of a querier on a responder node for delay measurement.

```
Router#configure
Router(config)#performance-measurement
Router(config-perf-meas)#protocol twamp-light
Router(config-pm-protocol)#measurement delay
Router(config-pm-proto-meas)#responder
Router(config-pm-proto-responder)#allow-querier
Router(config-pm-allowed-querier)#address ipv4 10.10.10.1
```

Running Configuration

```
performance-measurement
 protocol twamp-light
  measurement delay
   responder
    allow-querier
     address ipv4 10.10.10.1
    !
   !
  !
 !
!
End
```

Verification

The following example shows output from the IP address of a querier, which is configured on a responder node for delay measurement.

```
Router#show performance-measurement allowed-querier summary
Wed Oct 11 10:41:43.268 UTC

------------------------------------------------------------------------------------------------------------------------
0/RP0/CPU0
------------------------------------------------------------------------------------------------------------------------
Allowed-querier IPv4 prefix                                 : 1
  10.10.10.1/32
Allowed-querier IPv6 prefix                                 : 0

RX UDP port status:
  TWAMP-Light Default Unauthenticated responder port        : 862
    Opened IPv4 port                                        : 862
    IPv4 Port Update Time                                   : Oct 11 2023 10:37:48.118
    Opened IPv6 port                                        : 862
    IPv6 Port Update Time                                   : Oct 11 2023 10:37:47.778
```

Configure the source IP address on a querier

Perform this task to configure the IP address of a responder on a querier node for delay measurement.

```
Router#configure
Router(config)#performance-measurement
Router(config-perf-meas)#protocol twamp-light
Router(config-pm-protocol)#measurement delay
Router(config-pm-proto-meas)#querier
Router(config-pm-proto-querier)#allow-responder
Router(config-pm-allowed-responder)#address ipv4 10.10.10.1
```

Running Configuration

```
performance-measurement
 protocol twamp-light
  measurement delay
   querier
    allow-responder
     address ipv4 10.10.10.1
    !
   !
  !
 !
!
End
```

Verification

The following example shows output from the IP address of a responder, which is configured on a querier node for delay measurement.

```
Router#show performance-measurement allowed-responder summary
Wed Mar 29 19:38:06.381 UTC
------------------------------------------------------------------------------------------------------------------------

0/RP1/CPU0
------------------------------------------------------------------------------------------------------------------------


Allowed-responder IPv4 prefix                                  : 2
  10.10.10.1/32 [Auto]
  3.3.3.3/32
Allowed-responder IPv6 prefix                                  : 1
  fc00:0:1::1/128 [Auto] [Pending Add]
Querier CPU UDP port status:
  TWAMP-Light Default Unauthenticated querier port             : N/A
    Opened IPv4 port                                           : 27643
    IPv4 Port Update Time                                      : Mar 29 2023 18:43:49.080
    Opened IPv6 port                                           : 28274
    IPv6 Port Update Time                                      : Mar 24 2023 20:58:46.150
```

Synthetic Loss Measurement

Table 82: Feature History Table

Feature Name	Release	Description
Synthetic Loss Measurement	Release 24.1.1	You can now proactively monitor and address potential network issues before they impact users by measuring key parameters everywhere, packet loss, and jitter. Using this information, you can plan network capacity optimally and ensure quality of service. Such proactive action is possible because this feature reports synthetic Two-Way Active Measurement Protocol (TWAMP) test packets deployed in delay-profile or delay measurement sessions. It also enables you to set the upper and lower limits and notifies when the synthetic packet loss metric is out of the set limit. The feature introduces these changes: **CLI:** • The optional **anomaly-loss** and **anomaly-check** keywords are introduced in the **performance-measurement delay-profile** command. • **show performance-measurement history** **YANG Data Model** • New XPaths for `Cisco-IOS-XR-um-performance-measurement-cfg` • New operations for `Cisco-IOS-XR-perf-meas-oper.yang` (see GitHub, YANG Data Models Navigator)

Synthetic packets are data packets that are artificially generated for testing or managing a network, as opposed to being generated by normal network data traffic. These packets are unrelated to regular user activities or data. Instead, they are intentionally designed to serve monitoring, testing, and diagnostic purposes.

Synthetic TWAMP test packets that are deployed in the delay profile are used to simulate network traffic, test network performance, and evaluate network infrastructure. They are useful for assessing the quality of service (QoS) of a network, identifying network vulnerabilities, and troubleshooting network issues. These packets enable network administrators to identify potential problems before they impact the overall network performance and reliability. *For more information on TWAMP, refer to the IP Service Level Agreements chapter in the System Monitoring Configuration Guide for Cisco 8000 Series Routers*.

Overall, synthetic packets play a critical role in network management, providing network engineers with a powerful tool for optimizing network performance, identifying and resolving issues.

Note This is an inbuilt feature of delay measurement. To get the synthetic packet loss information for delay-measurement sessions, you must only configure the delay sessions. No additional configuration is required for Synthetic Loss Measurement. However, an optional **anomaly-loss** command is introduced.

Synthetic packets simulate various types of traffic that can be sent with the normal traffic to measure loss or latency, throughput, or packet loss. This is often done in network performance monitoring and testing, where the goal is to understand how the network or a specific network device performs under typical conditions.

This feature enables you to measure and monitor the Synthetic packets lost and help make informed decisions on the following:

- **Network Performance Monitoring**: Synthetic packets can be used to simulate different types of network traffic in order to measure and test the performance of network devices or the network itself.
- **Troubleshooting**: They can be used to diagnose problems in a network by sending them through different parts of the system to see where they may fail or be slowed down.
- **Security**: They can also be used in security applications, such as penetration testing, where synthetic packets are sent to a system to find vulnerabilities.

Configure Synthetic Loss Measurement

The following example enables Synthetic Loss Measurement for an SR Policy.

Note A delay-measurement session is good enough to get the synthetic packet loss automatically, without any extra configuration. The configure shown here is for anomaly loss, which is optional.

```
Router(config)#performance-measurement
Router(config-perf-meas)#delay-profile sr-policy default
Router(config-pm-dm-srpolicy)#advertisement
Router(config-pm-dm-srpolicy-adv)#anomaly-loss
Router(config-pm-dm-srpolicy-adv-anom-loss)#upper-bound 30 lower-bound 20
Router(config-pm-dm-srpolicy-adv-anom-loss)#commit
```

Note Similarly, you can configure Synthetic Loss Measurement for **endpoint default**, **interface default** or **name** (named profile).

Running Configuration

The running configuration for this feature is as shown:

```
performance-measurement
  delay-profile sr-policy default
    advertisement
```

```
      anomaly-loss
        upper-bound 30 lower-bound 20
        !
      !
   !
 !
!
```

Verification

Use the show commands to verify the running configuration as shown:

```
Router# show performance-measurement interfaces delay detail
Interface Name: GigabitEthernet0/2/0/0 (ifh: 0x1000020)
...
      Last advertisement:
      …
      Packets sent: 40, received: 40, lost: 0
      ...
    Next advertisement:
      ...
      Packets sent: 10, received: 10, lost: 0
      …
    Current computation:
      ...
      Packets sent: 4, received: 4, lost: 0
```

```
Router# show performance-measurement history probe-computation interfaces
Interface Name: GigabitEthernet0/2/0/0 (ifh: 0x1000020)
Delay-Measurement history (uSec):
  Probe Start Timestamp     Pkt(TX/RX)   Average       Min       Max
  Aug 01 2023 08:04:15.230       10/10       704       651       779
```

Endpoint

Use these show commands to verify the Endpoint configuration. The example include endpoint delay details and advertisement history. You can also verify the aggregation history.

```
Router# show performance-measurement endpoint delay detail
Endpoint name: IPv4-192.168.0.4-vrf-default
 …
    Last advertisement:
      ...
      Packets sent: 40, received: 40, lost: 0
...
    Next advertisement:
     ...
      Packets sent: 30, received: 30, lost: 0
      ...
    Current computation:
     ...
      Packets sent: 6, received: 6, lost: 0
      …
```

```
Router# show performance-measurement history advertisement endpoint
Endpoint name: IPv4-192.168.0.4-vrf-default
  ...
    Delay-Measurement history (uSec):
```

```
      Advertisement Timestamp   Pkt(TX/RX)   Average       Min       Max        Reason
      Aug 01 2023 08:31:18.835       40/40      3948      3127     15503       PER-MAX
```

RSVP-TE

Use these show commands to verify the RSVP-TE configuration. The example include rsvp-te delay details and the aggregation history. You can also verify the advertisement history.

```
Router# show performance-measurement rsvp-te detail
Tunnel name: tunnel-te1 (ifh: 0xd0)
  ...
    Last advertisement:
      ...
      Packets sent: 40, received:40, lost: 0
      ...
    Next advertisement:
      ...
      Packets sent: 10, received:10, lost: 0
      ...
    Last computation:
      Packets sent: 10, received:10, lost: 0
      ...
    Current computation:
      ...
      Packets sent: 6, received: 6, lost: 0
      …
…

Router# show performance-measurement history aggregation rsvp-te
...
Delay-Measurement history (uSec):
      Aggregation Timestamp     Pkt(TX/RX)   Average       Min       Max
      Aug 01 2023 08:37:23.702       40/40      3372      3172      4109
…
```

SR-Policy

Use these show commands to verify the SR-Policy configuration. The example include delay details and advertisement history. You can also verify the aggregation history.

```
Router# show performance-measurement sr-policy delay detail
SR Policy name: srte_c_10_ep_192.168.0.4
...    Last advertisement:
     …
      Packets sent: 88, received: 88, lost: 0
...
    Next advertisement:
      ...
      Packets sent: 132, received: 132, lost: 0
     ...
    Last computation:

      Packets sent: 4, received: 4, lost: 0
      ...
    Current computation:
      ...
      Packets sent: 4, received: 4, lost: 0
      ...
    Segment-List              : R4
      ...
      Last advertisement:
        ...
        Packets sent: 88, received: 88, lost: 0
```

```
    ...
  Next advertisement:
    ...
    Packets sent: 132, received: 132, lost: 0
    ...
  Last computation:
    Packets sent: 4, received: 4, lost: 0
    ...
  Current computation:
    Packets sent: 4, received: 4, lost: 0
    …
```

```
Router# show performance-measurement history advertisement sr-policy
...
Delay-Measurement history (uSec):
          Advertisement Timestamp   Pkt(TX/RX)   Average       Min       Max        Reason
          Aug 01 2023 10:05:14.072       24/24      3408      3408      3408     ACCEL-MAX
```

Packet loss advertisement

Use this show command to verify multiple configurations with a single show command:

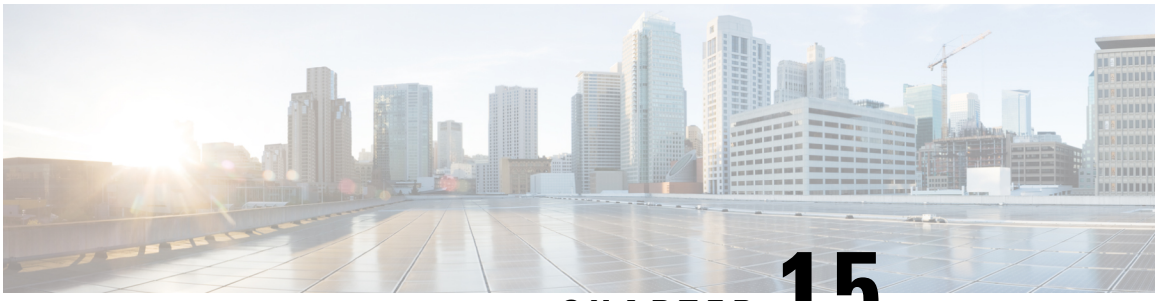
```
Router# show performance-measurement history advertisement sr-policy/endpoint
…
Delay-Measurement history (uSec):
          Advertisement Timestamp   Pkt(TX/RX)   Average       Min       Max        Reason
          Sep 14 2023 11:35:38.180       10/9    3595        3411      3696     NEW-SESSION

          Sep 14 2023 11:34:38.178       10/0     0           0         0      SESSION-ERROR

          Sep 14 2023 11:34:08.177       10/7    3733        3733      3733     ANOM-PKT-LOSS

           Sep 14 2023 11:34:02.177         8/7    3733        3733      3733      PKT-LOSS
           Sep 14 2023 11:33:38.176       10/10    3823        3617      4627         FIRST
...
```

```
Router# show performance-measurement sessions
…
Last advertisement:
    Advertised at: Sep 14 2023 08:47:16.540 (145.777 seconds ago)
    Advertised reason: Periodic timer, max delay threshold crossed
    Advertised delays (uSec): avg: 5373, min: 3992, max: 26212, variance: 0
    Packets sent: 30, received: 30, lost: 0
    Min-Max A flag set: False
    Loss A flag set: False
…
…
Last advertisement:
    Advertised at: Sep 14 2023 08:52:39.603 (12.522 seconds ago)
    Advertised reason: PM session anomaly due to packet loss
    Advertised delays (uSec): avg: 5373, min: 3992, max: 26212, variance: 0
    Packets sent: 30, received: 30, lost: 0
    Min-Max A flag set: False
    Loss A flag set: True
…
Last advertisement:
    Advertised at: Sep 15 2023 11:42:41.594 (47.660 seconds ago)
    Advertised reason: Performance measurement session error
…
```

Note No metrics or A bit is shown when there is a session error.

```
Router# show logging
RP/0/0/CPU0:Sep 14 11:33:38.176 PDT: perf_meas[1001]:
%ROUTING-PERF_MEAS-5-PM_DELAY_EXCEEDS_THRESHOLD : Reason: First advertisement, PM delay
metric avg: 3823, min: 3617, max: 4627, var: 138, min-max anomaly flag: Not set, pkt-loss
anomaly flag: Not set, exceeded threshold on SR Policy N:srte_c_10_ep_192.168.0.4, L:2
RP/0/0/CPU0:Sep 14 11:34:08.177 PDT: perf_meas[1001]:
%ROUTING-PERF_MEAS-5-PM_DELAY_EXCEEDS_THRESHOLD : Reason: PM session anomaly due to packet
 loss, PM delay metric avg: 0, min: 0, max: 0, var: 0, min-max anomaly flag: Not set,
pkt-loss anomaly flag: Set, exceeded threshold on SR Policy N:srte_c_10_ep_192.168.0.4, L:2

RP/0/0/CPU0:Sep 14 11:35:38.180 PDT: perf_meas[1001]:
%ROUTING-PERF_MEAS-5-PM_DELAY_EXCEEDS_THRESHOLD : Reason: PM  session anomaly  due to packet
 loss, PM delay metric avg: 3595, min: 3411, max: 3696, var: 58, min-max anomaly flag: Not
 set, pkt-loss anomaly flag: Not set, exceeded threshold on SR Policy
N:srte_c_10_ep_192.168.0.4, L:2

=============================================================================================
```

In this scenario, the router was running normally and then advertised a packet loss anomaly, the flag is set; later the router resumed to normal, the flag is unset, and it advertised a packet loss anomaly again.



CHAPTER 15

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

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

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

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

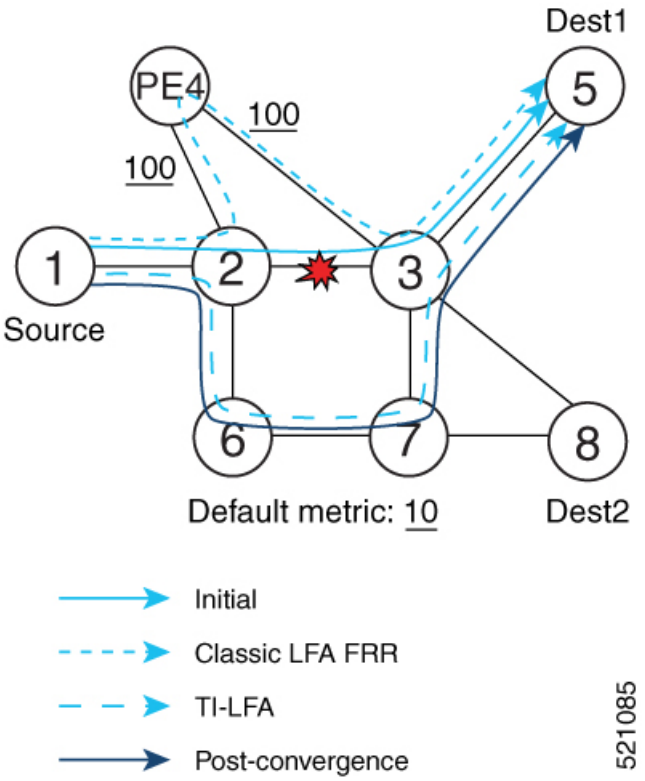
The goal of TI-LFA is to reduce the packet loss that results while routers converge after a topology change due to a link or node failure. Rapid failure repair (< 50 msec) is achieved through the use of pre-calculated backup paths that are loop-free and safe to use until the distributed network convergence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

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

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

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

Figure 55: TI-LFA Repair Path



Node 2 protects traffic to destination Node 5.

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

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

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

Node2 → Node6 → Node7 → Node3 → Node5

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

TI-LFA Protection Types

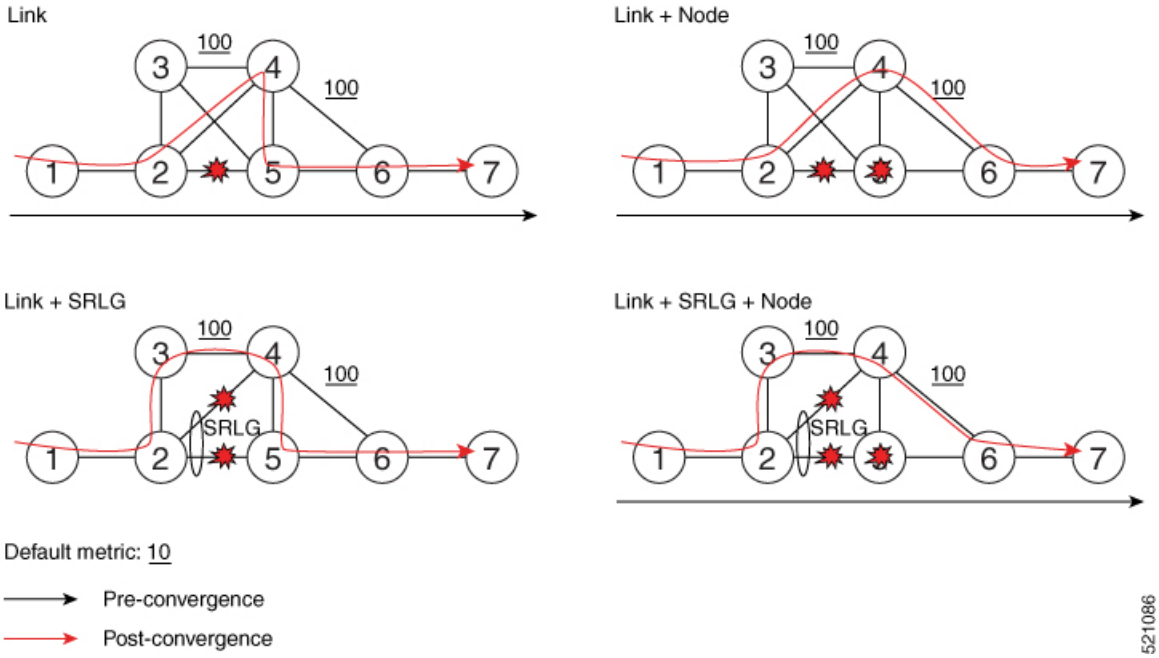
TI-LFA supports the following protection:

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

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

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

Figure 56: TI-LFA Protection Types



- Behaviors and Limitations of TI-LFA, on page 613
- Configuring TI-LFA for IS-IS, on page 615
- Configuring TI-LFA for OSPF, on page 617
- TI-LFA Node and SRLG Protection: Examples, on page 618
- Configuring Global Weighted SRLG Protection, on page 619

Behaviors and Limitations of TI-LFA

Table 83: Feature History Table

Feature Name	Release Information	Feature Description
Increased Number of Labels on Topology-Independent Loop-Free Alternate Backup Paths	Release 7.9.1	This feature increases the maximum number of labels that can be pushed on the Topology-Independent Loop-Free Alternate (TI-LFA) backup path (including the label of the protected prefix) from three to eight. The increase in TI-LFA backup labels provides support for primary and backup paths that go over the same Shared Risk Link Groups (SRLG).

Feature Name	Release Information	Feature Description
BFDv6-triggered TI-LFA	Release 7.3.2	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. BFDv6-triggered TI-LFA allows you to obtain link, node, and SRLG protection using the Bidirectional Forwarding Detection (BFD) over IPv6 protocol.
BFDv4-triggered TI-LFA	Release 7.3.1	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. BFD-triggered TI-LFA allows you to obtain link, node, and SRLG protection by using the Bidirectional Forwarding Detection (BFD) protocol.

The TI-LFA behaviors and limitations are listed below:

- IGP directly programs a TI-LFA backup path requiring eight or fewer labels, including the label of the protected destination prefix.
- Programming of TI-LFA backup paths requiring more than eight labels is not supported.

TI-LFA Functionality	IS-IS[1]	OSPFv2
Protected Traffic Types		
Protection for SR labeled traffic	Supported	Supported
Protection of IPv4 unlabeled traffic	Supported (IS-ISv4)	Supported
Protection of IPv6 unlabeled traffic	Supported (IS-ISv6)	N/A
Protection Types		
Link Protection	Supported	Supported
Node Protection	Supported	Supported
Local SRLG Protection	Supported	Supported
Weighted Remote SRLG Protection	Supported	Supported
Line Card Disjoint Protection	Unsupported	Unsupported
Interface Types		
Ethernet Interfaces	Supported	Supported
Ethernet Bundle Interfaces	Supported	Supported

TI-LFA Functionality	IS-IS[1]	OSPFv2
TI-LFA over GRE Tunnel as Protecting Interface	Unsupported	Unsupported
Additional Functionality		
Maximum number of labels that can be pushed on the backup path (including the label of the protected prefix)	8	8
BFDv4-triggered	Supported	Supported
BFDv6-triggered	Supported	N/A
Prefer backup path with lowest total metric	Unsupported	Unsupported
Prefer backup path from ECMP set	Unsupported	Unsupported
Prefer backup path from non-ECMP set	Unsupported	Unsupported
Load share prefixes across multiple backups paths	Supported	Supported
Limit backup computation up to the prefix priority	Supported	Supported

[1] Unless specified, IS-IS support is IS-ISv4 and IS-ISv6

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:

- Routers are configured with IS-IS.
- Segment routing for IS-IS is configured. See Enabling Segment Routing for IS-IS Protocol, on page 249.

