Segment Routing v6 Configuration Guide for Cisco 8000 Series Routers, Cisco IOS XR Releases

First Published: 2025-11-14

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

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Specific changes or updates tied to individual releases are clearly called out within the relevant sections. For a list of features introduced in a specific release, refer to the Release Notes or the IOS XR Feature Finder.

The table lists the release numbers for which this document has been updated since its initial publication.

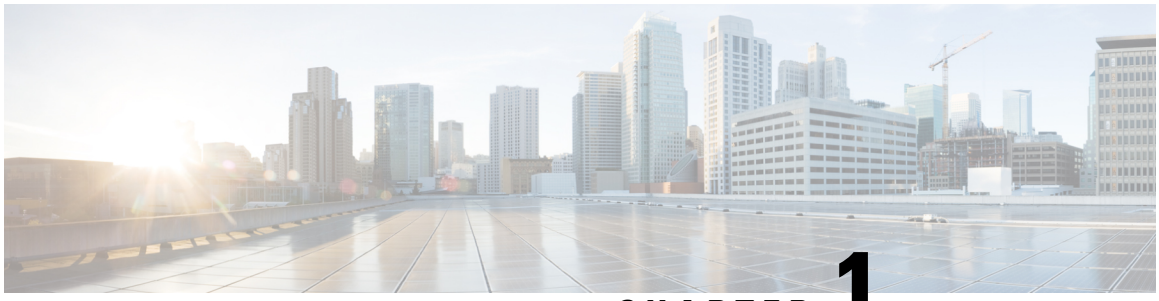
Table 1: Changes to this document

Date	Summary
October 2025	First published for Release 25.3.1

Related resources

- Segment Routing Command Reference
- Release Notes
- IOS XR Feature Finder
- MIBs dashboard
- Error Messages dashboard
- YANG dashboard

Preface



CHAPTER 1

YANG Data Models for SRv6 Features

This chapter provides information about the YANG data models for SRv6 features.

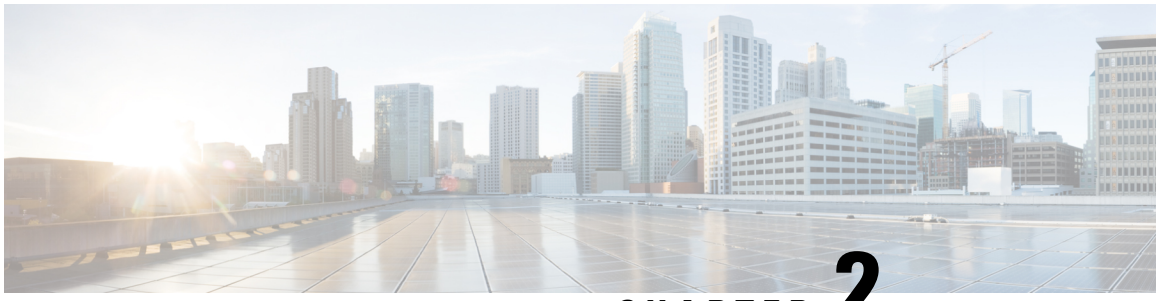
Using YANG Data Models

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 2

SRv6 Fundamentals and Key Concepts

In today's dynamic networking environment, the demand for scalable, efficient, and flexible routing solutions continues to grow. Traditional approaches, such as MPLS, often involve complex protocols and lack native IPv6 support, making them less suitable for modern networks. Segment Routing over IPv6 (SRv6) addresses these challenges by leveraging the native IPv6 protocol to implement a simplified, programmable, and scalable routing framework.

Segment Routing over IPv6

Segment Routing over IPv6 (SRv6) is a network routing framework that

- embeds routing and processing instructions directly into IPv6 packets
- uses the Segment Routing Header (SRH) to encode these instructions
- uses Segment Identifiers (SIDs), which are 128-bit values, to define routing and processing behaviors, and
- enables a network programming model to allow packets to carry instructions for advanced traffic engineering and automated service chaining.

The components of Srv6 include Segment Routing Header and Segment Identifiers. See Key concepts of SRv6, on page 5.

Key difference between SR-MPLS and SRv6

The table compares the key benefits and features of SR-MPLS and SRv6 to highlight their differences and advantages:

Table 2: SR-MPLS and SRv6 comparision

Feature	SR-MPLS (Segment Routing with MPLS)	SRv6 (Segment Routing with IPv6)
Underlying technology	Uses MPLS labels to encode segment routing paths.	Uses IPv6 addresses (Segment Identifiers or SIDs) in IPv6 header.
Packet Header	MPLS label stack steers packets through the network.	IPv6 extension headers carry the segment list for source routing.
Scalability and flexibility	Limited by MPLS label space and MPLS forwarding capabilities.	Larger address space and more flexible network programming due to IPv6 extensibility.
Network requirements	Requires MPLS-enabled infrastructure.	Requires IPv6-enabled infrastructure and SRv6 processing support.

Benefits of SRv6

Modern networks require solutions that support growing scale, advanced services, and seamless operations. Traditional routing technologies often face limitations such as complex architectures, limited scalability, and rigid traffic management. Segment Routing over IPv6 (SRv6) addresses these challenges. It provides a highly programmable, scalable, and efficient solution by leveraging native IPv6 capabilities to simplify operations and enhance service delivery.

SRv6 delivers these key advantages:

- Simplified network architecture: SRv6 consolidates routing functions into a single IPv6-based data plane. This eliminates multiple legacy protocols and streamlines inter-domain routing with basic IP functions and prefix summarization.
- Native IPv6 integration and enhanced scalability: SRv6 fully leverages IPv6's capabilities and address space, ensuring long-term scalability and future readiness. Its flexible, hierarchical Segment Identifier (SID) structure efficiently scales networks, overcoming traditional label limitations and maximizing Forwarding Information Base (FIB) utilization across diverse segments.
- Flexible and intent-based traffic engineering: SRv6 offers precise and flexible path selection through its built-in traffic engineering capabilities. This enables network programming with SIDs, supports intent-based routing via IGP Flex-Algo, and facilitates complex service chaining and Network Function Virtualization (NFV) integration.
- Optimal load balancing: SRv6 efficiently distributes traffic across available paths. It leverages native IPv6 flow label capabilities for Equal-Cost Multi-Path (ECMP) load balancing, ensuring optimal traffic distribution based on inner packet entropy without additional protocols.
- Unified service delivery and cloud integration: SRv6 provides a single, unified IPv6-based data plane for various services, including Virtual Private Networks (VPNs), micro-services, and cloud connectivity. It offers native support for host and cloud environments without protocol translation.
- Seamless deployment and migration: SRv6 integrates smoothly into existing brownfield networks. It supports granular, phased adoption strategies, facilitating a seamless transition to modern network infrastructures.

- Improved reliability and performance: SRv6 enhances network resilience with features like Fast Reroute (FRR). It also enables better Quality of Service (QoS) management through granular control over traffic paths and functions, ensuring robust and high-performing networks.

Use cases for SRv6

SRv6 is a technology that enhances various network applications through several use cases, such as:

- Traffic engineering (TE): SRv6 optimizes network paths to enhance performance, meet SLAs, routing requirements, as well as specific needs such as security or data sovereignty.
- Network segmentation: SRv6 enables the creation of logical network overlays, including L3VPNs and L2VPNs, over a common SRv6 underlay.
- Simplified OPEX: By providing a common end-to-end data plane, SRv6 simplifies operations, making it easier than managing multiple technologies.

Key concepts of SRv6

The key concepts of SRv6 are described in this table.

Table 3: SRv6 Concepts

SRv6 Concept	Description
Network program in the packet header	The network programming model refers to the approach of encoding network operations and services directly into IPv6 headers using the concept of segments.
Segment Routing Header (SRH):	Encodes an ordered list of instructions (segments) within each packet.
Segment Identifiers (SIDs)	28-bit values that represent routing instructions, functions, or paths.
SRv6 node roles, on page 8	Nodes in an SRv6 network perform specific functions as source, transit, or endpoint nodes based on their position and SID processing requirements.

Network program in the packet header

The network programming model in SRv6 refers to the approach of encoding network operations and services directly into IPv6 headers using the concept of segments. SRv6 leverages the native IPv6 protocol to implement segment routing functionalities, allowing for flexible and scalable traffic engineering, and network programmability.

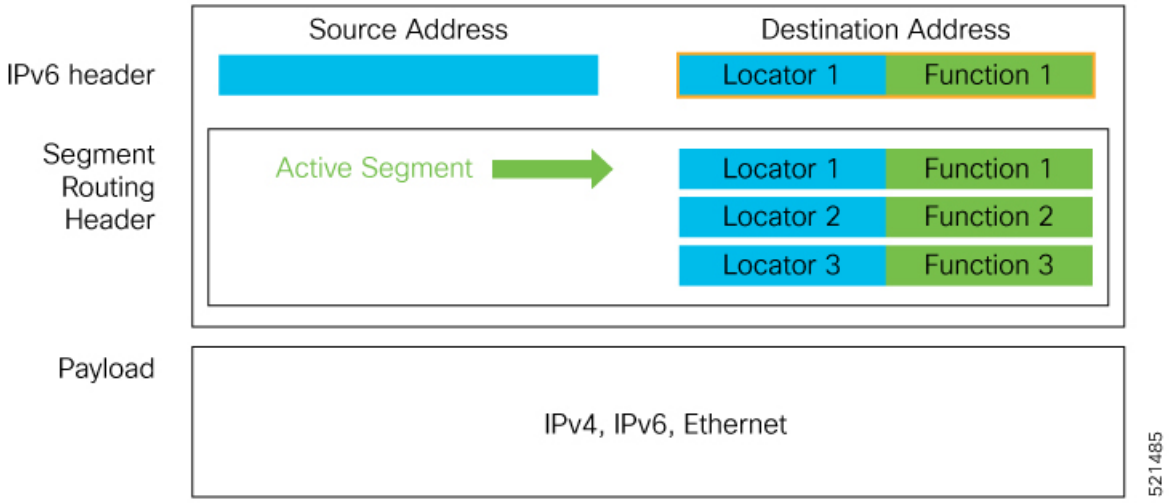
In SRv6, an IPv6 address represents an instruction. SRv6 uses a new type of IPv6 Routing Extension Header, called the Segment Routing Header (SRH), in order to encode an ordered list of instructions. The active segment is indicated by the destination address of the packet, and the next segment is indicated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

An packet header includes these components:

- IPv6 header: This is the standard IPv6 header, which includes the source address of the packet and destination address.
- Destination address: In a standard IPv6 header, the destination address contains a locator and a function.
- Segment Routing Header (SRH): SRH contains an ordered list of segments that the packet must traverse and header is inserted between the IPv6 header and the payload. SRH includes these components:
  - Active segment: Indicates the current segment being processed.
  - Locator and Function: Each segment consists of a locator and a function. The locator identifies a network node or a set of nodes, while the function specifies the action to be performed at that node.
- Payload: This section contains the actual data being transmitted. It can encapsulate various protocols, such as IPv4, IPv6, or Ethernet.

This figure illustrates the structure of a network program within the SRv6 packet header. In the example, there are multiple segments, each with its locator (Locator 1, Locator 2, Locator 3) and function (Function 1, Function 2, Function 3). The number of segments is determined by the Segments Left field in the SRH.

Figure 1: Packet Header



Segment Routing Header

Segment Routing Header (SRH) is an IPv6 routing extension header that

- embeds an ordered list of instructions, called segments, within each packet
- facilitates efficient routing, and network programmability, and
- encodes explicit paths using a list of IPv6 addresses.