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis 1`	Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. **Note** You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command.
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-isis)# interface GigabitEthernet0/0/0/1`	Enters interface configuration mode.
Step 4	**address-family** {**ipv4** | **ipv6**} [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.
Step 5	**fast-reroute per-prefix** **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix`	Enables per-prefix fast reroute .
Step 6	**fast-reroute per-prefix ti-lfa** **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa`	Enables per-prefix TI-LFA fast reroute link protection.
Step 7	**fast-reroute per-prefix tiebreaker** {**node-protecting** | **srlg-disjoint**} **index** *priority* **Example:** `RP/0/RP0/CPU0:router(config-isis-if-af)#`	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.

	Command or Action	Purpose
	`fast-reroute per-prefix srlg-disjoint index 100`	**Note** The same attribute cannot be configured more than once on an interface.

TI-LFA has been successfully configured for segment routing.

Configuring TI-LFA for OSPF

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

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

Before you begin

Ensure that the following topology requirements are met:

- Routers are configured with OSPF.
- Segment routing for OSPF is configured. See Enabling Segment Routing for OSPF Protocol, on page 283.

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

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

	Command or Action	Purpose
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
Step 3	**area** *area-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# area 1`	Enters area configuration mode.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/0/0/1`	Enters interface configuration mode.
Step 5	**fast-reroute per-prefix** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix`	Enables per-prefix fast reroute.
Step 6	**fast-reroute per-prefix ti-lfa** **Example:** `RP/0/RP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix ti-lfa`	Enables per-prefix TI-LFA fast reroute link protection.
Step 7	**fast-reroute per-prefix tiebreaker** {**node-protecting** | **srlg-disjoint**} **index** *priority* **Example:** `RP/0/RP0/CPU0:router(config-isis-ar-if)# fast-reroute per-prefix srlg-disjoint index 100`	Enables TI-LFA node or SRLG protection and specifies the tiebreaker priority. Valid *priority* values are from 1 to 255. The higher the *priority* value, the higher the priority of the rule. 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
    fast-reroute per-prefix
    fast-reroute per-prefix ti-lfa
    fast-reroute per-prefix tiebreaker node-protecting index 100
```

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

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

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

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

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

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

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 Global Weighted SRLG Protection

A shared risk link group (SRLG) is a set of links sharing a common resource and thus shares the same risk of failure. The existing loop-free alternate (LFA) implementations in interior gateway protocols (IGPs) support SRLG protection. However, the existing implementation considers only the directly connected links while computing the backup path. Hence, SRLG protection may fail if a link that is not directly connected but shares the same SRLG is included while computing the backup path. Global weighted SRLG protection feature provides better path selection for the SRLG by associating a weight with the SRLG value and using the weights of the SRLG values while computing the backup path.

To support global weighted SRLG protection, you need information about SRLGs on all links in the area topology. You can flood SRLGs for remote links using ISIS or manually configuring SRLGS on remote links.

Configuration Examples: Global Weighted SRLG Protection

There are three types of configurations that are supported for the global weighted SRLG protection feature.

- local SRLG with global weighted SRLG protection
- remote SRLG flooding
- remote SRLG static provisioning

This example shows how to configure the local SRLG with global weighted SRLG protection feature.

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg)# name group value 100
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-protection
weighted-global
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint
index 1
RP/0/RP0/CPU0:router(config-isis)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-isis-if)# point-to-point
RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis)# srlg
RP/0/RP0/CPU0:router(config-isis-srlg)# name group1
RP/0/RP0/CPU0:router(config-isis-srlg-name)# admin-weight 5000
```

This example shows how to configure the global weighted SRLG protection feature with remote SRLG flooding.The configuration includes local and remote router configuration. On the local router, the global weighted SRLG protection is enabled by using the **fast-reroute per-prefix srlg-protection weighted-global** command. In the remote router configuration, you can control the SRLG value flooding by using the **advertise application lfa link-attributes srlg** command. You should also globally configure SRLG on the remote router.

The local router configuration for global weighted SRLG protection with remote SRLG flooding is as follows:

```
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-protection
weighted-global
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint
index 1
RP/0/RP0/CPU0:router(config-isis-if-af)# exit
RP/0/RP0/CPU0:router(config-isis)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-isis-if)# point-to-point
RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis-if-af)# exit
RP/0/RP0/CPU0:router(config-isis)# srlg
```

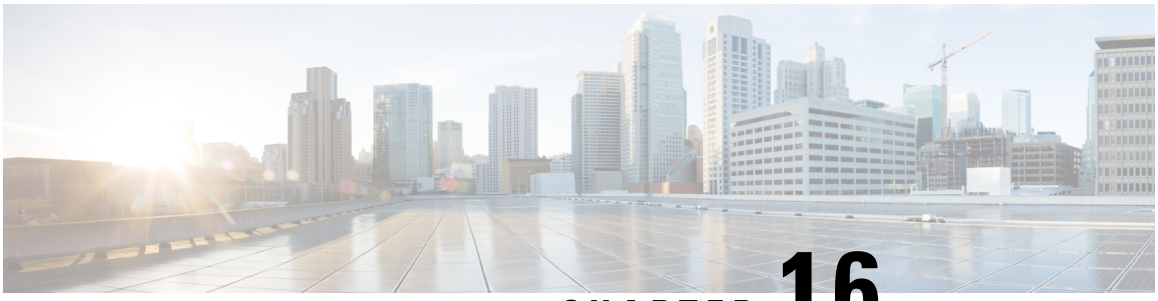
```
RP/0/RP0/CPU0:router(config-isis-srlg)# name group1
RP/0/RP0/CPU0:router(config-isis-srlg-name)# admin-weight 5000
```

The remote router configuration for global weighted SRLG protection with remote SRLG flooding is as follows:

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg)# name group value 100
RP/0/RP0/CPU0:router(config-srlg)# exit
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-af)# advertise application lfa link-attributes srlg
```

This example shows configuring the global weighted SRLG protection feature with static provisioning of SRLG values for remote links. You should perform these configurations on the local router.

```
RP/0/RP0/CPU0:router(config)# srlg
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg-if)# exit
RP/0/RP0/CPU0:router(config-srlg)# interface TenGigE0/0/0/1
RP/0/RP0/CPU0:router(config-srlg-if)# name group1
RP/0/RP0/CPU0:router(config-srlg)# name group value 100
RP/0/RP0/CPU0:router(config-srlg)# exit
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-protection
weighted-global
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint
index 1
RP/0/RP0/CPU0:router(config-isis)# interface TenGigE0/0/0/0
RP/0/RP0/CPU0:router(config-isis-if)# point-to-point
RP/0/RP0/CPU0:router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix
RP/0/RP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
RP/0/RP0/CPU0:router(config-isis)# srlg
RP/0/RP0/CPU0:router(config-isis-srlg)# name group1
RP/0/RP0/CPU0:router(config-isis-srlg-name)# admin-weight 5000
RP/0/RP0/CPU0:router(config-isis-srlg-name)# static ipv4 address 10.0.4.1 next-hop ipv4
address 10.0.4.2
RP/0/RP0/CPU0:router(config-isis-srlg-name)# static ipv4 address 10.0.4.2 next-hop ipv4
address 10.0.4.1
```



CHAPTER 16

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, on page 623
- Configure Segment Routing Microloop Avoidance for IS-IS, on page 625
- Configure Segment Routing Microloop Avoidance for OSPF, on page 630

About Segment Routing Microloop Avoidance

IP hop-by-hop routing may induce microloops (uLoops) at any topology transition. Microloops are a day-one IP challenge. 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 a node converges and sends 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.

Segment Routing resolves the microloop problem. A router with the Segment Routing Microloop Avoidance feature detects if microloops are possible for a destination on the post-convergence path following a topology change associated with a remote link event.

If a node determines that a microloop could occur on the new topology, the IGP computes a microloop-avoidant path to steer the traffic to that destination loop-free over the post-convergence path.

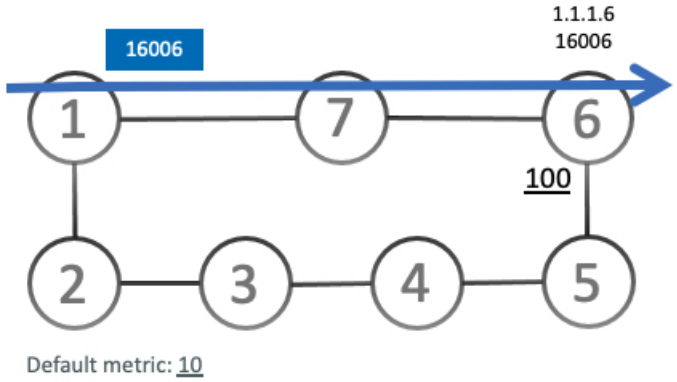
The IGP updates the forwarding table and temporarily (based on a RIB update delay timer) installs the SID-list imposition entries associated with the microloop-avoidant path for the destination with possible microloops.

After the RIB update delay timer expires, IGP updates the forwarding table, removing the microloop-avoidant SID list and traffic now natively follows the post-convergence path.

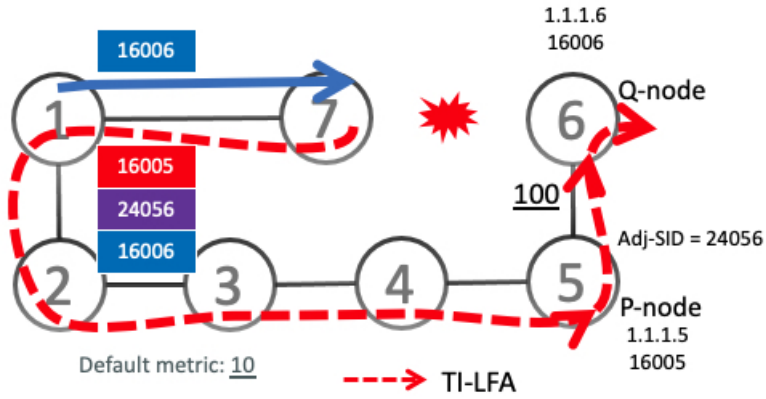
SR microloop avoidance is a local behavior and therefore not all nodes need to implement it to get the benefits.

In the topology below, microloops can occur after the failure of the link between Node6 and Node7.

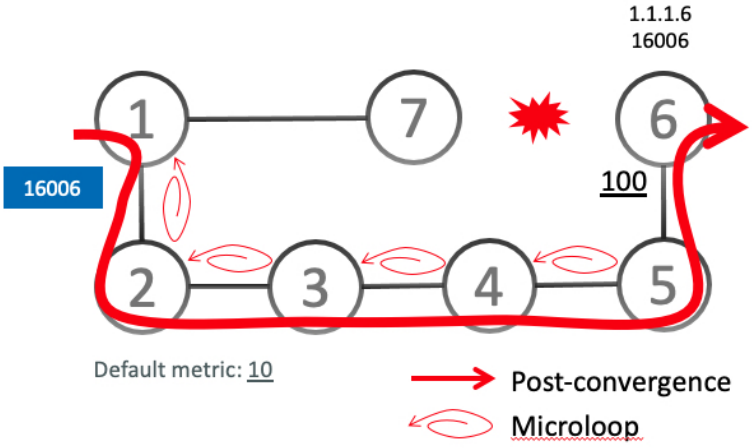
At steady state, Node1 sends traffic to node 6 (16006) via Node7. Node 7 is configured with TI-LFA to protect traffic to Node6.



TI-LFA on Node7 pre-computes a backup path for traffic to Node6 (prefix SID 16006) that will be activated if the link between Node7 and Node6 goes down. In this network, the backup path would steer traffic toward Node5 (prefix SID 16005) and then via link between Node5 and Node6 (adj-SID 24056). All nodes are notified of the topology change due to the link failure.



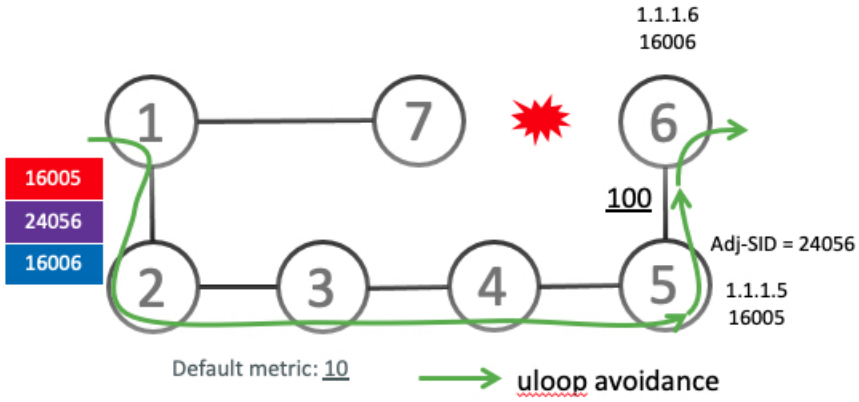
However, if nodes along the path do not converge at the same time, microloops can be introduced. For example, if Node2 converged before Node3, Node3 would send traffic back to Node2 as the shortest IGP path to Node6. The traffic between Node2 and Node3 creates a microloop.



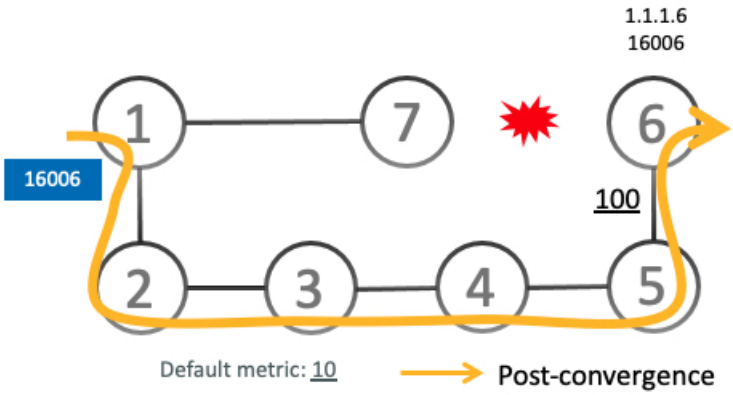
With microloop avoidance configured on Node1, a post-convergence path is computed and possible microloops on the post-convergence path for any destination are detected.

If microloops are possible on the post-convergence path to Node6, a microloop-avoidant path is constructed to steer the traffic to Node6 loop-free over the microloop-avoidant path {16005, 24056, 16006}.

Node1 updates the forwarding table and installs the SID-list imposition entries for those destinations with possible microloops, such as Node6. All nodes converge and update their forwarding tables, using SID lists where needed.



After the RIB update delay timer expires, the microloop-avoidant path is replaced with regular forwarding paths; traffic now natively follows the post-convergence path.



Configure Segment Routing Microloop Avoidance for IS-IS

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

Before you begin

Ensure that the following topology requirements are met:

- Router interfaces are configured as per the topology.
- Routers are configured with IS-IS.

- Segment routing for IS-IS is configured. See Enabling Segment Routing for IS-IS Protocol, on page 249.

SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** {**ipv4** | **ipv6**} [**unicast**]
4. **microloop avoidance segment-routing**
5. **microloop avoidance rib-update-delay** *delay-time*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis 1`	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.
Step 3	**address-family** {**ipv4** | **ipv6**} [**unicast**] **Example:** `RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family and enters router address family configuration mode.
Step 4	**microloop avoidance segment-routing** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# microloop avoidance segment-routing`	Enables Segment Routing Microloop Avoidance.
Step 5	**microloop avoidance rib-update-delay** *delay-time* **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# microloop avoidance rib-update-delay 3000`	Specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The *delay-time* is in milliseconds. The range is from 1-60000. The default value is 5000.

Microloop Avoidance for IS-IS with Per-Prefix Filtering

Table 84: Feature History Table

Feature Name	Release Information	Feature Description
Microloop Avoidance for IS-IS with Per-Prefix Filtering	Release 7.11.1	Currently, when SR Microloop Avoidance for IS-IS is enabled, it applies to all prefixes. This feature allows you to selectively allow or deny specific IPv4 or IPv6 prefixes or routes that may cause microloops, which allows for efficient use of hardware resources and ensures overall network stability. This feature introduces these changes: **CLI**: • The command is modified with the new **route-policy** *name* option for IS-IS. **YANG Data Model:** • This feature extends the native `Cisco-IOS-XR-um-router-isis-cfg.yang` model (see GitHub, YANG Data Models Navigator)

Per-prefix filtering is an enhancement to the existing IOS XR IS-IS SR microloop avoidance feature. Per-prefix filtering allows network administrators to specify a subset of IP prefixes (v4 and v6) to which micro loop avoidance mechanisms can be applied.

Note Per-prefix filtering is available only for SR microloop avoidance and is not supported for local microloop avoidance.

When SR microloop avoidance is enabled, it applies to all prefixes. However, it might be important to preserve hardware resources for certain prefixes. In such a cases, it is beneficial to use SR microloop avoidance per-prefix filtering to allow only those prefixes without such limitations to be subjected to SR microloop avoidance. Per-prefix filtering provides a level of granularity that allows you to apply microloop avoidance only for the prefixes that require it, and to avoid consumption of resources that might otherwise be exhausted.

SR Microloop avoidance per-prefix filtering is configured under the IPv4 or IPv6 address family (AF). It will only be used for filtering in that specific AF. Filtering is applied to prefixes from all algorithms (Algo 0, Flexible Algorithms 128 to 255).

- For SR MLPS – If a prefix has multiple Flexible Algorithm paths and the filtering configuration permits SR microloop avoidance for that prefix, then SR microloop avoidance is allowed for "all" Flexible Algorithm paths associated with that prefix. On the other hand, if a prefix has multiple Flexible Algorithm paths and the filtering configuration prohibits SR microloop avoidance for that prefix, then microloop avoidance is disabled for all Flexible Algorithm paths associated with that prefix.
- For SRv6 – Regardless of the association of the prefix to the algorithm, filtering is applied solely on a per-prefix basis.

SR microloop avoidance per-prefix filtering uses route policies to identify the prefixes subjected to microloop avoidance. When SR microloop avoidance per-prefix filtering is enabled, the prefixes are verified against the route policy as follows:

- If the route policy permits the prefixes (pass), SR microloop avoidance computes the explicit path for the prefixes.
- If the route policy prevents the prefixes from being considered for SR microloop avoidance (drop), it is treated as if there is no explicit path defined for that prefix. The network will rely on the standard routing mechanisms to determine the path for those prefixes after convergence.

Usage Guidelines and Limitations

- SR microloop avoidance per-prefix filtering is supported only for IS-IS.
- A route policy must be defined before it can be attached to the SR microloop avoidance configuration.
- Inline modification of a route policy is not supported. Once a route policy is defined and attached to the SR microloop avoidance configuration, it cannot be modified or removed until the route policy is removed from the SR microloop avoidance configuration.
- The following match types are supported for route policies used for SR microloop avoidance per-prefix filtering:
  - Destination-based match (prefix or prefix set)
  - Tag-based match

Example: Configuration

1. Identify the prefixes to be filtered by defining a prefix set or applying a tags to prefixes.
   - Prefix Set

```
prefix-set pset-sample-ipv4
  2.3.3.3/32,
  2.4.4.4/32
  2.5.5.5/32
end-set

prefix-set pset-sample-ipv6
  2001:0:0:1::/64,
  2001:0:0:2::/64,
  2001:0:0:2::/64,
  2001:0:0:2::/64
end-set
```

   - Tag

```
router isis 1
 interface Loopback1
  address-family ipv4 unicast
   tag 7
   prefix-sid index 7
```

2. Create a route policy for the destinations (prefix set) or tagged prefixes.

- Destination

```
route-policy BAR
  if destination in pset-sample-ipv4 then
    pass
  else
    drop
  endif
end-policy

route-policy BAR2
  if destination in (2.3.3.3/32, 2.4.4.4/32) then
    pass
  else
    drop
  endif
end-policy

route-policy BAR3
  if destination in pset-sample-ipv6 then
    drop
  else
    pass
  endif
end-policy
```

- Tag

```
route-policy FOO
  if tag eq 7 then
    drop
  endif
  pass
end-policy

route-policy FOO2
  if tag eq 7 then
    pass
  else
    drop
  endif
end-policy
```

3. Use the **microloop avoidance segment-routing route-policy** *name* command to attach the route policy to the SR Microloop Avoidance configuration.

```
router isis 1
 address-family ipv4 unicast
  microloop avoidance segment-routing route-policy FOO2
  !
 !
 address-family ipv6 unicast
  microloop avoidance segment-routing route-policy BAR3
```

Verify

Use the **show isis** command to verify that SR microloop avoidance is enabled under the AF and the route policy is applied for per-prefix filtering.