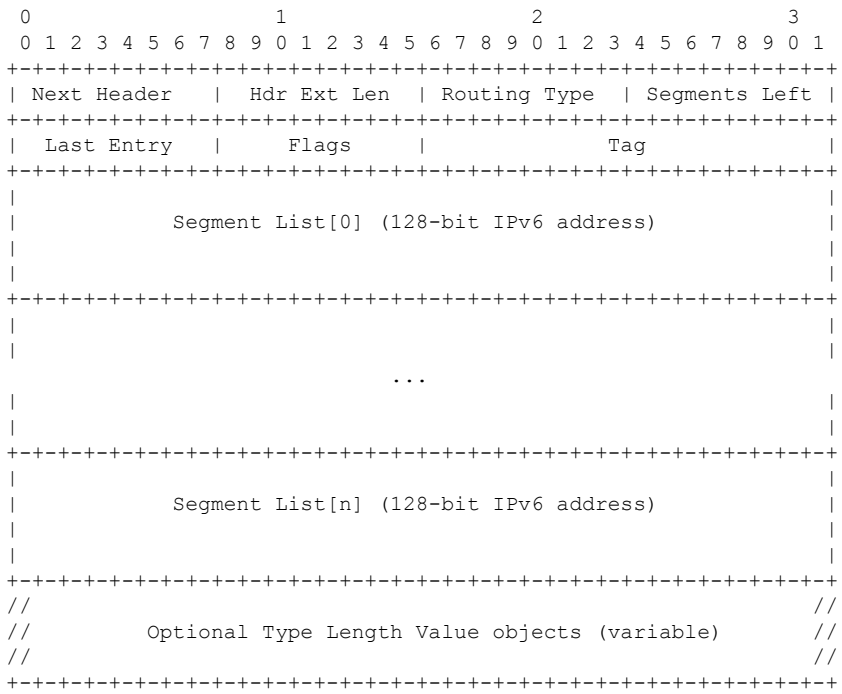
Segments are instructions represented as IPv6 addresses, that specify the sequence of nodes or functions a packet should traverse through a network using segment routing. For more information about SRH, see IETF RFC IPv6 Segment Routing Header (SRH).

Fields in the SRH

The SRH enables source routing in IPv6 by inserting a sequence of segments directly into packets. The active segment specifies the current destination address, while a pointer in the SRH identifies the next segment to be processed.

Field	Description
Next Header	Identifies the type of header immediately following the SRH.
Hdr Ext Len	The length of the SRH in 8-octet units, not including the first 8 octets.
Segments Left	Specifies the number of route segments remaining, i.e., 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	Tags a packet as part of a class or group of packets, such as packets sharing the same set of properties.
Segment List	A list of 128-bit IPv6 addresses representing the nth segment in the segment list. The encoding starts from the last segment of the SR policy: Segment List [0] contains the last segment, Segment List [1] contains the penultimate segment, and so on.

Format of SRH



Segment Identifiers

A Segment Identifier (SID) is a 128-bit value that identifies segments in an SRv6 network and consists of these three parts:

- Locator (LOC): This is the first part of the SID with most significant bits and represents an address of a specific SRv6 node.
  - SID Block (B): The network designator, representing the SRv6 domain or a known address space.
  - Node ID (N): The specific identifier for the node within the SRv6 network.
- Function (FUNCT): The portion of the SID that specifies a local behavior, or network instruction, executed by the node identified by the Locator. The Function field is opaque and bound to the SID's behavior.
- Arguments (ARG): An optional field containing additional information needed for SID processing. ARG values should remain constant within a flow to maintain consistent Equal-Cost Multi-Path (ECMP) hashing and avoid packet reordering.

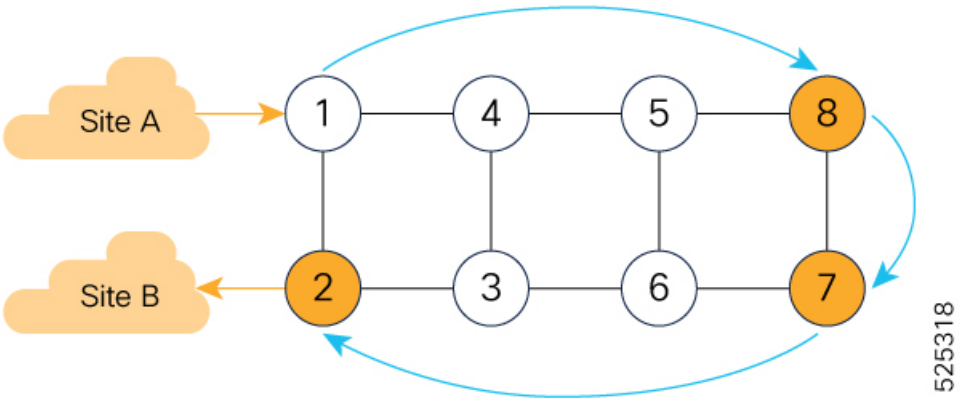
SRv6 node roles

In an SRv6 network, nodes along the packet path perform specific roles based on their position in the network and the SID processing requirements. Each node along the SRv6 packet path has a different function.

The figure visually represents the different node roles in an SRv6 network.

- Node 1 is the headend or the source node.
- Node 4 and 5 are the transit node.
- Node 8 is the endpoint node.

Figure 2: SRv6 Nodes



This table provides an overview of the roles and descriptions of various node types, highlighting their specific functions within an SRv6 network.

Table 4: SRv6 node roles

SRv6 node	Description
Source node or headend node	A node that can generate an IPv6 packet with an SRH, or an ingress node that can impose an SRH on an IPv6 packet. Also known as headend node. For more information, see SRv6 Head-End Behaviors.
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. A node where the SRv6 segment is terminated. For more information, see SRv6 endpoint behaviors, on page 11.

SRv6 headend behaviors

A segment routing headend behavior is a set of actions and mechanisms performed by a head-end router (or source node) that:

- implements segment routing policies to manage and direct traffic through a network
- initiates and enforces SR policies by defining the path traffic should take, and
- ensures efficient traffic forwarding by leveraging SRv6.

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
H.Encap.Red headend behavior	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) The H.Encap.Red is a headend behavior that **encapsulates** the original packet into a new IPv6 packet with a Segment Routing Header (SRH). By encapsulating with the SRH, you can control and manage the path the data packets take through a network. This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

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

SRv6 headend behavior variants

SRv6 supports these head-end behaviors:

- H.Encaps.Red—H.Encaps with reduced encapsulation
- H.Insert.Red—H.Insert with reduced insertion. Starting from Cisco IOS XR Release 24.3.1, the H.Insert.Red is supported.

Comparision between H.Encaps.Red and H.Insert.Red headend behaviors

Table 6: H.Encaps.Red versus H.Insert.Red headend 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 SRH. **Note** We support Starting from Cisco IOS XR Release 24.3.1, the H.Insert.Red.	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.

Headend behaviors	H.Encaps.Red	H.Insert.Red
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.
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.
References	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 a set of actions and mechanisms that:

- processes an SRv6 segment matching the network device's own address
- terminates the SRv6 segment at a node within the SRv6 domain, and
- executes the function bound to the SID.

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

List of SRv6 endpoint behaviors

These are 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). Starting from IOS XR Release Release 7.5.3, the End.DT46 endpoint is supported.
- 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

Segment Routing over IPv6 (SRv6) defines several endpoint behavior variants for the `End` and `End.X` behaviors, depending on how the Segment Routing Header (SRH) is processed. These variants can be supported individually or in combinations.

Table 7: Comparision of endpoint behavior variants

	Penultimate Segment Pop (PSP)	Ultimate Segment Pop (USP)	Ultimate Segment Decapsulation (USD)
Definition	Penultimate Segment Pop is a mechanism where the penultimate SR endpoint node processes the SRH by modifying the destination address and segments list.	Ultimate Segment Pop (USP) is a mechanism where the ultimate (last) SR Segment Endpoint Node removes the SRH from the IPv6 header when the `Segments Left` field equals 0.	Ultimate Segment Decapsulation (USD) is a mechanism that processes upper-layer headers by removing the outer IPv6 header and forwarding the inner packet to its destination.
Key operations	as part of the SID processing, copies the last SID from the SRH into the IPv6 Destination Address and decrements the segmentsl list value from one to zero.	If Segments Left is 0, then: • Updates the `Next Header` field in the preceding header to the next header value of the SRH. • Decreases the IPv6 header Payload Length by 8*(Hdr Ext Len+1) • Removes the SRH from the IPv6 extension header chain • Proceeds to process the next header in the packet	See SRv6 endpoint behavior variants, on page 12 and SRv6 endpoint behavior variants, on page 12

	Penultimate Segment Pop (PSP)	Ultimate Segment Pop (USP)	Ultimate Segment Decapsulation (USD)
Where performed	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.	USP is performed only at the ultimate SR Segment Endpoint Node. riggered when the `Segments Left` field equals 0.	USD is performed at the last SR Segment Endpoint Node in the repair path.
Applications	PSP is used when the Source SR Node instructs the penultimate SR Segment Endpoint Node to remove the SRH from the IPv6 header. The Source SR Node achieves this by using a PSP-flavored SID.	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.	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.

Table 8: Key operation for the end behavior of USD

Upper-layer header type is ...	then ...
41 (IPv6)	**1.** Removes the outer IPv6 Header with all its extension headers. **2.** Submits the packet to the egress IPv6 FIB lookup and transmission to the new destination.
4 (IPv4)	**1.** Removes the outer IPv6 Header with all its extension headers. **2.** Submits the packet to the egress IPv4 FIB lookup and transmission to the new destination.
else	process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

Table 9: Key operation for the End.X behavior of USD

Upper-layer header type is ...	then ...
41 (IPv6) or 4 (IPv4)	**1.** Removes the outer IPv6 header with all its extension headers. **2.** Forwards the exposed IP packet to the L3 adjacency J.
else	process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

Supported and unsupported features for SRv6

Supported hardware PIDs for SRv6

- SRv6 is supported on these 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
  - Cisco 8101-32-FH

Supported SRv6 features

- IGP redistribution/leaking between levels
- Prefix Summarization on ABR routers
- IS-IS TI-LFA
- Microloop Avoidance
- Flex-algo
- OAM: Ping and traceroute are supported.

Unsupported features for SRv6

- SRv6 is not supported on Q100-based line cards and fixed-port routers.
- SRv6 is supported on Cisco 8000 series routers used as core routers (P-role).
- SRv6 is not supported on Cisco 8000 series routers used as edge routers (PE-role). Services over SRv6 are not supported.
- SRv6 over GRE interface is not supported
- SRv6 over BVI interface is not supported
- Egress marking on the outer header during SRv6 encapsulation operations (TI-LFA) is not supported.

Techniques for effective SID management

SIDs are a fundamental building block of SRv6, enabling flexible and programmable routing by encoding instructions directly into IPv6 headers. As networks grow in complexity, efficient management of SIDs becomes critical to ensure scalability, reduce overhead, and optimize network operations.

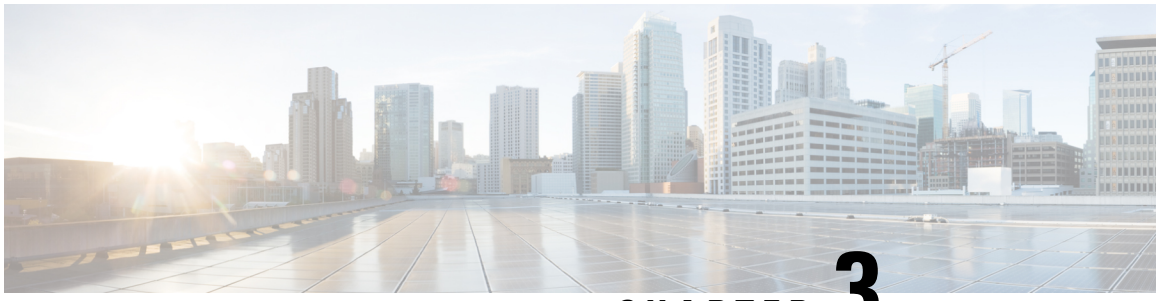
SRv6 employs specific techniques to effectively uses the 128-bit SID for optimized network operations. The main techniques are:

- Micro-SID (uSID): This technique divides a single 128-bit SID into smaller sub-SIDs, referred to as micro-segments. It allows multiple operations to be represented within a single SID, which results in reduced header overhead and improved scalability, particularly useful in environments with limited resources. For more information, see SRv6 Micro-segment (uSID) Overview, on page 17.
- Full-Length SID: This technique utilizes a full 128-bit IPv6 address to represent each segment, function, or instruction. This method ensures simplicity and aligns with standard IPv6 routing practices.

Difference between full-length SID and uSID

Table 10: Comparision of full-length SID and uSID

Feature	Full-Length SID	uSID
Definition	A full 128-bit IPv6 address used to represent a single segment, function, or instruction.	A single 128-bit SID divided into smaller sub-SIDs (micro-segments) to represent multiple operations.
Size	Each SID uses 128-bits in the SRH.	Multiple smaller sub-SIDs are packed into a single 128-bit SID.
Header size	Larger SRH due to 128-bits per SID.	Smaller SRH by encoding multiple operations into fewer SIDs.
Examples	Each SID is a full 128-bit IPv6 address. `SRH = [2001:db8:1::1, 2001:db8:2::1, 2001:db8:3::1]`	Multiple sub-SIDs are encoded in one 128-bit SID. `SRH = [2001:db8:1::1]`
Reference	SRv6 Micro-segment (uSID) Overview, on page 17	Configure Segment Routing over IPv6 (SRv6) with Full-Length SIDs



CHAPTER 3

SRv6 Micro-segment (uSID) Overview

Micro-segments offers a simplified approach to network segmentation. By leveraging the principles of Segment Identifiers (SIDs) and the network programming model of SRv6, micro-segments enable greater flexibility and scalability in modern networks.

This chapter explains the functionality of uSIDs, highlighting how they enhance the efficiency of SRv6 routing while reducing header overhead. It also explores their key benefits, limitations, and provides guidance for their implementation and configuration.

Note This chapter refers to SRv6 micro-segment as *uSID*.

SRv6 micro-segments

An SRv6 micro-segment (uSID) is an extension of the SRv6 architecture that:

- encodes up to six SRv6 segment (SID) instructions within a single 128-bit SID address, known as a uSID Carrier.
- leverages the existing SRv6 network programming architecture without changing the SRv6 data plane or control plane, and
- provides low MTU overhead and supports efficient hardware operation.

MTU overhead refers to the extra data, such as headers, metadata, or encapsulation, added to a packet on top of the actual payload. Low MTU overhead means the additional data is kept minimal to maximize the usable payload size within the MTU.

For more information about SRv6 uSID, see Network Programming extension: SRv6 uSID instruction and Compressed SRv6 Segment List Encoding in SRH .

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Micro-Segment (uSID)	Release 25.2.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100]) (select variants only*) *This feature is now supported on the Cisco 8404-SYS-D routers.
SRv6 Micro-Segment (uSID)	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100] , 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SRv6 Micro-Segment (uSID)	Release 7.3.1	This feature is an extension of the SRv6 architecture. It leverages the existing SRv6 Network Programming architecture to encode up to six SRv6 Micro-SID (uSID) instructions within a single 128-bit SID address. Such a SID address is called a uSID Carrier. In addition, this feature leverages the existing SRv6 data plane and control plane with no changes. It also provides low MTU overhead. For example, if there are 6 uSIDs per uSID carrier, this configuration results in 18 source-routing waypoints using only 40 bytes of overhead (in SRH).

Benefits of uSID

SRv6 uSID enhances the SRv6 framework by providing improved compatibility, scalability, efficiency, hardware efficency, and ease of deployment. Key benefits include:

Compatibility with SRv6 Framework

- Leverages the existing SRv6 network programming model without requiring changes. SRv6 uSID is a new pseudo code in the existing SRv6 network programming framework.
- Leverages the SRv6 data plane (SRH) with no change without any modifications, allowing any SID in the destination address or SRH to act as an SRv6 uSID carrier.

- Leverages the SRv6 control plane without any modifications.

Scalability

- Supports a highly scalable number of globally unique nodes within the domain. For instance:
  - 16-bit uSID ID size: 65k uSIDs per domain block
  - 32-bit uSID ID size: 4.3M uSIDs per domain block
- Enables scalable control plane through summarization at area or domain boundaries, allowing significant scaling without routing extensions.

Efficiency

SRv6 uSID achieves the lowest MTU overhead by supporting 6 uSIDs per uSID carrier, enabling efficient routing.

For example, it supports up to 18 source-routing waypoints with only 40 bytes of overhead.

- + H.Encaps.Red with an SRH of 40 bytes (8 fixed + 2 * 16 bytes). This capability is available starting from Cisco IOS XR Release 24.3.1.
- + 6 uSIDs in DA and 12 in SRH

Hardware efficency

- Leverages hardware capabilities, such as inline IP Destination Address editing and IP Destination Address longest match, to ensure seamless performance.
- Eliminates the need for additional lookups in indexed mapping tables, enhancing processing speed.
- Operates with legacy IP-in-IP encapsulation behavior for micro-programs with 6 or fewer uSIDs, simplifying hardware requirements.

Seamless deployment

- Allows a uSID to function as a SID, with the carrier holding a single uSID.
- Keeps the inner structure of an SR Policy opaque to the source.
- A carrier with uSIDs is just seen as a SID by the policy headend Security.

Security

- Leverages SRv6's native SR domain security.

Limitations for uSID

Ensure that you follow these limitations when working with SRv6 uSID behaviors:

Supported SRv6 uSID endpoint behaviors

The SRv6 network programming is extended with new types of SRv6 SID endpoint behaviors:

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

  Starting from Cisco IOS XR Release 7.5.3, End.DT46 endpoint is supported.
- uDX—uDX is the short notation for the NEXT-CSID End.DX behavior with the same pseudocode as End.DX4/End.DX6/End.DX2.

Supported SRv6 uSID headend behaviors

- H.Encap.Red (1 uSID carrier with up to 6 uSIDs)
- H.Insert.Red : Starting from IOS XR Release 24.3.1, H.Insert.Red is supported only on Cisco Silicon One P100-based routers.

Limitations for encapsulation capabilities and Parameters

Ensure that you adhere to these requirements when configuring IPv6 header fields for SRv6 encapsulated packets.

- Source address: Supports a single source address (SA) for SRv6 encapsulated packets. This source address is derived from the SRv6 global configuration or, if not configured, from the IPv6 Loopback address.
- Hop limit:
  - Do not manually configure the hop-limit value in the outer IPv6 header, as this action is not supported.
  - Overlay encapsulation does not propagate the hop limit by default. Use the **hop-limit propagate** command to enable propagation from the inner header to the outer header.
  - Underlay encapsulation (TI-LFA) behavior is always in propagate mode, regardless of the CLI.
- **Traffic-class**:
  - By default, traffic-class propagation is disabled for overlay encapsulation unless you enable it with the **traffic-class propagate** command, which allows propagation from the inner header to the outer header.
  - Manual configuration of the traffic-class value in the outer IPv6 header is not supported.
  - Underlay encapsulation (TI-LFA) always uses propagate mode, regardless of the CLI configuration.
- **Flow Label**:
  - Cisco 8000 series routers use the flow-label from the incoming IPv6 header. During 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.

- Underlay H-Encapsulation (P Role): A maximum of 6 SIDs (Segment Identifiers) per carrier is used for SRH underlay encapsulation. This applies to P devices responsible for maintaining the underlay transport.
- Underlay H-Insertion (PE Role):A maximum of 3 SIDs (1 carrier with 3 SIDs per carrier) is used for SRH underlay insertion. This applies to PE devices integrating underlay routing.
- Overlay H-Encapsulation (PE Role): A maximum of 3 SIDs (1 carrier with 3 SIDs per carrier) is used for SRH overlay encapsulation. This applies to PE devices managing overlay routing.

SRv6 uSID allocation within a uSID block

The SRv6 uSID allocation within a uSID block is a mechanism that

- involves the assignment of micro-segment identifiers (uSIDs) from a defined IPv6 prefix block and
- supports both global and local ID allocation for diverse routing and service requirements.

Table 12: Feature History Table

Feature Name	Release	Description
Wide LIB uSID Allocation for End.DT46 SRv6 SIDs	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is now supported on: • 8011-4G24Y4H-I
Wide LIB uSID Allocation for End.DT46 SRv6 SIDs	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) * This feature support is now extended to the Cisco 8712-MOD-M routers.
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.

uSID allocation blocks

In the Segment Routing (SR) domain, uSID allocation is categorized into three types of ID blocks. Each serves a specific purpose and is defined by its scope and behavior.

- Global ID Block (GIB)
- Local ID Block (LIB)
- Wide Local ID Block (W-LIB)

Table 13: Comparison of uSID allocation blocks

Attributes	Global ID Block (GIB)	Local ID Block (LIB)	Wide LIB (W-LIB)
Description	Set of IDs for globally scoped uSID allocation, providing reachability to a node and identifying shortest paths	Set of IDs for locally scoped uSID allocation, tied to local (endpoint) behavior, not independently routable.	Extended set of IDs for local uSID allocation, supporting nodes with large-scale requirements
Key Characteristics	Provides shortest path to a node in the SR domain. Advertised via an IP route (e.g., `/48`). Parent node executes a variant of END behavior. Supports Anycast uSIDs.	Must be preceded by a globally scoped uSID. Identifies a local micro-instruction (e.g., cross-connect or VPN context). Not routable. Same locally scoped uSID can differ between nodes	Provides more local uSIDs than the standard LIB. Useful for nodes with large-scale Pseudowire termination
Examples	Nodal uSID (uN) is a globally scoped behavior. Multiple nodes may share the same globally scoped uSID for Anycast.	Locally scoped uSID **L** may bind two different behaviors on nodes **N1** and **N2**. Used for local endpoint-specific actions.	A PE node that requires more local uSIDs for extensive Pseudowire termination.

uSID allocation example

This section explains how locally scoped and globally scoped uSIDs are allocated, provides an example with a specific uSID Locator Block, and illustrates the allocation scheme. We also explore how global and local uSIDs are assigned for a node within an SRv6 domain.

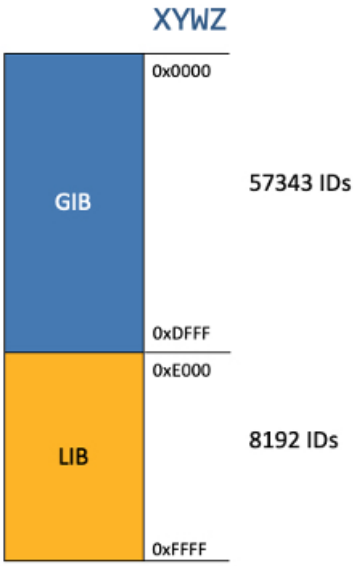
The request to allocate locally scoped uSIDs comes from SRv6 clients (such as IS-IS or BGP). The request can be to allocate any available ID (dynamic allocation) or to allocate a specific ID (explicit allocation).

- **uSID Allocation request source**: 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).
- **Example parameters for uSID Allocation:**
  - 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)

- **Allocation Scheme:** 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)

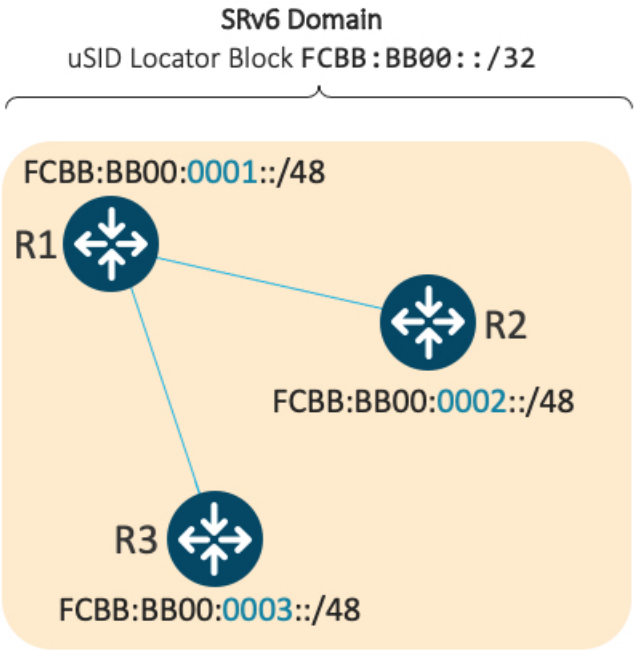
Figure 3: Allocation Scheme



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, this image depicts the global uSIDs allocated for 3 nodes within the SRv6 domain.

Figure 4: Global uSIDs allocated for 3 nodes



Examining **R1** in more detail, this node has **Local uSIDs** that are associated with **uA end-point behaviors** as follows:

- 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

Limitations for uSID allocation

Cisco IOS XR supports uSID allocation using GIB, LIB, and W-LIB. The supported features, ID ranges, and endpoint behaviors depend on the software release version.