```
Router# show isis

IS-IS Router: 1
  System Id: 0000.0000.0001
    IS Levels: level-2-only
  Manual area address(es):
    49.0001
  Routing for area address(es):
    49.0001
  Multi-Instance Id: 0
  Job Id: 1013
  PID: 61171
  Respawn count: 1
  Started: Thu Feb 23 02:57:58 2023
  LSP MTU: 1400
  LSP Full: level-1: No, level-2: No
  Non-stop forwarding: Cisco Proprietary NSF Restart enabled
  Most recent startup mode: Cold Restart
  TE connection status: Up
  XTC connection status: Up
  Overload Bit: not configured
  Maximum Metric: not configured
  Topologies supported by IS-IS:
    IPv4 Unicast
      Rib connected
      Level-2
        Metric style (generate/accept): Wide/Wide
        Metric: 10
        Microloop avoidance: Enabled
          Configuration: Type: Segment Routing, RIB update delay: 60000 msec, Policy: FOO2
      No protocols redistributed
      Distance: 115
      Advertise Passive Interface Prefixes Only: No
    IPv6 Unicast
      Rib connected
      Level-2
        Metric: 10
        Microloop avoidance: Enabled
          Configuration: Type: Segment Routing, RIB update delay: 5000 msec, Policy: BAR3
      No protocols redistributed
      Distance: 115
      Advertise Passive Interface Prefixes Only: No
  SR-MPLS:
    SRLB allocated: 15000 - 15999
    SRGB allocated: 16000 - 23999
  SRv6:
    Configured locators:
      USID_ALG0 (Active)
      USID_ALG128 (Active)
  Interfaces supported by IS-IS 1:
    Loopback0 is running actively (active in configuration)
    GigabitEthernet0/2/0/0 is running actively (active in configuration)
    GigabitEthernet0/2/0/3 is running actively (active in configuration)
    GigabitEthernet0/2/0/4 is running actively (active in configuration)
    GigabitEthernet0/2/0/6 is running actively (active in configuration)
    GigabitEthernet0/2/0/7 is running actively (active in configuration)
```

Configure Segment Routing Microloop Avoidance for OSPF

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

Before you begin

Ensure that the following topology requirements are met:

- Router interfaces are configured as per the topology.
- Routers are configured with OSPF.
- Segment routing for OSPF is configured. See Enabling Segment Routing for OSPF Protocol, on page 283.

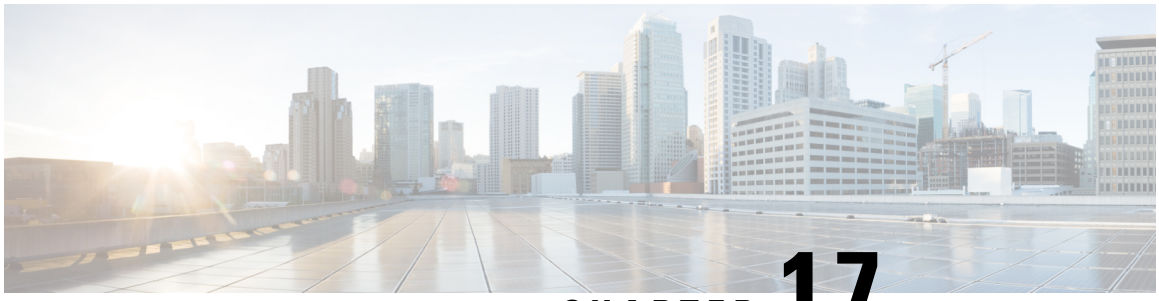
SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enables OSPF routing for the specified routing process, and places the router in router configuration mode.
Step 3	**microloop avoidance segment-routing** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# microloop avoidance segment-routing`	Enables Segment Routing Microloop Avoidance.
Step 4	**microloop avoidance rib-update-delay** *delay-time* **Example:** `RP/0/RP0/CPU0:router(config-ospf)# microloop avoidance rib-update-delay 3000`	Specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The *delay-time* is in milliseconds. The range is from 1-60000. The default value is 5000.



CHAPTER 17

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, on page 633
- Segment Routing and LDP Interoperability, on page 634
- Configuring Mapping Server, on page 636
- Enable Mapping Advertisement, on page 638
- Enable Mapping Client, on page 641

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

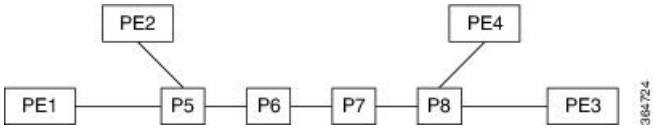
IGP provides mechanisms through which segment routing (SR) interoperate with label distribution protocol (LDP). The control plane of segment routing co-exists with LDP.

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. Use the **commit** or **end** command.
8. **show segment-routing mapping-server prefix-sid-map** [**ipv4** | **ipv6**] [**detail**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**segment-routing** **Example:** `RP/0/RP0/CPU0:router(config)# segment-routing`	Enables segment routing.
Step 3	**mapping-server** **Example:** `RP/0/RP0/CPU0:router(config-sr)# mapping-server`	Enables mapping server configuration mode.
Step 4	**prefix-sid-map** **Example:** `RP/0/RP0/CPU0:router(config-sr-ms)# prefix-sid-map`	Enables prefix-SID mapping configuration mode. **Note** Two-way prefix SID can be enabled directly under IS-IS or through a mapping server.
Step 5	**address-family ipv4**	**ipv6** **Example:** This example shows the address-family for ipv4:	Configures address-family for IS-IS.

	Command or Action	Purpose
	RP/0/RP0/CPU0:router(config-sr-ms-map)# **address-family ipv4** This example shows the address-family for ipv6: RP/0/RP0/CPU0:router(config-sr-ms-map)# **address-family ipv6**	
Step 6	*ip-address*/*prefix-length* *first-SID-value* **range** *range* **Example:** RP/0/RP0/CPU0:router(config-sr-ms-map-af)# **10.1.1.1/32 10 range 200** RP/0/RP0/CPU0:router(config-sr-ms-map-af)# **20.1.0.0/16 400 range 300**	Adds SID-mapping entries in the active local mapping policy. In the configured example: • Prefix 10.1.1.1/32 is assigned prefix-SID 10, prefix 10.1.1.2/32 is assigned prefix-SID 11,…, prefix 10.1.1.199/32 is assigned prefix-SID 200 • Prefix 20.1.0.0/16 is assigned prefix-SID 400, prefix 20.2.0.0/16 is assigned prefix-SID 401,…, and so on.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**show segment-routing mapping-server prefix-sid-map** [**ipv4** | **ipv6**] [**detail**] **Example:** RP/0/RP0/CPU0:router# **show segment-routing mapping-server prefix-sid-map ipv4** Prefix SID Index Range Flags **20.1.1.0/24 400 300** **10.1.1.1/32 10 200** Number of mapping entries: 2 RP/0/RP0/CPU0:router# **show segment-routing mapping-server prefix-sid-map ipv4 detail** Prefix **20.1.1.0/24** SID Index: **400** Range: **300** Last Prefix: 20.2.44.0/24 Last SID Index: 699 Flags: 10.1.1.1/32	Displays information about the locally configured prefix-to-SID mappings. **Note** Specify the address family for IS-IS.

	Command or Action	Purpose
	SID Index: **10** Range: **200** Last Prefix: 10.1.1.200/32 Last SID Index: 209 Flags: Number of mapping entries: 2	

What to do next

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

Enable Mapping Advertisement

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

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

Configure Mapping Advertisement for IS-IS

SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**router isis** *instance-id* **Example:** `RP/0/RP0/CPU0:router(config)# router isis 1`	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.
Step 2	**address-family** { **ipv4**	**ipv6** } [**unicast**] **Example:** The following is an example for ipv4 address family:	Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast`	
Step 3	**segment-routing prefix-sid-map advertise-local** **Example:** `RP/0/RP0/CPU0:router(config-isis-af)# segment-routing prefix-sid-map advertise-local`	Configures IS-IS to advertise locally configured prefix-SID mappings.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5	**show isis database verbose**
	Example:
	`RP/0/RP0/CPU0:router# show isis database verbose`
	`<...removed...>`
	`SID Binding: 10.1.1.1/32 F:0 M:0 S:0 D:0 A:0 Weight:0 Range:200`
	`SID: Start:10, Algorithm:0, R:0 N:0 P:0 E:0 V:0 L:0`
	`SID Binding: 20.1.1.0/24 F:0 M:0 S:0 D:0 A:0 Weight:0 Range:300`
	`SID: Start:400, Algorithm:0, R:0 N:0 P:0 E:0 V:0 L:0`
		Displays IS-IS prefix-SID mapping advertisement and TLV.

Configure Mapping Advertisement for OSPF

SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**router ospf** *process-name* **Example:** `RP/0/RP0/CPU0:router(config)# router ospf 1`	Enables OSPF routing for the specified routing instance, and places the router in router configuration mode.
Step 2	**segment-routing prefix-sid-map advertise-local** **Example:** `RP/0/RP0/CPU0:router(config-ospf)# segment-routing prefix-sid-map advertise-local`	Configures OSPF to advertise locally configured prefix-SID mappings.
Step 3	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 4	**show ospf database opaque-area**
	Example:
	`RP/0/RP0/CPU0:router# show ospf database opaque-area`
	`<...removed...>`
	`Extended Prefix Range TLV: Length: 24`
	`AF : 0`
	`Prefix : 10.1.1.1/32`
	`Range Size: 200`
	`Flags : 0x0`
	`SID sub-TLV: Length: 8`
	`Flags : 0x60`
	`MTID : 0`
	`Algo : 0`
	`SID Index : 10`
		Displays OSP prefix-SID mapping advertisement and TLV.

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.

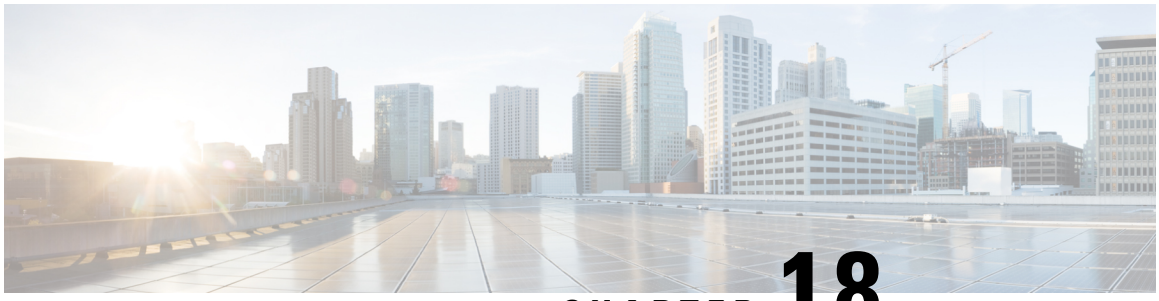
You can re-enable the mapping client functionality by using the **segment-routing prefix-sid-map receive** command.

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

```
RP/0/RP0/CPU0:router(config)# router isis 1
RP/0/RP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RP0/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/RP0/CPU0:router(config)# router ospf 1
RP/0/RP0/CPU0:router(config-ospf)# segment-routing prefix-sid-map receive
```



CHAPTER 18

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 BGP prefix SID, Nil-FEC (forwarding equivalence classes) LSP Ping and Traceroute functionality.

- MPLS Ping and Traceroute for BGP and IGP Prefix-SID, on page 643
- MPLS LSP Ping and Traceroute Nil FEC Target, on page 646
- Segment Routing Ping and Traceroute, on page 648
- Segment Routing Policy Nil-FEC Ping and Traceroute, on page 653
- Segment Routing Data Plane Monitoring , on page 655
- Data Plane Validation Support for SR-MPLS IPv6-based LSPs, on page 660
- MPLS OAM support for SR-TE Policies using IPv6-based LSPs, on page 662

MPLS Ping and Traceroute for BGP and IGP Prefix-SID

MPLS Ping and Traceroute operations for Prefix SID are supported for various BGP and 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
- Combinations of BGP and LDP signaled LSPs

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

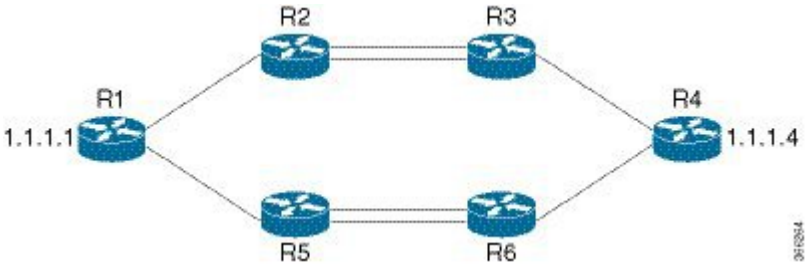
transit node, it checks the TTL value and if it is expired, the packet is passed to the control plane, else the message is forwarded. If the echo message is passed to the control plane, a reply message is generated based on the contents of the request message.

The MPLS LSP Tree Trace (traceroute multipath) operation is also supported for BGP and 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 BGP and 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). The BGP IPv4 prefix sub-TLV contains the prefix SID and the prefix length.

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

These examples use the following topology:



MPLS Ping for Prefix-SID

```
RP/0/RP0/CPU0:router-arizona# ping mpls ipv4 1.1.1.4/32
Thu Dec 17 01:01:42.301 PST

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

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

Type escape sequence to abort.