Supported GIB and LIB features and ranges in Cisco IOS XR Release 7.5.3 and later

New functionalities added:

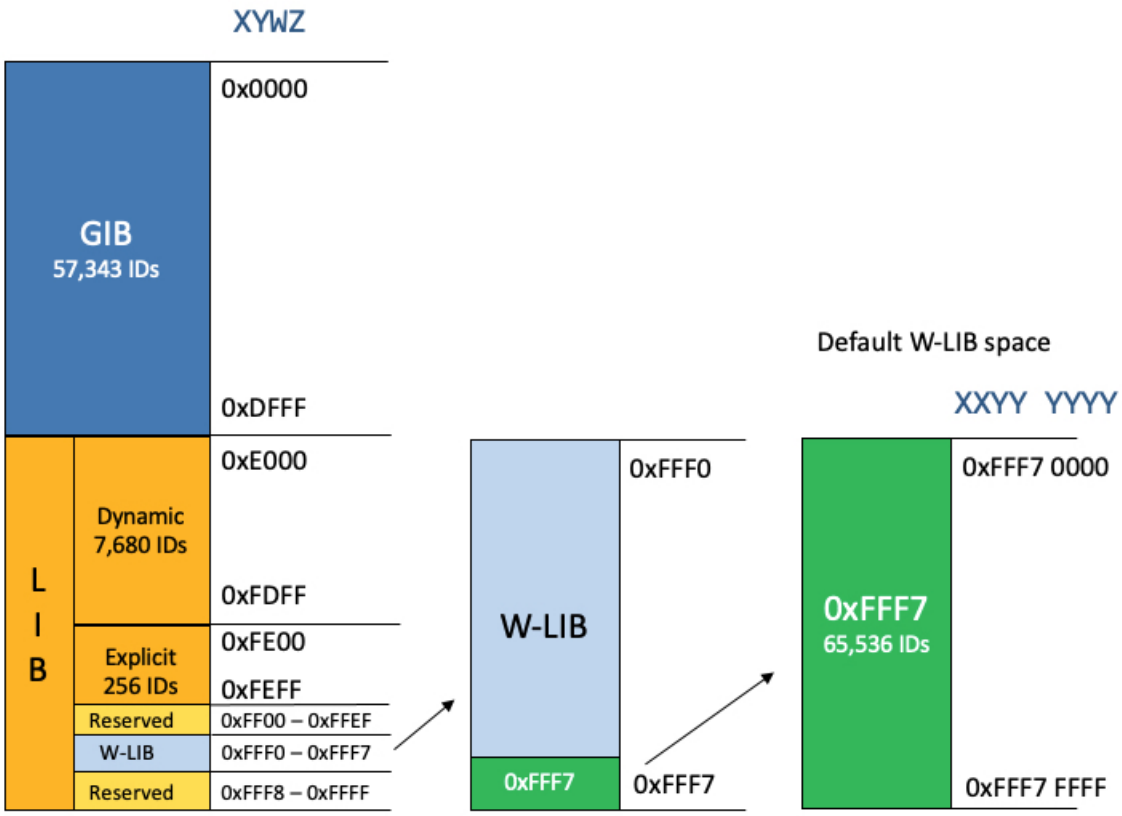
- Configurable explicit LIB range.
- Assignment of explicit LIB for user-assigned IDs of local segments.
- Manual uDT46 allocation from explicit LIB.
- Support for Wide LIB (W-LIB)

- Configurable explicit W-LIB range
- Explicit W-LIB allocation for user-assigned IDs of local segments.
- Manual uDT46 from explicit W-LIB

These are the supported range of IDs:

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

Figure 5: GIB/LIB/W-LIB



Supported GIB and LIB features and ranges in Cisco IOS XR Release 7.5.2 and earlier

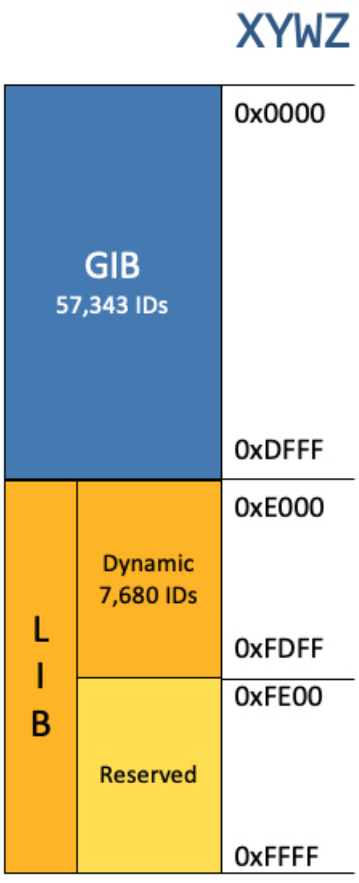
New functionalities added:

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

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 6: GIB/LIB



uSID allocation recommendations

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:

- Use a /24 subnet 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)

How SRv6 uSIDs are allocated

The process describes how SRv6 uSID-based VPN and traffic engineering operate to enable traffic forwarding between two VPNv4 sites (Site A and Site B) over an SRv6 domain with a traffic-engineered path.

Summary

The key components involved in the process are:

- Ingress PE node (Node 1): Encapsulates IPv4 packets from Site A into IPv6 packets with SRv6 uSID instructions
- Egress PE node (Node 2): Decapsulates IPv6 packets and performs IPv4 table lookup to deliver packets to Site B.
- SRv6 capable nodes (Nodes 8 and 7): Execute SRv6 uSID-based instructions to forward packets along the traffic-engineered path. The nodes are configured with 32-bit SRv6 block = fcbb:bb01 and 16-bit SRv6 ID.

  For example:
  - Node 7 uN = fcbb:bb01:0700::/48
  - Node 8 uN = fcbb:bb01:0800::/48
- Classic IPv6 nodes (Nodes 3, 4, 5, and 6): Forward packets using standard IPv6 shortest-path forwarding without modifying the outer destination address (DA).

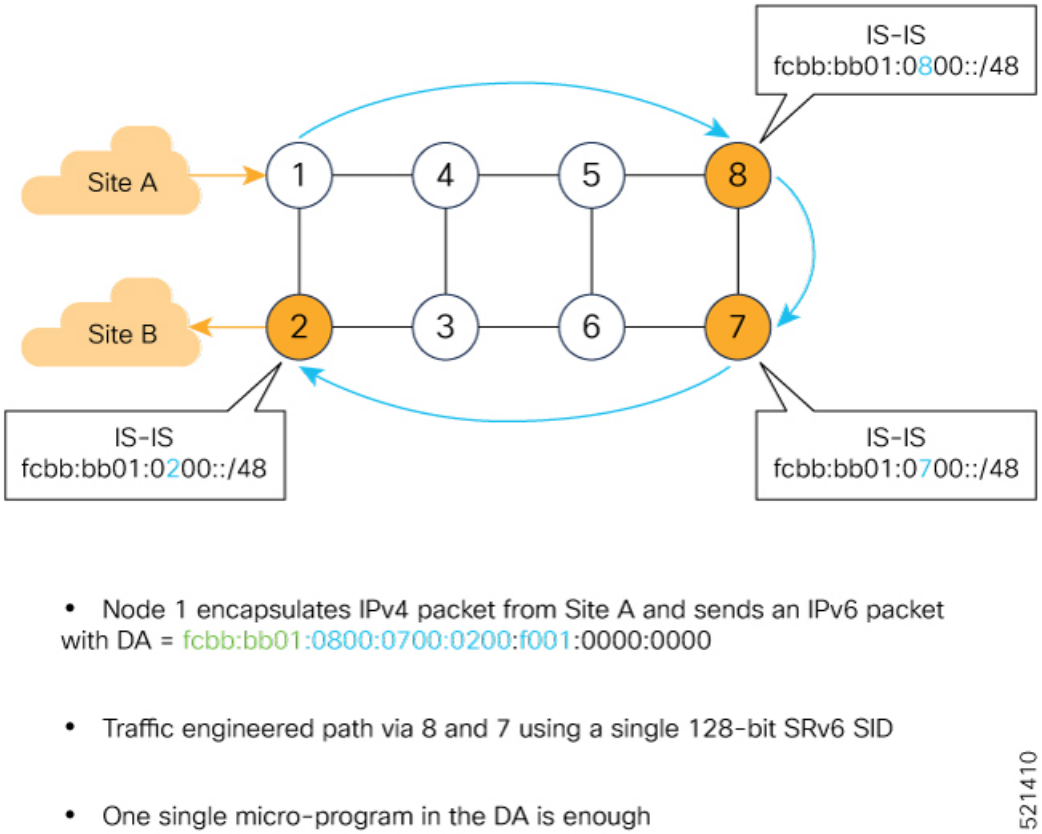
These IGP routes are advertised:

- 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

Workflow

Figure 7: Integrated VPN and traffic engineering SRv6 uSID usecase



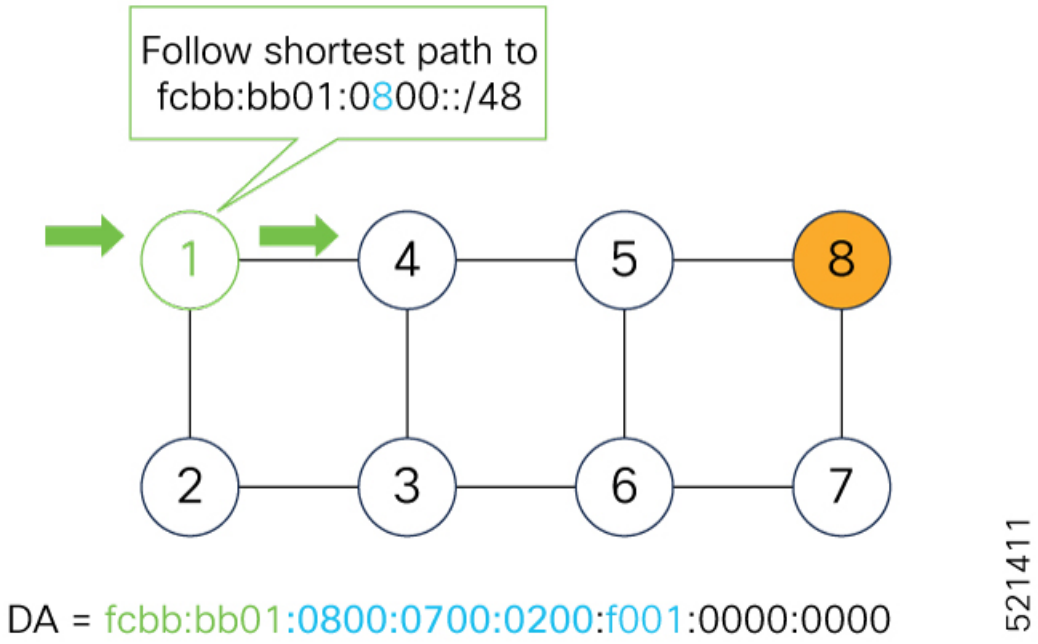
These stages describe the process of SRv6 uSID allocation:

1. Packet Encapsulation (Node 1 – Ingress PE):
   - Node 1 receives an IPv4 packet from VPNv4 Site A.
   - Encapsulates the IPv4 packet into an IPv6 packet with the destination address set to `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).

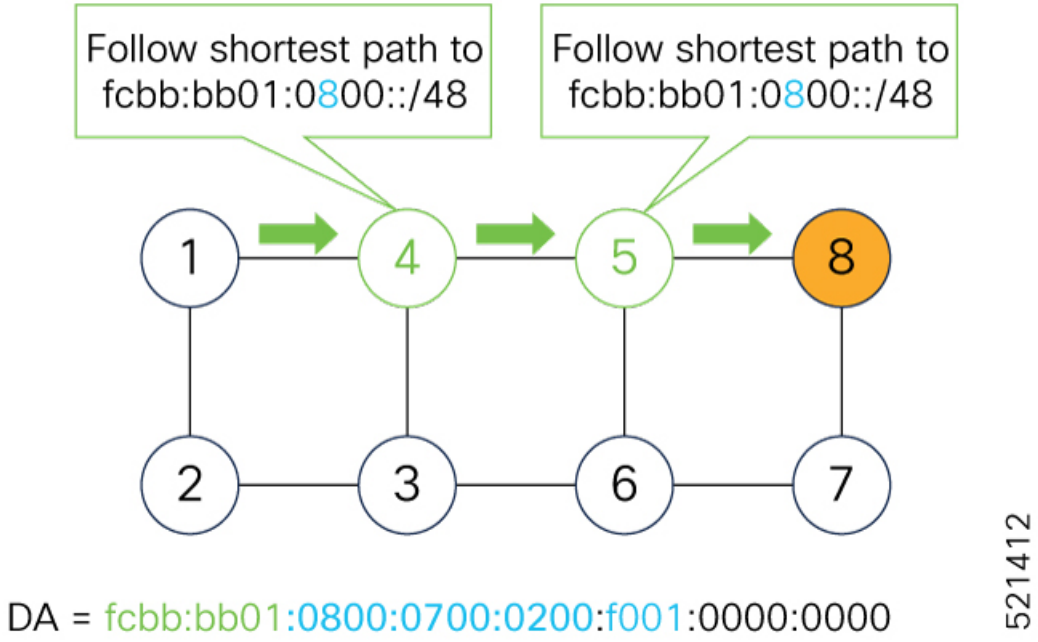
     uSIDs (uNs) 0800, 0700, 0200 are used to realize the traffic engineering path to Node 2 with way points at Nodes 8 and 7. uSID f001 is the BGP-signalled instruction (uDT4) advertized by Node 2 for the VPNv4 service

Figure 8: Node 1: End.B6.Encaps Behavior



2. Packet forwarding through classic IPv6 nodes (Nodes 4 and 5): Nodes 4 and 5 simply forward the packet along the shortest path to Node 8, providing seamless deployment through classic IPv6 nodes.

Figure 9: Node 4 and Node 5: Classic IPv6 Nodes



3. Processing at Node 8 (SRv6 uN behavior): 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 performs these actions:

a. Pops its own uSID (0800)

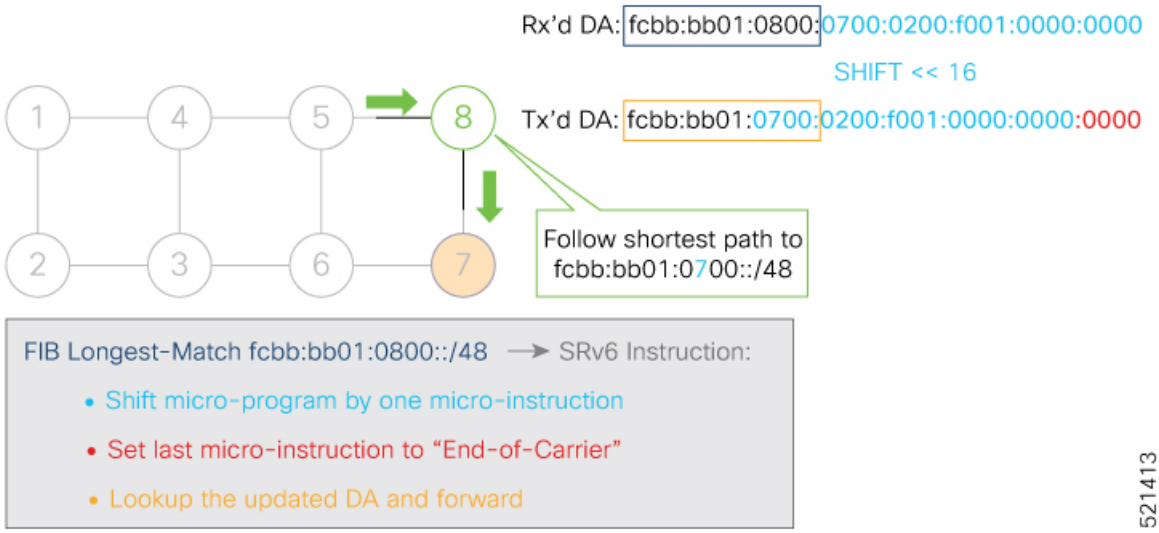
b. Shifts the remaining DA by 16-bits to the left

c. Fills the remaining bits with 0000 (End-of-Carrier)

d. Performs a lookup for the shortest path to the next DA (fcbb:bb01:**0700::/48**)

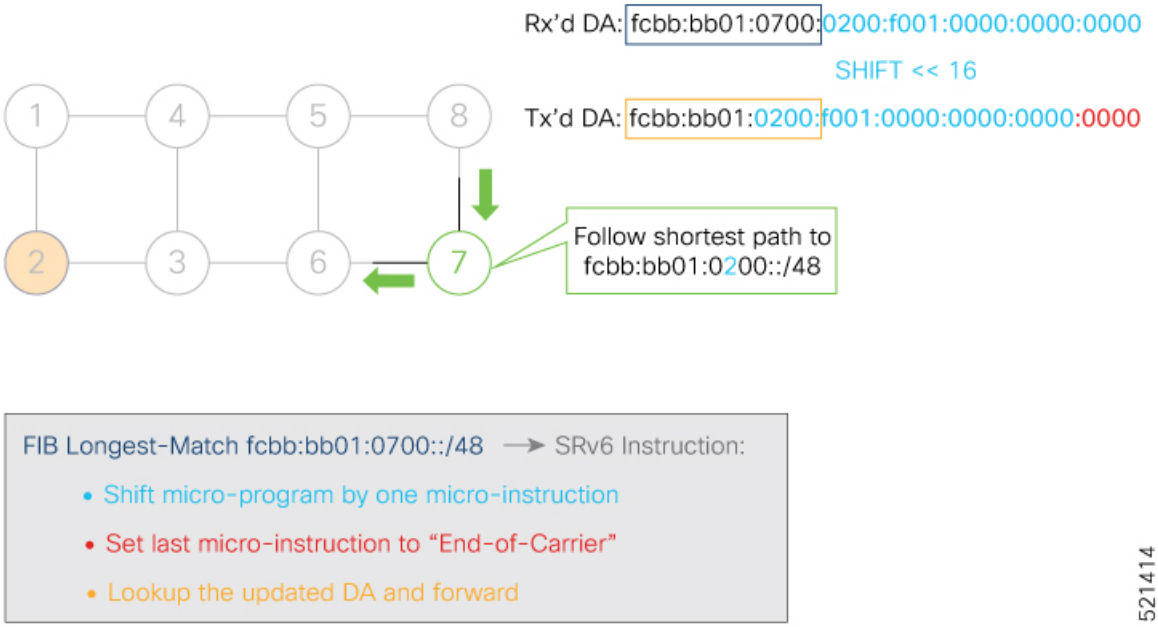
e. Forwards it using the new DA fcbb:bb01:**0700**:0200:f001:0000:0000:0000

Figure 10: Node 8: SRv6 uN Behavior (Shift and Forward)



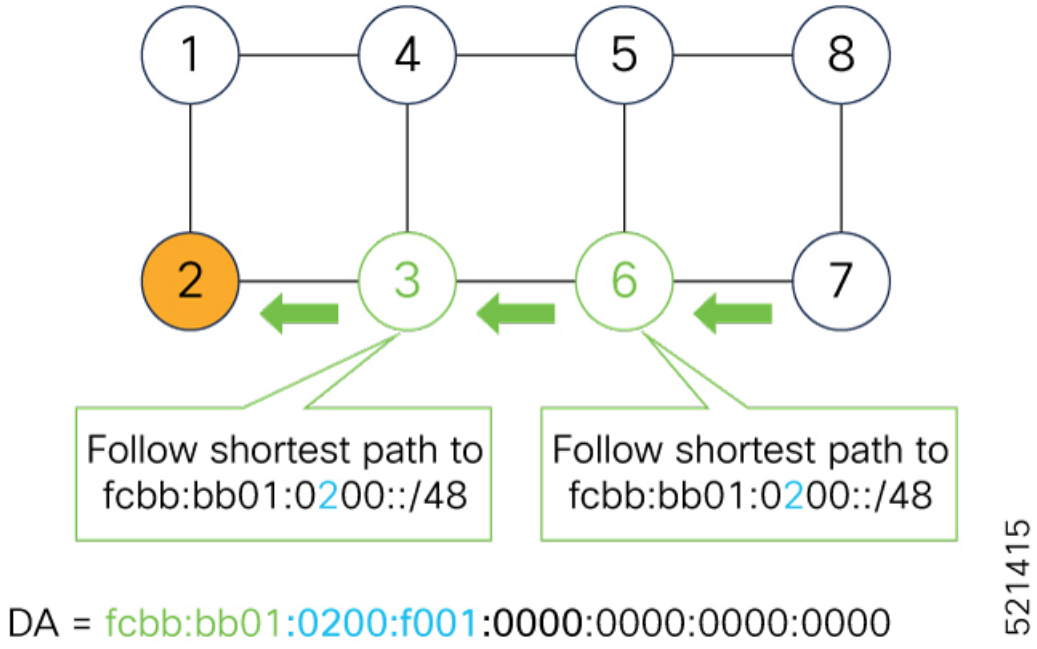
4. Processing at Node 7 (SRv6 uN behavior): When Node 7 receives the packet, it performs the same SRv6 uN behavior (shift-and-lookup with PSP/USD), forwarding it using the new DA fcbb:bb01:**0200**:f001:0000:0000:0000:0000

Figure 11: Node 7: SRv6 uN Behavior (Shift and Forward)



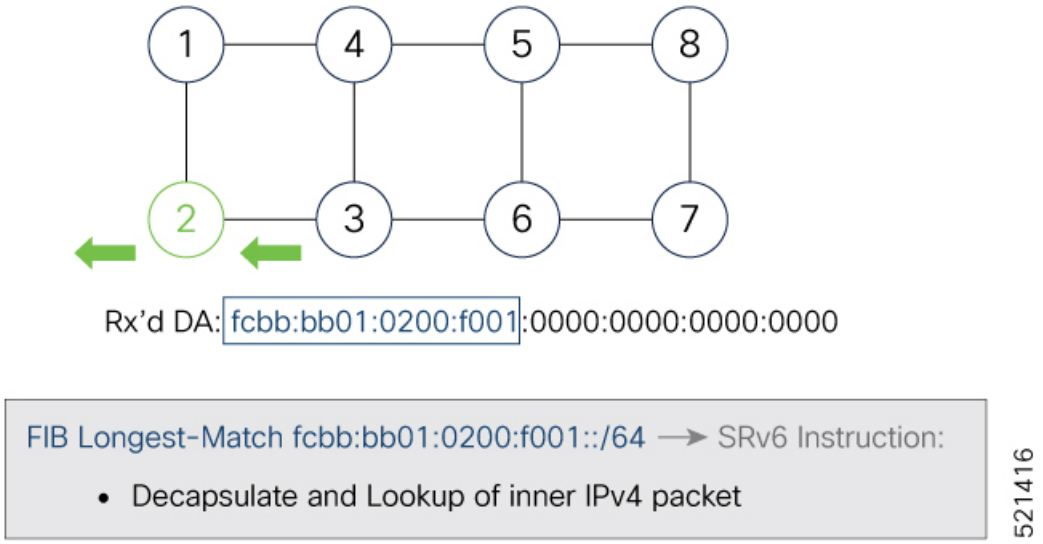
5. Packet forwarding through classic IPv6 nodes (Nodes 6 and 3): Nodes 6 and 3 simply forward the packet along the shortest path to Node 2, providing seamless deployment through classic IPv6 nodes.

Figure 12: Node 6 and Node 3: Classic IPv6 Nodes



6. Packet decapsulation (Node 2 – Egress PE): 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 13: Node 2: SRv6 uDT4 Behavior



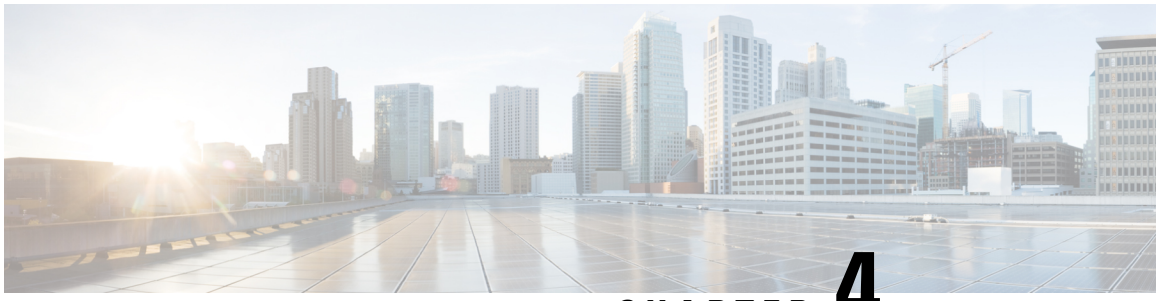
SRv6 uSID features

This table outlines key SRv6 features and describes how uSIDs are utilized within each to optimize network operations and improve performance.