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

MPLS Traceroute for Prefix-SID

```
RP/0/RP0/CPU0:router-arizona# traceroute mpls ipv4 1.1.1.4/32
Thu Dec 17 14:45:05.563 PST

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

Type escape sequence to abort.

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

MPLS Tree Trace for Prefix-SID

```
RP/0/RP0/CPU0:router-arizona# traceroute mpls multipath ipv4 1.1.1.4/32
Thu Dec 17 14:55:46.549 PST

Starting LSP Path Discovery for 1.1.1.4/32

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

Type escape sequence to abort.

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

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

MPLS LSP Ping and Traceroute Nil FEC Target

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

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

- label stack
- outgoing interface
- nexthop address

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

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

Table 85: LSP Ping and Traceroute Nil FEC Commands

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

Examples: LSP Ping and Traceroute for Nil_FEC Target

These examples use the following topology:

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

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


RP/0/RP0/CPU0:router-utah# show mpls forwarding

Tue Jul  5 13:44:31.999 EDT
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16004  Pop         No ID              Gi0/0/0/1    10.1.1.4        1392
       Pop         No ID              Gi0/0/0/2    10.1.2.2        0
16005  16005       No ID              Gi0/0/0/0    10.1.1.4        0
       16005       No ID              Gi0/0/0/1    10.1.2.2        0
16007  16007       No ID              Gi0/0/0/0    10.1.1.4        4752
       16007       No ID              Gi0/0/0/1    10.1.2.2        0
24000  Pop         SR Adj (idx 0)     Gi0/0/0/0    10.1.1.4        0
24001  Pop         SR Adj (idx 2)     Gi0/0/0/1    10.1.1.4        0
24002  Pop         SR Adj (idx 0)     Gi0/0/0/1    10.1.2.2        0
```

```
24003  Pop         SR Adj (idx 2)     Gi0/0/0/1    10.1.2.2        0
24004  Pop         No ID              tt10         point2point     0
24005  Pop         No ID              tt11         point2point     0
24006  Pop         No ID              tt12         point2point     0
24007  Pop         No ID              tt13         point2point     0
24008  Pop         No ID              tt30         point2point     0
```

Ping Nil FEC Target

```
RP/0/RP0/CPU0:router-arizona# ping mpls nil-fec labels 16005,16007 output interface
GigabitEthernet 0/0/0/1 nexthop 10.1.1.4 repeat 1
Sending 1, 72-byte MPLS Echos with Nil FEC labels 16005,16007,
     timeout is 2 seconds, send interval is 0 msec:

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

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

Traceroute Nil FEC Target

```
RP/0/RP0/CPU0:router-arizona# traceroute mpls nil-fec labels 16005,16007 output interface
GigabitEthernet 0/0/0/1 nexthop 10.1.1.4
Tracing MPLS Label Switched Path with Nil FEC labels 16005,16007, timeout is 2 seconds

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

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

Segment Routing Ping and Traceroute

Table 86: Feature History Table

Feature Name	Release Information	Feature Description
SR OAM for SR Policy (Policy Name / Binding SID / Custom label stack)	Release 7.3.1	This feature extends SR OAM ping and traceroute function for an SR policy (or binding SID)-LSP end-point combination. This addresses the limitations of the Nil-FEC LSP Ping and Traceroute function which cannot perform a ping operation to a segment list that is not associated with an installed SR policy. Also, it cannot validate egress device-specific SR policies.

Segment Routing Ping

The MPLS LSP ping feature is used to check the connectivity between ingress and egress of 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. Segment routing ping is an extension of the MPLS LSP ping to perform the connectivity verification on the segment routing control plane.

Note Segment routing ping can only be used when the originating device is running segment routing.

You can initiate the segment routing ping operation only when Segment Routing control plane is available at the originator, even if it is not preferred. This allows you to validate the SR path before directing traffic over the path. Segment Routing ping can use either generic FEC type or SR control-plane FEC type (SR-OSPF, SR-ISIS). In mixed networks, where some devices are running MPLS control plane (for example, LDP) or do not understand SR FEC, generic FEC type allows the device to successfully process and respond to the echo request. By default, generic FEC type is used in the target FEC stack of segment routing ping echo request. Generic FEC is not coupled to a particular control plane; it allows path verification when the advertising protocol is unknown or might change during the path of the echo request. If you need to specify the target FEC, you can select the FEC type as OSPF, IS-IS, or BGP. This ensures that only devices that are running segment routing control plane, and can therefore understand the segment routing IGP FEC, respond to the echo request.

Configuration Examples

These examples show how to use segment routing ping to test the connectivity of a segment routing control plane. In the first example, FEC type is not specified. You can also specify the FEC type as shown in the other examples.

```
RP/0/RP0/CPU0:router# ping sr-mpls 10.1.1.2/32

Sending 5, 100-byte MPLS Echos to 10.1.1.2/32,
      timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
```

```
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/5 ms

RP/0/RP0/CPU0:router# ping sr-mpls 10.1.1.2/32 fec-type generic

Sending 5, 100-byte MPLS Echos to 10.1.1.2/32,
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms


RP/0/RP0/CPU0:router# ping sr-mpls 10.1.1.2/32 fec-type igp ospf

Sending 5, 100-byte MPLS Echos to 10.1.1.2/32,
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms


RP/0/RP0/CPU0:router# ping sr-mpls 10.1.1.2/32 fec-type igp isis

Sending 5, 100-byte MPLS Echos to 10.1.1.2/32,
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

!!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms


RP/0/RP0/CPU0:router# ping sr-mpls 10.1.1.2/32 fec-type bgp

Sending 5, 100-byte MPLS Echos to 10.1.1.2/32,
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
```

Ping for SR Policy

You can perform the ping operation for an SR policy (or binding SID), and LSP end-point combination. Use the **ping** command's **policy name lsp-end-point** and **policy binding-sid lsp-end-point** options to perform this task. You can instantiate the policy through the CLI, Netconf, PCEP or BGP-TE process.

IPv6 policies are not supported for SR OAM function.

Note As a prerequisite, you must enable the MPLS OAM function.

```
Router(config)# mpls oam
Router(config)# commit

Router# ping sr-mpls policy name srte_c_4_ep_10.0.0.1 lsp-end-point 209.165.201.1
Router# ping sr-mpls policy binding-sid 1000 lsp-end-point 209.165.201.1
```

Segment Routing Traceroute

The MPLS LSP traceroute 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. Segment routing traceroute feature extends the MPLS LSP traceroute functionality to segment routing networks.

Similar to segment routing ping, you can initiate the segment routing traceroute operation only when Segment Routing control plane is available at the originator, even if it is not preferred. Segment Routing traceroute can use either generic FEC type or SR control-plane FEC type (SR-OSPF, SR-ISIS). By default, generic FEC type is used in the target FEC stack of segment routing traceroute echo request. If you need to specify the target FEC, you can select the FEC type as OSPF, IS-IS, or BGP. This ensures that only devices that are running segment routing control plane, and can therefore understand the segment routing IGP FEC, respond to the echo request.

The existence of load balancing at routers in an MPLS network provides alternate paths for carrying MPLS traffic to a target router. The multipath segment routing traceroute feature provides a means to discover all possible paths of an LSP between the ingress and egress routers.

Configuration Examples

These examples show how to use segment routing traceroute to trace the LSP for a specified IPv4 prefix SID address. In the first example, FEC type is not specified. You can also specify the FEC type as shown in the other examples.

```
RP/0/RP0/CPU0:router# traceroute sr-mpls 10.1.1.2/32

Tracing MPLS Label Switched Path to 10.1.1.2/32, timeout is 2 seconds

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

Type escape sequence to abort.

  0 10.12.12.1 MRU 1500 [Labels: implicit-null Exp: 0]
! 1 10.12.12.2 3 ms


RP/0/RP0/CPU0:router# traceroute sr-mpls 10.1.1.2/32  fec-type generic

Tracing MPLS Label Switched Path to 10.1.1.2/32, timeout is 2 seconds

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

Type escape sequence to abort.

  0 10.12.12.1 MRU 1500 [Labels: implicit-null Exp: 0]
! 1 10.12.12.2 2 ms


RP/0/RP0/CPU0:router# traceroute sr-mpls 10.1.1.2/32  fec-type  igp ospf

Tracing MPLS Label Switched Path to 10.1.1.2/32, timeout is 2 seconds

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

Type escape sequence to abort.

  0 10.12.12.1 MRU 1500 [Labels: implicit-null Exp: 0]
! 1 10.12.12.2 2 ms
```

```
RP/0/RP0/CPU0:router# traceroute sr-mpls 10.1.1.2/32  fec-type  igp isis

Tracing MPLS Label Switched Path to 10.1.1.2/32, timeout is 2 seconds

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

Type escape sequence to abort.

  0 10.12.12.1 MRU 1500 [Labels: implicit-null Exp: 0]
! 1 10.12.12.2 2 ms


RP/0/RP0/CPU0:router#traceroute sr-mpls 10.1.1.2/32 fec-type bgp


Tracing MPLS Label Switched Path to 10.1.1.2/32, timeout is 2 seconds

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

Type escape sequence to abort.


  0 10.12.12.1 MRU 1500 [Labels: implicit-null/implicit-null Exp: 0/0]
! 1 10.12.12.2 2 ms
```

This example shows how to use multipath traceroute to discover all the possible paths for a IPv4 prefix SID.

```
RP/0/RP0/CPU0:router# traceroute sr-mpls multipath 10.1.1.2/32

Starting LSP Path Discovery for 10.1.1.2/32

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

Type escape sequence to abort.

!
Path 0 found,
 output interface GigabitEthernet0/0/0/2 nexthop 10.13.13.2
source 10.13.13.1 destination 127.0.0.0
!
Path 1 found,
 output interface Bundle-Ether1 nexthop 10.12.12.2
source 10.12.12.1 destination 127.0.0.0

Paths (found/broken/unexplored) (2/0/0)
Echo Request (sent/fail) (2/0)
```

```
Echo Reply (received/timeout) (2/0)
Total Time Elapsed 14 ms
```

Traceroute for SR Policy

You can perform the traceroute operation for an SR policy (or binding SID), and LSP end-point combination. Use the **traceroute** command's **policy name lsp-end-point** and **policy binding-sid lsp-end-point** options to perform this task. You can instantiate the policy through the CLI, Netconf, PCEP or BGP-TE process.

IPv6 policies are not supported for SR OAM function.

Note As a prerequisite, you must enable the MPLS OAM function.

```
Router(config)# mpls oam
Router(config)# commit

Router# traceroute sr-mpls policy name srte_c_4_ep_10.0.0.1 lsp-end-point 209.165.201.1
Router# traceroute sr-mpls policy binding-sid 1000 lsp-end-point 209.165.201.1
```

Segment Routing Policy Nil-FEC Ping and Traceroute

Segment routing OAM supports Nil-FEC LSP ping and traceroute operations to verify the connectivity for segment routing MPLS data plane. For the existing Nil-FEC ping and traceroute commands, you need to specify the entire outgoing label stack, outgoing interface, as well as the next hop. SR policy Nil-FEC ping and SR policy Nil-FEC traceroute enhancements extend the data plane validation functionality of installed SR policies through Nil-FEC ping and traceroute commands while simplifying the operational process. Instead of specifying the entire outgoing label-stack, interface, and next-hop, you can use the policy name or the policy binding-SID label value to initiate Nil-FEC ping and traceroute operations for the SR policies. Specification of outgoing interface and next-hop is also not required for policy Nil-FEC OAM operations.

Restrictions and Usage Guidelines

The following restrictions and guidelines apply for this feature:

- You cannot select a specific candidate path for SR policy Nil-FEC ping and traceroute.
- You cannot use SR policy Nil-FEC ping or traceroute for non-selected candidate paths.

Examples: SR Policy Nil-FEC Ping

These examples show how to use SR policy Nil-FEC ping for a SR policy. The first example refers the SR policy-name while the second example refers the BSID.

```
RP/0/0/CPU0:router# ping sr-mpls nil-fec policy name POLICY1
Thu Feb 22 06:56:50.006 PST
Sending 5, 100-byte MPLS Echos with Nil FEC for SR-TE Policy POLICY1,
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0
```

```
Type escape sequence to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/5/22 ms


RP/0/0/CPU0:router# ping sr-mpls nil-fec policy binding-sid 100001
Thu Dec 17 12:41:02.381 EST
Sending 5, 100-byte MPLS Echos with Nil FEC with labels [16002,16003],
      timeout is 2 seconds, send interval is 0 msec:
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/3/3 ms
```

Examples: SR Policy Nil-FEC Traceroute

These examples show how to use SR policy Nil-FEC traceroute for a SR policy. The first example refers the SR policy-name while the second example refers the binding SID (BSID).

```
RP/0/0/CPU0:router#  traceroute sr-mpls nil-fec policy name POLICY1
Thu Feb 22 06:57:03.637 PST
Tracing MPLS Label Switched Path with Nil FEC for SR-TE Policy POLICY1, timeout is 2 seconds
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 11.11.11.1 MRU 1500 [Labels: 16003/explicit-null Exp: 0/0]
L 1 11.11.11.2 MRU 1500 [Labels: implicit-null/explicit-null Exp: 0/0] 4 ms
! 2 14.14.14.3 2 ms


RP/0/0/CPU0:router# traceroute sr-mpls nil-fec binding-sid 100001
Tracing MPLS Label Switched Path with Nil FEC with labels [16002/16004], timeout is 2 seconds

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0
Type escape sequence to abort.
  0 99.1.2.1 MRU 4470 [Labels: 16002/16004/explicit-null Exp: 0/0/0]
L 1 99.1.2.2 MRU 4470 [Labels: 16004/explicit-null Exp: 0/0] 3 ms
L 2 99.2.6.6 MRU 4470 [Labels: implicit-null Exp: 0] 3 ms
! 3 99.4.6.4 11 ms
```

Segment Routing Data Plane Monitoring

Table 87: Feature History Table

Feature Name	Release Information	Feature Description
Segment Routing Data Plane Monitoring	Release 7.3.1	Unreported traffic drops in MPLS networks could be difficult to detect and isolate. They can be caused by user configuration, out-of-sync neighbors, or incorrect data-plane programming. Segment Routing Data Plane Monitoring (SR DPM) provides a scalable solution to address data-plane consistency verification and unreported traffic drops. SR DPM validates the actual data plane status of all FIB entries associated with SR IGP prefix SIDs.

The primary benefits of SR DPM include:

- **Automation** – A node automatically verifies the integrity of the actual forwarding entries exercised by transit traffic.
- **Comprehensive Coverage** – Tests validate forwarding consistency for each set of destination prefixes across each combination of upstream and downstream neighbors and across all ECMP possibilities.
- **Scalability** – SR DPM is a highly scalable solution due to its localized detection process.
- **Proactive and Reactive modes of operation** – Solution caters to both continuous and on-demand verification.
- **Standards-based** – SR DPM uses existing MPLS OAM tools and leverages SR to enforce test traffic path.

DPM performs data plane validation in two phases:

- Adjacency Validation—Using special MPLS echo request packets, adjacency validation ensures that all local links are able to forward and receive MPLS traffic correctly from their neighbors. It also ensures that DPM is able to verify all local adjacency SID labels and to flag any inconsistencies, including traffic drops, forwarding by the local or neighboring device to an incorrect neighbor that is not associated with the specified adjacency, or forwarding by the local or neighboring device to the correct neighbor but over an incorrect link not associated with the specified adjacency. DPM validates the following adjacencies for each link when available:
  - Unprotected adjacency
  - Protected adjacency
  - Static adjacency
  - Dynamic adjacency
  - Shared adjacency

Note Observe the following limitations for adjacency validation:

- The adjacency validation phase only validates links that are participating in IGP (OSPF and IS-IS) instances. If one or more link is not part of the IGP, it will not be validated since there are no Adjacency SID labels.
- Adjacency validation only validates physical and bundle links, including broadcast links.

- Prefix Validation—Prefix validation identifies any forwarding inconsistency of any IGP Prefix SID reachable from the device. The validation is done for all upstream and downstream neighbor combinations of each prefix SID, and identifies inconsistencies in the downstream neighbor. The prefix validation phase simulates customer traffic path by validating both ingress and egress forwarding chain at the DPM processing node.

  Since prefix validation is localized to a device running DPM as well as its immediate neighbors, it does not suffer from scale limitations of end-to-end monitoring.

  Prefix validation builds on top of adjacency validation by using special MPLS echo requests that travel to the upstream node, return to the DPM-processing node, and time-to-live (TTL) expire at the immediate downstream node, thus exercising entire forwarding path towards the downstream.

Note Observe the following limitations for prefix validation:

- Because prefix validation builds on top of adjacency validation, if a link is not part of adjacency validation, it is not used in prefix validation.
- If all adjacencies are marked as “Faulty” during adjacency validation, prefix validation is not performed.
- If a node only has downstream links at a specific node, but no upstream node (possible in certain PE node scenarios), Prefix Validation is not performed.
- Prefix validation does not support TI-LFA.

DPM maintains a database of all prefixes and adjacencies being monitored.

The prefix database is populated by registering as a redistribution client to RIB, which enables DPM to keep the database up-to-date whenever IGP pushes a new prefix SID to RIB, deletes an existing prefix SID, or when the path of an existing prefix SID is modified.

DPM maintains the following prefix data:

- IPv4 Prefix
- Prefix Length
- Prefix SID label
- Error stats

DPM also maintains a list of all local adjacencies. DPM maintains a database that contains local links, their respective local and remote adjacency labels and IP addresses, and error stats.

SR-DPM Operation: Example

The following SR-DPM operation example use the following scenarios:

Figure 57: Test Iteration A Path

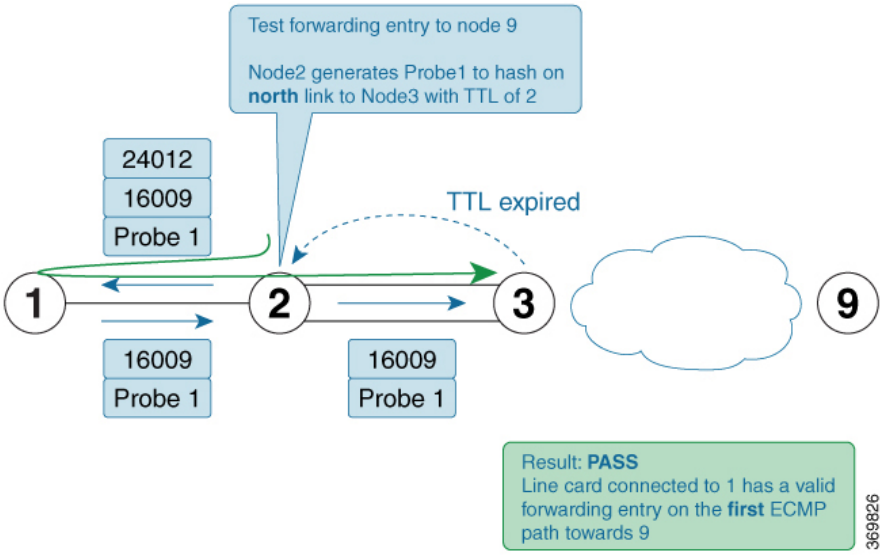
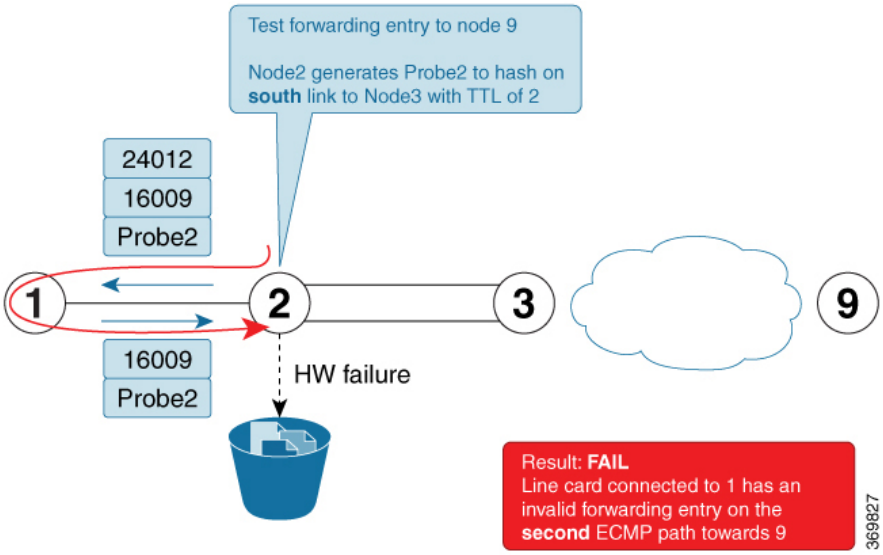


Figure 58: Test Iteration B Path



Node 2 is a DPM-capable device. DPM is enabled ***in proactive mode*** to perform forwarding consistency tests for all prefix-SIDs in the network. For each destination prefix, the router identifies the directly connected upstream and downstream neighbors used to reach a given destination.

Using node 9 as the prefix under test (prefix-SID = 16009), node 1 is designated as the upstream node and node 3 as the downstream nodes with 2 ECMPs.

1. Node 2 generates test traffic (MPLS OAM ping with source_ip of node 2) to test its forwarding for every upstream/downstream combination. In this case, two combinations exist:
   - Prefix-SID node 9 - test iteration A path = Node 2 to Node 1 to Node 2 to Node 3 (via **top** ECMP)
   - Prefix-SID node 9 - test iteration B path = Node 2 to Node 1 to Node 2 to Node 3 (via **bottom** ECMP)
2. Node 2 adds a label stack in order to enforce the desired path for the test traffic. For example, two labels are added to the packet for test iterations A and B:
   - The top label is equal to the adjacency-SID on node 1 for the interface facing node 2 (adjacency SID = 24012). The bottom label is the prefix-SID under test (16009). The test traffic is sent on the interface facing node 1.
   - The top label (after being POPed at node 1) causes the test traffic to come back to node 2. This returning traffic is completely hardware-switched based on the forwarding entry for the prefix-SID under test (16009). Note that the labeled test traffic has a time-to-live (TTL) of 2 and it will never be forwarded beyond the downstream router(s).
   - When test traffic reaches node 3, a TTL expired response is sent back to node 2. If the response packet arrives over the expected interface (**top** ECMP link) then the forwarding verification on node 2 for the first iteration towards node 9 is considered to be a success.
   - The difference between the test traffic for test iteration A and B in this example is the destination_ip of the MPLS OAM ping. Node 2 calculates them in this order to exercise a given ECMP path (if present). Thus, test traffic for iteration A is hashed onto the **top** ECMP and test traffic for iteration B is hashed onto the **bottom** ECMP link.
3. The DPM tests are then repeated for the remaining prefix-SIDs in the network

Configure SR DPM

To configure SR-DPM, complete the following configurations:

- Enable SR DPM
- Configure SR DPM interval timer
- Configure SR DPM rate limit

Enable SR DPM

Use the **mpls oam dpm** command to enable SR DPM and enter MPLS OAM DPM command mode.

```
Router(config)# mpls oam dpm
Router(config-oam-dpm)#
```

Configure SR DPM Interval Timer

Use the **interval** *minutes* command in MPLS OAM DPM command mode to specify how often to run DPM scan. The range is from 1 to 3600 minutes. The default is 30 minutes.

```
Router(config-oam-dpm)# interval 240
Router(config-oam-dpm#
```

Configure SR DPM Rate Limit

Use the **pps** *pps* command in MPLS OAM DPM command mode to rate limit the number of echo request packets per second (PPS) generated by DPM. The range is from 1 to 250 PPS. The default is 50 PPS.

Note If the specified rate limit is more than the rate limit for overall MPLS OAM requests, DPM generates an error message.

```
Router(config-oam-dpm)# pps 45
Router(config-oam-dpm#
```

Verification

```
Router# show mpls oam dpm summary
  Displays the overall status of SR-DPM from the last run.
Router# show mpls oam dpm adjacency summary
  Displays the result of DPM adjacency SID verification for all local interfaces from the
last run.
Router# show mpls oam dpm adjacency interface
  Displays the result of DPM adjacency SID verification for all adjacencies for the specified
 local interface.
Router# show mpls oam dpm counters
  Outputs various counters for DPM from last run as well as since the start of DPM process.
Router# show mpls oam dpm prefix summary
  Displays the result of DPM prefix SID verification for all reachable IGP prefix SIDs from
 the last run.
Router# show mpls oam dpm prefix prefix
  Displays the result of DPM prefix SID verification for the specified prefix including all
 upstream and downstream combinations.
Router# show mpls oam dpm trace
  Returns logged traces for DPM.
```

In addition, the existing **show mpls oam** command is extended to specify DPM counters.

```
Router# show mpls oam counters packet dpm
```

Data Plane Validation Support for SR-MPLS IPv6-based LSPs

Table 88: Feature History Table

Feature Name	Release Information	Feature Description
Data Plane Validation for SR-MPLS IPv6-based Controller Instantiated LSPs	Release 24.2.11	You can now verify the network configuration and paths and policies set up, without interrupting or potentially disrupting live network traffic, for SR-MPLS (Segment Routing over Multiprotocol Label Switching) IPv6-based Label Switched Paths (LSPs). With this feature, you can validate controller instantiated LSPs programmed directly into the forwarding hardware. Previously, SR data plane validation was possible over IPv4-based LSPs. The feature introduces these changes: **CLI:** • The **dataplane-only** keyword is introduced in the **traceroute sr-mpls** and **ping sr-mpls** commands. **YANG Data Models:** • `Cisco-IOS-XR-mpls-traceroute-act.yang` • `Cisco-IOS-XR-mpls-ping-act.yang` See (GitHub, Yang Data Models Navigator)

With this configuration, you can validate the SR-MPLS (Segment Routing over Multiprotocol Label Switching) IPv6-based LSPs and policies without disrupting the live network traffic. You can also validate Service-layer Application Programming Interface (SL-API) initiated LSPs such as controller instantiated LSPs. For more information about SL-API, refer *Use Service Layer API to Bring your Controller on Cisco IOS XR Router* chapter in *Programmability Configuration Guide*.

In the earlier releases, you could perform SR-MPLS data plane validation over IPv4-based LSPs. For more information, refer Segment Routing Ping, on page 648 and Segment Routing Traceroute, on page 650 sections.

Examples: SR-MPLS Data Plane Validation over IPv6-based LSPs

The following example shows how to use segment routing ping to validate SR-MPLS over IPv6-based LSPs:

```
Router#ping sr-mpls dataplane-only 2001:DB8::1/32
Tue Jan 16 15:05:19.120 EST

Sending 5, 100-byte MPLS Echos with Nil FEC to 2001:DB8::1/32,
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/8 ms
```

The following example shows how to use segment routing traceroute to validate SR-MPLS over IPv6-based LSPs:

```
Router#traceroute sr-mpls dataplane-only 2001:DB8::1/32
Tue Jan 16 15:08:54.681 EST

Tracing MPLS Label Switched Path with Nil FEC to 2001:DB8::1/32, timeout is 2 seconds

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

Type escape sequence to abort.

  0 11:11:11::1 MRU 1500 [Labels: 18004/explicit-null Exp: 0/0]
L 1 11:11:11::2 MRU 1500 [Labels: implicit-null/explicit-null Exp: 0/0] 3 ms
! 2 15:15:15::4 3 ms
```

The following example shows how to trace the SR-MPLS LSPs with Nil-FEC that includes labels:

```
Router#traceroute sr-mpls nil-fec labels 18004 output interface GigabitEthernet 0/0/0/0
nexthop 10:10:10::2
Tue Jan 16 15:28:03.162 EST

Tracing MPLS Label Switched Path with Nil FEC with labels [18004], timeout is 2 seconds

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

Type escape sequence to abort.

  0 10:10:10::1 MRU 1500 [Labels: 18004/explicit-null Exp: 0/0]
```

```
L 1 10:10:10::2 MRU 1500 [Labels: implicit-null/explicit-null Exp: 0/0] 2 ms
! 2 15:15:15::4 2 ms
```

MPLS OAM support for SR-TE Policies using IPv6-based LSPs

Table 89: Feature History Table

Feature Name	Release Information	Feature Description
MPLS OAM support for SR-TE Policies using MPLS IPv6-based LSPs	Release 24.2.11	You can now verify the network configuration and paths and SR-TE policies set up, without interrupting or potentially disrupting live network traffic, for SR-MPLS (Segment Routing over Multiprotocol Label Switching) IPv6-based Label Switched Paths (LSPs). Previously, MPLS OAM support was only for IPv4-based LSPs. The feature introduces these changes: **CLI:** The **traceroute sr-mpls** and **ping sr-mpls** commands are extended to support IPv6 nexthop addresses. **YANG Data Models:** • `Cisco-IOS-XR-mpls-traceroute-act.yang` • `Cisco-IOS-XR-mpls-ping-act.yang` See (GitHub, Yang Data Models Navigator)

With this feature, you can now verify the SR-MPLS (Segment Routing over Multiprotocol Label Switching) IPv6-based LSPs and Segment routing for traffic engineering (SR-TE) policies without disrupting the live network traffic. In the earlier releases, SR-MPLS support was limited to IPv4-based LSPs. For more information, refer Segment Routing Ping and Segment Routing Traceroute sections.

MPLS OAM support for SR-TE Policies using MPLS IPv6-based LSPs

The following usage guidelines and limitations apply:

For SR-TE policies, provide a valid LSP endpoint for non-Nil-FEC ping and traceroute operations.

Examples: MPLS OAM support for SR-TE Policies with IPv6-based LSPs

The following example shows how to use segment routing traceroute for SR-TE policies with IPv6-based LSPs:

```
Router#traceroute sr-mpls nil-fec policy name srte_c_40_ep_2001:DB8::1
Tue Feb  6 12:07:38.295 EST

Tracing MPLS Label Switched Path with Nil FEC for SR-TE Policy srte_c_40_ep_2001:DB8::1,
timeout is 2 seconds

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

Type escape sequence to abort.

  0 12:12:12::1 MRU 1500 [Labels: 26134/explicit-null Exp: 0/0]
L 1 12:12:12::3 MRU 1500 [Labels: implicit-null/explicit-null Exp: 0/0] 16 ms
! 2 16:16:16::4 16 ms
```

The following example shows how to use segment routing ping for SR-TE policies with IPv6-based LSPs:

```
Router#ping sr-mpls nil-fec policy name srte_c_40_ep_2001:DB8::1
Tue Feb  6 12:08:28.277 EST

Sending 5, 100-byte MPLS Echos with Nil FEC for SR-TE Policy srte_c_40_ep_2001:DB8::1,
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

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

For more information about configuring SR-TE policies, refer SR-TE Policy Overview section.

The following example shows how to use segment routing traceroute with labels using IPv6 LSPs:

```
Router#traceroute sr-mpls labels 18004 lsp-end-point 2001:DB8::1
Tue Feb  6 12:10:41.928 EST

Tracing MPLS Label Switched Path to NIL FEC with lsp end point 2001:DB8::1, SID Label(s)
[18004], timeout is 2 seconds

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

```
Type escape sequence to abort.

  0 11:11:11::1 MRU 1500 [Labels: 18004/explicit-null Exp: 0/0]
L 1 11:11:11::2 MRU 1500 [Labels: implicit-null/explicit-null Exp: 0/0] 7 ms
! 2 15:15:15::4 3 ms
```

The following example shows how to use segment routing ping with labels using IPv6 LSPs:

```
Router#ping sr-mpls labels 18004 lsp-end-point 2001:DB8::1
Tue Feb  6 12:11:05.349 EST

Sending 5, 100-byte MPLS Echos with NIL FEC with lsp end point 2001:DB8::1, SID Label(s)
[18004],
      timeout is 2 seconds, send interval is 0 msec:

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

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/4 ms
```

INDEX

C

configuring path tracing 593

M

mid 593
    point 593
Multiprotocol Label Switching (MPLS), Segment Routing, IS-IS 249
Multiprotocol Label Switching (MPLS), Segment Routing, OSPF 285

P

path tracing 593
Path tracing midpoint 593