Table 14:

Supported uSID features	Description
SRv6 Locators, on page 35	SRv6 locators are fundamental IPv6 address prefixes that define the address space for Segment Identifiers (SIDs) within an SRv6 network. They are crucial for enabling efficient routing and precise traffic steering.
Implementing SRv6 Flexible Algorithms with IS-IS, on page 41	SRv6 flexible algorithms (Flex-Algo) empower network operators to define routing behaviors tailored to specific operational requirements. Flex-Algo enables precise path computation beyond traditional shortest-path routing, allowing for customized traffic engineering, network slicing, and transport SLA assurance.
Path Computation Element Protocol, on page 57	Path Computation Element Protocol (PCEP) facilitates centralized management and dynamic optimization of SR paths. PCEP enables Path Computation Clients (PCCs) to communicate with Path Computation Elements (PCEs) for path computation, delegation, and real-time adjustments. It is used to enhance network efficiency, flexibility, and resilience by supporting advanced traffic engineering constraints and ensuring secure control plane communications.

Supported uSID features	Description
Traffic Engineering in IPv6 Networks Using SRv6-TE , on page 75	Segment Routing over IPv6 traffic engineering (SRv6-TE) provides granular control and flexibility in managing network traffic. SRv6-TE enables administrators to steer traffic across IPv6 networks according to specific policies and requirements, facilitating explicit path creation for applications demanding particular QoS levels. It is used to enhance scalability, reduce complexity, optimize resource utilization, and integrate seamlessly with existing IPv6 infrastructure.
SRv6 Layer 3 VPN Services and Global Routing, on page 103	SRv6 enables scalable, flexible, and programmable Layer 3 VPN (L3VPN) and global routing services, which are essential for modern network traffic engineering and service delivery.
Advanced SRv6 Layer 3 features, on page 145	Advanced SRv6 Layer 3 features provide the agility and efficiency required by modern networks. Specialized mechanisms offer advanced functionalities to optimize resource utilization and streamline traffic management across complex network segments. These capabilities are used to enhance network visibility and diagnostics, supporting adaptable, resilient, and future-proof deployments.
SRv6-Based Layer 2 and Integrated VPN Services, on page 171	SRv6-based Layer 2 and integrated VPN services provide flexible, scalable, and resilient solutions for modern network demands.
SRv6-MPLS L3 Service Interworking Gateways, on page 197	When bridging SRv6 and MPLS domains, the SRv6 side will often use uSIDs for its service and transport SIDs. The interworking gateway needs to correctly interpret and translate these uSIDs to their MPLS equivalents or other SRv6 formats.
SRv6 Network Performance Measurement , on page 219	Tools like liveness monitoring, delay measurement, and path tracing operate by sending probes along SRv6 paths. If these paths are constructed using uSIDs, the performance measurement mechanisms must be designed to correctly interpret and traverse these uSID-based paths to gather accurate metrics.
SRv6 Traffic Accounting, on page 263	Traffic accounting tracks data flow over SRv6 paths. If the network utilizes uSIDs for its segment routing, the accounting system needs to correctly attribute traffic to these uSIDs and their associated locators to provide granular insights into resource consumption and traffic patterns.



CHAPTER 4

Enable Traffic Steering with SRv6 Locators

This chapter focuses on SRv6 locators, which are fundamental IPv6 address prefixes used to identify blocks of Segment Identifiers within an SRv6 network. It examines the structure and various types of locators, including Base Format, uSID, and Anycast locators, and discusses their specific applications and limitations.

SRv6 Locators

An SRv6 locator is a unique IPv6 address prefix that

- identifies a block of Segment Identifiers (SIDs) in an SRv6-enabled network
- belongs to a specific node in an SRv6-enabled network, and
- enables routing and forwarding based on SRv6 SIDs.

A SID is a 128-bit value in SRv6 networks that represents a specific instruction or path in the network. For more information, see Segment Identifiers, on page 8.

Structure of a SRv6 locator

The structure of an SRv6 locator consists of the most significant bits of the SID and serves as the address of a specific SRv6 node. The locator can be further divided into:

- SID block: The most significant portion that designates the SRv6 domain.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.

Types of locators

SRv6 supports several locator types to identify nodes and services within an SRv6 domain. Each locator type addresses specific requirements in network design and operation. The main SRv6 locator types are Base Format locator (F1), uSID locator (F3216), and Anycast locator.

Table 15: Comparison of SRv6 locator types

Feature	Base Format Locator (F1)	uSID Locator (F3216)	Anycast Locator
Definition	The Base Format locator is a standard SRv6 locator that identifies a specific router or node within the SRv6 domain. It uses a prefix to indicate the node and a function identifier (Func ID) to specify what action the node should perform.	The uSID locator is an advanced SRv6 locator that compresses multiple SRv6 segments into a single IPv6 address.	An anycast locator is a shared locator that identifies a group of nodes (Anycast group). All nodes in the group advertise the same locator, allowing traffic to be routed to the nearest node.
Components	Locator prefix and Function identifier (Func ID)	Locator prefix Micro-SID stack (uSIDs)	Anycast locator Prefix Function identifier (Func ID)
Format	`<Locator Prefix>::<Function ID>`	`<Locator Prefix>::<uSID1>:<uSID2>:<uSID3>`	`<Anycast Locator Prefix>::<Function ID>`
Example	`2001:db8:1::1` Prefix: 2001:db8:1::/48 Func ID: 1	`2001:db8:2::100:200:300` Prefix: 2001:db8:2::/48 uSIDs: 100, 200, 300	`2001:db8:3::2` Prefix: 2001:db8:3::/48 advertised by multiple nodes Func ID: 2
Usecase	Node Identification Simple SRv6 Deployments	Efficient Path Encoding Service Chaining	Redundancy and Load Balancing Exit Points in SRv6 Domains

Limitations for locators

Limitations for uSID locators

- Cisco IOS XR supports uSIDs with 32-bit uSID blocks and 16-bit uSID IDs (3216). You must use a single UCF format for uSID locators in an SRv6 uSID domain.
- Cisco IOS XR supports up to 16 uSID locator prefixes. Configure multiple locator prefixes when using anycast locators or SRv6 Flexible Algorithm instances.
- Cisco IOS XR supports uSID locator prefixes from different uSID blocks. You can configure up to 256 uSID blocks across all uSID locators in the network.

Limitations for anycast locators

- ISIS: Unlike a normal locator, IS-IS does not program or advertise uA SIDs associated with an anycast locator.
- TI-LFA backup: 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.
- Flexible algorithm: SRv6 anycast locators may have non-zero algorithm (Flexible Algorithm) values.

Configure SRv6 locators

Define and configure SRv6 locators, which represent IPv6 address spaces and behaviors used for SRv6. A locator configuration is essential for enabling SRv6 routing and forwarding within your network.

Follow these steps to configure SRv6 locators:

Procedure

Step 1 Enable SRv6 and configure a locator globally.

Example:

```
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
```

You need to configure the locator prefix value, specify the locator as a micro-segment (uSID) locator and specify that IGP underlay uSID (uN/uA) variant is PSP-USD for this locator.

- To configure a locator with a flexible algorithm:

```
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
```

- To configure an 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
```

- To advertise anycast prefixes on an interface

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

- To configure an SRv6-TE locator and binding SID:

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

Step 2 (Optional) Customize SRv6 logging and SID allocation parameters.

Example:

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

Step 3 Verify overall SRv6 state and platform capabilities using SRv6 manager.

Example:

```
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#
```

Step 4 Verify the locator configuration and its operational status.

Example:

```
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)
```

Step 5 Verify local SID allocation from locators.

Displays allocation information for SIDs belonging to a locator

Example:

```
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
```

Step 6 View the detailed information for a specific SRv6 local SID.

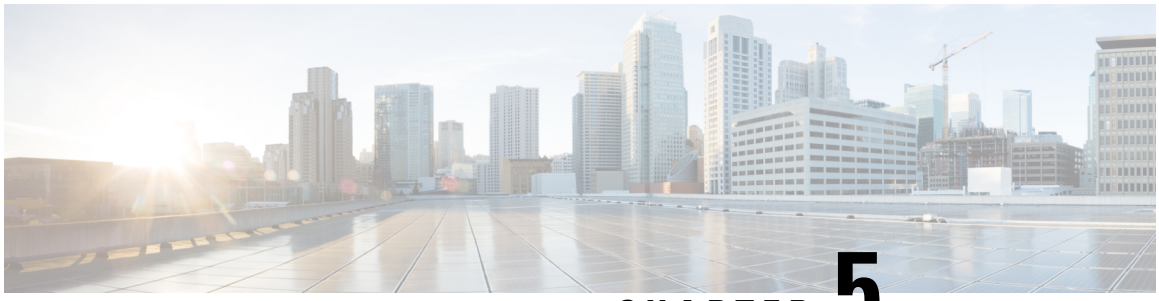
Example:

```
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)
```

What to do next

You can use these **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.



CHAPTER 5

Implementing SRv6 Flexible Algorithms with IS-IS

Today's evolving networking environment requires advanced routing solutions that offer greater customization, efficiency, and adaptability.

The SRv6 Flexible Algorithm (Flex-Algo) meets the demands by empowering network operators to define routing behaviors tailored to specific operational requirements, extending the capabilities of Segment Routing over IPv6 (SRv6), and integrating seamlessly with Interior Gateway Protocols (IGPs), like IS-IS. Flex-Algo enables precise path computation beyond traditional shortest-path routing.

This chapter explains the key concepts of the SRv6 Flexible Algorithm. It also provides guidance on how to configure it.

SRv6 Flexible Algorithms with IS-IS

Flex-Algo is an SRv6 feature that:

- enables custom path computation within an IGP such as IS-IS by associating specific metric types and constraints, rather than relying solely on the default IGP metric
- allows logical network segmentation by distributing multiple, parallel routing topologies through a standard IGP protocol such as IS-IS, and
- supports adaptive, traffic-engineered paths by associating prefix-SIDs with each algorithm to meet specific operational requirements.

You can identify each Flex-Algo by a numeric value from 128 to 255, and configure it through a Flex-Algo Definition.

Flex-Algo definition

A Flex-Algo definition (FAD) is a mechanism that combines a calculation type, a metric type, and a set of constraints. IS-IS advertises this definition using a Flex-Algo definition sub-TLV (type, length, value).

Flex-Algo definition Sub-TLV Format

The Flex-Algo definition Sub-TLV format includes these fields.

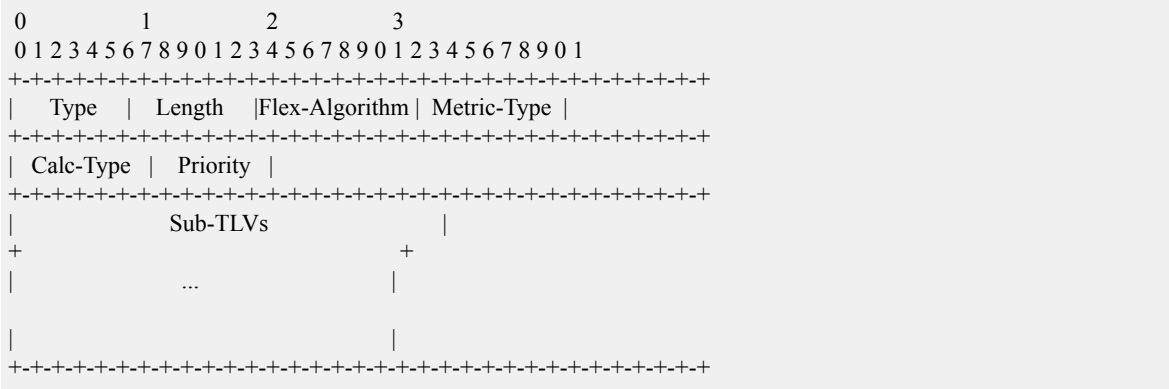


Table 16: Flex-Algo definition field description

Field	Description
Type	Specifies the sub-TLV type.
Length	Specifies the size of the data that follows the Type and Length fields, including any sub-TLVs. Varies based on the sub-TLVs included.
Flex-Algorithm	Represents the Flex-Algo number with a single-octet value ranging from 128 to 255.
Metric-type	Identifies the metric used for path calculation, such as IGP-metric or delay-metric.
Calc-type	Defines the calculation logic with a value from 0 to 127.
Priority	Indicates the advertisement priority with a value between 0 and 255.
Sub-TLVs	Optional; used to specify additional attributes and constraints.

Key concepts of SRv6 Flex-Algo

- Algorithm: An algorithm defines how IGP computes the best path. Routers advertise the support for the algorithm as a node capability. Routers also advertise prefix-SIDs with an algorithm value, so each SID is tightly coupled with its algorithm."An algorithm is a one octet value. You can reserve values from 128 to 255 for Flex-Algo representation.

- Metric types: Metric types are values used by routing protocols to evaluate and select the best network paths. They help determine the preferred route when multiple paths exist to a destination. Each metric type measures specific network characteristics, which may include IGP-metric and delay-metric.
- Constraints: Constraints are rules or conditions that affect which network paths are considered valid or preferred. If you set constraints, you can control route selection based on specific requirements or policies. Constraints can be set to include (require) or exclude (avoid) specific link-affinities in the path computation. Alternatively, constraints can instruct the routing algorithm to avoid using multiple links from the same SRLG.

Benefits of SRv6 Flex-Algo

Modern networks require advanced capabilities to compute optimal paths, incorporating diverse constraints and metrics beyond traditional shortest-path routing. Flex-Algo addresses these demands by allowing operators to define custom routing behaviors. All routers in a domain share a common understanding of these user-defined algorithm values. This consistency ensures reliable path computation and prevents traffic loops. This approach facilitates several key advantages:

- Custom path selection: Flex-Algo enables operators to define custom algorithms that consider factors beyond the traditional IGP shortest path. This includes advanced constraints such as plane selection in multi-plane networks, extended metrics (like delay), and specific link affinities (avoiding or preferring links with certain attributes or combinations of attributes).
- Granular traffic engineering and network slicing: Flex-Algo allows for precise traffic steering tailored to service requirements. Each network slice can be associated with a unique locator and a user-defined Flex-Algo instance, effectively distributing multiple, parallel routing topologies through a standard IGP protocol such as IS-IS.
- Transport SLA assurance: By incorporating specified metrics, Flex-Algo steers traffic over paths that meet specific transport Service Level Agreements (SLAs), such as minimum delay or minimum cost, rather than relying solely on default shortest path metrics.
- Simplified control plane operations: Flex-Algo reduces operational complexity by enabling direct traffic steering without the need for explicit Segment Routing Traffic Engineering (SR-TE) policies or route coloring.
- Enhanced forwarding efficiency: Flex-Algo improves forwarding efficiency by allowing packets to be steered directly along desired paths, optimizing network resource utilization.

Restrictions to configure SRv6 Flex-Algo with IS-IS

When configuring SRv6 Flex-Algo with IS-IS, ensure that you follow these restrictions:

- You can configure up to eight locators to support SRv6 Flex-Algo.
- The locator algorithm value must be in the range of 128 to 255.
- At least one router in the area, preferably two for redundancy, must advertise the Flex-Algo definition. Without the valid definition being advertised, the Flex-Algo will not function.

- To ensure loop-free forwarding for paths computed for a specific Flex-Algo, all routers in the network must share the same Flex-Algo definition.
- Flex-Algo paths to any prefix must be installed in forwarding using the Prefix-SID advertised for that Flex-Algo. If the Prefix-SID is unknown, the path will not be installed in forwarding for that prefix.

How Flex-Algo prefix-SIDs are advertised

Summary

The key components involved in the process are:

- Routers: Devices that compute paths for Flex-Algos and install IPv6 forwarding entries with SRv6-specific SIDs.
- Flex-Algo SIDs: IPv6 addresses advertised for prefixes to enable Flex-Algo-specific forwarding.
- Shortest path tree computation: A method that determines the optimal path for packet forwarding according to algorithm constraints.

The process of advertising Flex-Algo SIDs in SRv6 involves configuring routers to support and compute paths for Flex-Algo. The IS-IS protocol is used by network nodes to announce their support and advertise SRv6 locators, making all routers aware of available algorithms. Routers then compute paths based on Flex-Algo definitions and install corresponding IPv6 forwarding entries. This ensures that only advertised paths are used for forwarding within the network.

Workflow

These stages describe the process of advertising and forwarding using Flex-Algos in SRv6 with IS-IS:

1. SRv6 locators and slices: Each network slice is assigned a distinct SRv6 locator block. This segmentation enables specific traffic engineering and policy requirements per slice.
2. Advertising Flex-Algo participation: Network nodes announce their support for Flex-Algo instances by advertising their associated SRv6 locators via the IS-IS protocol. This process ensures all routers are aware of which algorithms are available and how they map to specific locators.
3. Shortest Path Tree Computation: Routers compute paths for each Flex-Algo by:
   - Pruning nodes that do not support the Flex-Algo
   - Excluding links with affinities that are not allowed by the algorithm definition
   - Using only links that advertise the metric specified in the Flex-Algo's definition.
4. Forwarding entry installation : Routers install IPv6 forwarding entries for Flex-Algo-computed paths using the advertised SRv6 Locator or End SIDs. If the SRv6 SID for a Flex-Algo is not advertised or known for a given prefix, the corresponding path cannot be installed in the forwarding table.

   This table shows how the system behaves under different conditions during the Flex-Algo advertisement and forwarding process:

When...	And...	Then...	And...
all routers are configured with the same Flex-Algo ID	the Flex-Algo definition is consistently advertised,	all routers can compute and agree on Flex-Algo paths	loop-free and consistent routing is ensured.
a Prefix-SID is advertised for a prefix and Flex-Algo	the prefix meets all Flex-Algo constraints,	the prefix is included in Flex-Algo-specific forwarding	traffic is forwarded along Flex-Algo-computed paths.
a router does not receive a Flex-Algo definition	or the definition is inconsistent	the router does not compute or install Flex-Algo paths	those paths are excluded from forwarding.
a link is marked with an excluded affinity	the Flex-Algo definition excludes that affinity,	the link is pruned from Flex-Algo path computation	traffic is automatically rerouted over eligible links.
A prefix-SID is not advertised for a prefix,	--	Flex-Algo forwarding entry for that prefix is not installed	prefix follows default IGP path instead.

Configure SRv6 Flex-Algo with IS-IS

This task involves configuring SRv6 locators and integrating them with IS-IS using Flex-Algo to optimize routing decisions based on various metrics.

Follow these steps to configure SRv6 Flex-Algo with IS-IS:

Procedure

Step 1 Configure SRv6 locators on the router.

Each locator is assigned a name, a prefix, and a micro-segment behavior. Optionally, you can associate a locator with a specific flexible algorithm.

In this example, two locators are created. The first locator uses Algo 0 (best-effort), and the second uses Algo 128 (low-latency).

Example:

```
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
```

Step 2 Assign SRv6 locators to IS-IS on the router to advertise the defined SRv6 locator prefixes throughout the network

Example:

```
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
```

Step 3 Define the Flex-Algo with the desired metric on the router to enable path computation.

Example:

```
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 HundredGigE0/0/0/0
Router(config-isis-if)# address-family ipv6 unicast
```

Step 4 Configure the delay probe on the interface on the router to measure and monitor network latency.

Example:

```
Router(config)# performance-measurement
Router(config-perf-meas)# interface HundredGigE0/0/0/0
Router(config-pm-intf)# delay-measurement
Router(config-pm-intf-dm)# commit
```

Step 5 Verify the SRv6 locators and their associated algorithms on the router to confirm their correct configuration and advertisement within the network.

Example:

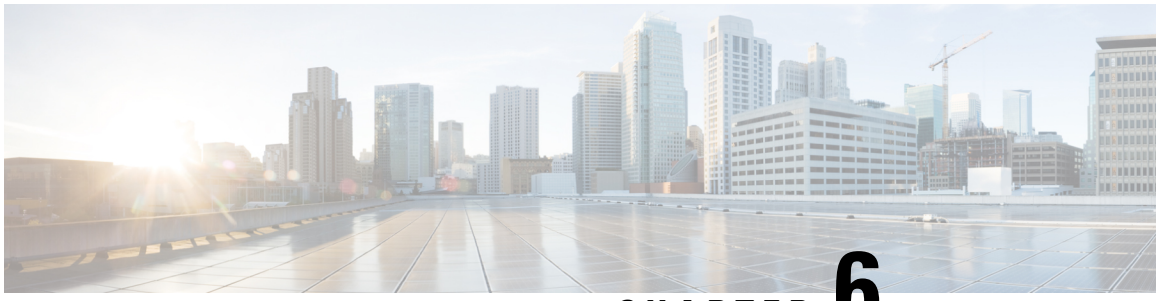
```
Router# show segment-routing srv6 locator

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
```

Step 6 Verify the IS-IS Flex-Algo configuration and status for Level-1 and Level-2.

Example:

```
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
```



CHAPTER 6

Implementing TI-LFA for SRv6

This chapter details Topology Independent Loop-Free Alternate (TI-LFA) for SRv6, a mechanism ensuring rapid network recovery and service continuity.

Topology-Independent Loop-Free Alternates for SRv6

A Topology-Independent Loop-Free Alternate (TI-LFA) is a network protection mechanism for SRv6 deployments that:

- provides link, node, and Shared Risk Link Groups (SRLG) protection
- ensures rapid failure repair using pre-calculated optimal SRv6 backup paths, and
- reduces packet loss during router convergence after a topology change.

Key concepts for SRv6 TI-LFA

- Optimal Repair Path (Post-Convergence Path): This is the path that traffic will eventually follow after the Interior Gateway Protocol (IGP) has fully converged following a failure. SRv6 TI-LFA's pre-calculated backup paths are designed to be these optimal post-convergence paths, ensuring seamless transition and minimal traffic oscillation. This approach is preferred for capacity planning, operational simplicity and reduced traffic transitions.
- TI-LFA protection types: SRv6 TI-LFA supports comprehensive protection against various failure modes:
  - Link protection: Excludes the failed link during backup path calculation.
  - Node protection: Excludes the neighbor node during backup path calculation.
- Shared Risk Link Group (SRLG): A Shared Risk Link Group (SRLG) is a situation in a network where links share a common fiber or physical attribute, these links have a shared risk, and if one fails, others in the group might also fail

Benefits of TI-LFA

- Overcoming traditional LFA limitations: SRv6 TI-LFA addresses the topology dependency and suboptimal path issues of classic LFA, and the incomplete coverage and operational complexity of remote LFA, providing robust protection where older methods fall short in SRv6 networks.
- Maximized network uptime and availability: SRv6 TI-LFA's sub-50ms failure repair drastically reduces the duration of service interruptions and minimizes downtime, leading to near-instantaneous recovery. Its comprehensive protection against link, node, and SRLG failures ensures network-wide resilience in SRv6 infrastructures.
- Reduced packet loss: By pre-calculating and immediately activating loop-free SRv6 backup paths, SRv6 TI-LFA ensures that traffic is quickly steered around failures, preventing packet drops during router convergence.
- Optimized network performance: The SRv6 backup paths are pre-computed to align with the post-convergence path, meaning traffic is routed efficiently even during a failure, avoiding suboptimal detours. This leads to greater stability and less oscillation in the network, as traffic only shifts once.
- Simplified network operations and management: SRv6 TI-LFA streamlines fast reroute configuration and management compared to older methods, reducing operational complexity. It automates best-path selection, eliminating the need for manual intervention or case-by-case adjustments in SRv6 networks.
- Efficient resource utilization and capacity planning: Knowing that traffic will follow optimal paths even during failures allows for more accurate and efficient network capacity planning and resource allocation within SRv6 deployments.

Comparison with other Loop-Free Alternate (LFA) techniques for SRv6

The table below highlights the key differences between TI-LFA and other LFA techniques, specifically in the context of SRv6 deployments

Table 17: Comparison with other Loop-Free Alternate (LFA) techniques for SRv6

Feature	Classic Loop-Free Alternate (LFA)	Remote LFA (RLFA	Topology-Independent Loop-Free Alternate (TI-LFA) for SRv6
Topology dependency	Topology dependent	Extends coverage but remains topology dependent	Topology independent
Protection coverage	Cannot protect all destinations in all networks.	Extends coverage to 90-95% of destinations.	Provides link, node, and SRLG protection in any SRv6 topology.
Repair Path Optimality	May not always provide the optimal LFA	Does not always provide the most desired repair path.	Provides optimal repair paths (post-convergence paths) using SRv6 SIDs
Operational complexity	Simpler, but limited coverage	Adds operational complexity by requiring a targeted LDP session for LDP traffic	Maintains the simplicity of the IPFRR solution within SRv6

Feature	Classic Loop-Free Alternate (LFA)	Remote LFA (RLFA	Topology-Independent Loop-Free Alternate (TI-LFA) for SRv6
Mechanism	Relies on local topology for backup path calculation	Uses tunnels (for example, LDP, GRE) to reach a remote LFA.	Uses SRv6 Segment Identifiers (SIDs) to steer packets along a pre-calculated post-convergence path.

Usage Guidelines for SRv6 Ti-LFA

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.

Link Protection

- TI-LFA provides link protection by default. Additional tiebreaker configuration is required to enable node or SRLG protection.

How SRv6 TI-LFA works

SRv6 TI-LFA operates by proactively computing optimal, loop-free backup paths that traffic can immediately use upon a failure. This ensures rapid restoration of connectivity and minimal disruption, allowing the network's Interior Gateway Protocol (IGP) to converge to a new primary path without significant packet loss.

Summary

SRv6 TI-LFA is a mechanism that ensures rapid network connectivity restoration and minimal disruption by proactively computing and activating optimal, loop-free backup paths. It involves routers (acting as Points of Local Repair - PLRs) that pre-compute these backup paths for potential failures. Upon rapid failure detection (often via BFD), the PLR immediately activates the pre-computed backup path, encapsulating traffic with SRv6 Segment Identifiers (SIDs) to steer it around the failure. Concurrently, the Interior Gateway Protocol (IGP) re-converges the network, and traffic seamlessly transitions from the temporary SRv6 TI-LFA backup path to the newly established optimal primary paths.

Workflow

Figure 14: TI-LFA Repair Path

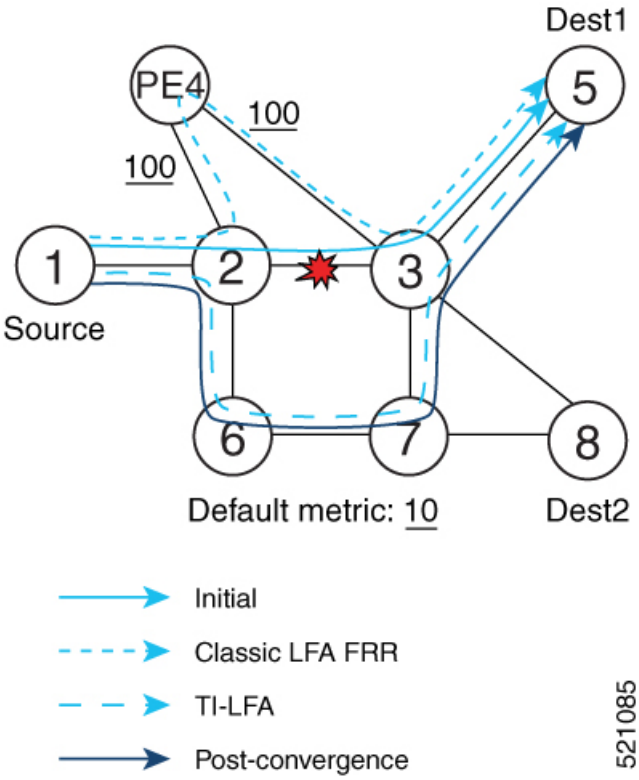
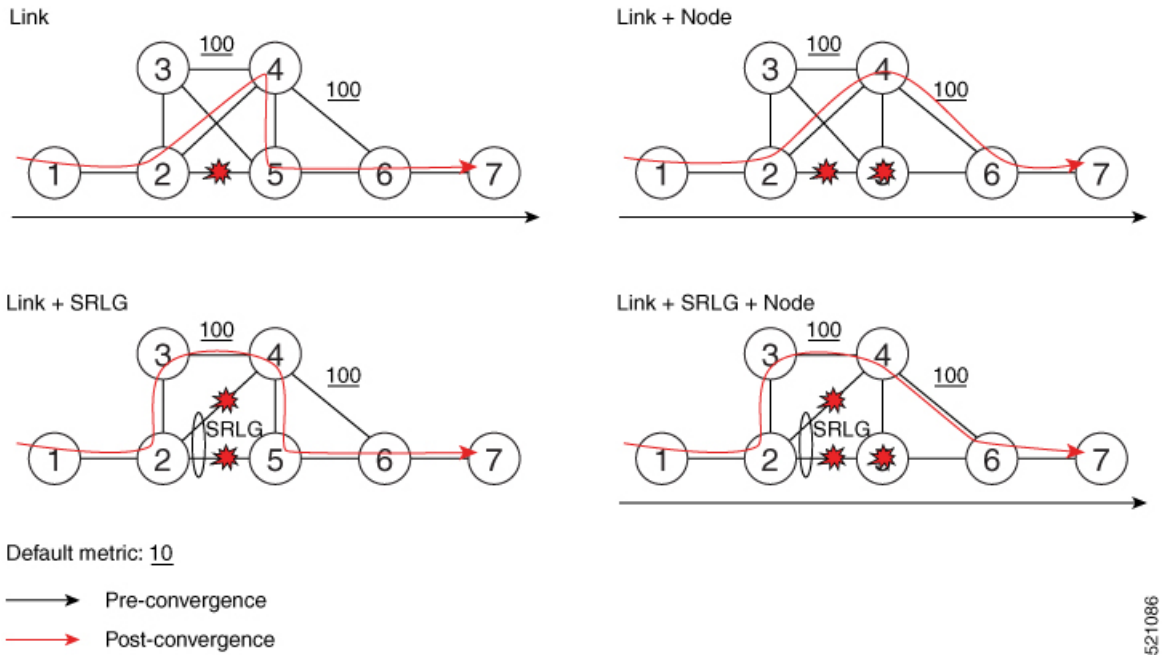


Figure 15: TI-LFA Protection Types



These stages describe how SRv6 TI-LFA works within a network:

1. Initial network state and primary path calculation :The Interior Gateway Protocol (IGP) establishes the primary forwarding paths across the network. Each router maintains its routing information base (RIB) and forwarding information base (FIB) based on the current network topology.
2. SRv6 TI-LFA backup path pre-computation: Each router, acting as a Point of Local Repair (PLR), proactively computes one or more optimal, loop-free backup paths that represent the "post-convergence path"—the exact path traffic would eventually take after the IGP fully re-converges. This pre-computation is done for every potential primary path failure, including link protection (where the failed link is excluded), node protection (where the neighbor node is excluded), and Shared Risk Link Group (SRLG) protection (where all local links sharing any SRLG with the protecting link are excluded during backup path calculation). This optimal path is preferred because it is optimal for capacity planning, simple to operate by avoiding case-by-case adjustments, and results in fewer traffic transitions since the repair path is equal to the post-convergence path. The PLR uses SRv6 segment routing principles to ensure these backup paths are safe and avoid the specific failed component. When enabling link protection, node protection, SRLG protection, or both can also be enabled, and a tiebreaker priority can be specified for multiple LFAs. For example, in a topology, Node2 can apply different protection models to protect traffic to Node7.
3. Failure detection: A link or node failure occurs in the network. The PLR rapidly detects this failure, often through mechanisms like Bidirectional Forwarding Detection (BFD), which provide sub-50ms detection times.
4. Fast reroute activation and traffic steering with SRv6 SIDs: Upon detecting the failure, the PLR immediately activates the pre-computed SRv6 TI-LFA backup path. TI-LFA calculates a post-convergence path and derives the segment list required to steer packets along this path without looping back. It encapsulates the traffic destined for the failed primary path with an SRv6 Segment Identifier (SID) list that explicitly directs the packets along the calculated backup path. For example, if Node 2 protects traffic to destination Node 5 and the protected link fails, the shortest post-convergence path might be Node2 → Node6 → Node7 → Node3 → Node5. If 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 this repair path. This steering ensures traffic continues to flow around the failure without looping back or being dropped, unlike classic LFA which might steer traffic to a suboptimal path like Node 4 that is routed over edge nodes with lower capacity links.
5. Network-wide IGP convergence: IGP reacts to the detected failure. It floods updated topology information throughout the network, and all routers perform a full re-convergence, calculating new optimal primary paths that reflect the changed topology. This process typically takes longer than the initial fast reroute.
6. Traffic restoration to new primary path: Once the IGP has fully converged and new optimal primary paths are established, traffic naturally switches from the temporary SRv6 TI-LFA backup path to these newly converged primary paths. This transition is seamless, as the SRv6 TI-LFA backup path was designed to be the post-convergence path, minimizing further traffic shifts.

Configure TI-LFA

Procedure

Step 1 Configure different types of TI-LFA protection for SRv6 IS-IS.

Example:

```
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
```

You can configure 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.

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

Step 2 Verify the SRv6 IS-IS TI-LFA configuration.

Example:

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
```

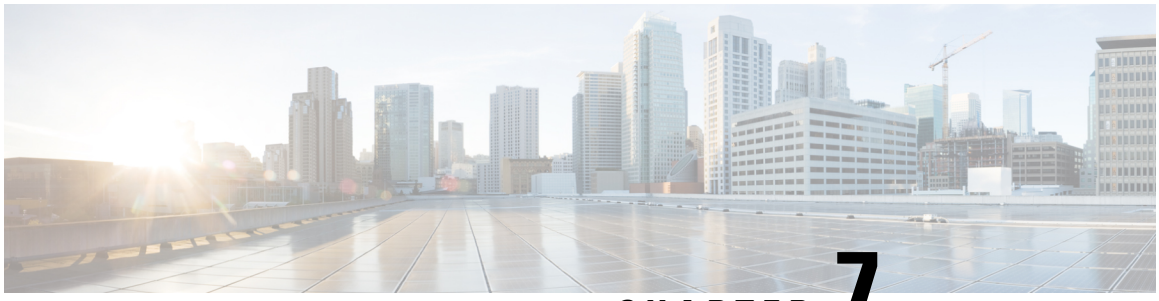
This example shows how to verify the SRv6 IS-IS TI-LFA configuration using the **show route ipv6** *ipv6-prefix* **detail** command.

```
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
```



CHAPTER 7

Path Computation Element Protocol

This chapter provides a comprehensive overview of the Path Computation Element Protocol (PCEP), a mechanism for centralized path computation and orchestration in Segment Routing networks.

It explains the roles of Path Computation Elements (PCEs) and Path Computation Clients (PCCs), detailing how they interact to request, compute, and manage Segment Routing over IPv6 Label Switched Paths (LSPs). The chapter also covers PCEP's benefits, session establishment guidelines, authentication methods, various timers for session stability, and the PCC-centric redundancy model for high availability.

Path computation element protocol

The path computation element protocol (PCEP) is a set of procedures that

- enables Path Computation Clients (PCCs) to report and delegate control of head-end SR paths to Path Computation Element (PCE) peers
- allows PCEs to request updates or modifications to path parameters, and
- supports stateful models where PCEs can initiate computations for network-wide orchestration and establishes channels over TCP with a lightweight keep-alive (KA) mechanism.

Key concepts for PCEP

- Path Computation Element (PCE): PCE is a network element that computes and orchestrates paths in a segment routing network. It identifies each segment using an IPv6 address known as a Segment Identifier or SID rather than an MPLS label, and can optimize, update, and orchestrate SRv6 paths across the network.
- Path Computation Client (PCC): PCC is a network device that interacts with the PCE to request path computations, delegate path control, and report the status of SRv6 LSPs. The PCC programs the IPv6 SIDs into the Segment Routing Header, or SRH, of packets.
- Label Switched Path (LSP): LSP is a sequence of IPv6 SIDs that define the forwarding path in the network. These SIDs are carried in the SRH. The path is established and managed.

- Maximum SID Depth (MSD): MSD defines the maximum number of Segment Identifiers (SIDs) that can be included in a Segment Routing (SR) path. It acts as a critical constraint during path computation, ensuring that computed paths adhere to the capabilities of network devices or specific policy requirements. PCCs can signal their MSD capabilities or requirements to PCEs via PCEP. This influences how paths are calculated and validated.
- PCEP related timers: PCEP timers are configurable parameters that govern the operational aspects of PCEP sessions, including session liveness (keepalives) and the management of delegated Segment Routing policies. They ensure session stability and proper handling of policy states, especially during network events or PCE unreachability. For more information, see PCEP related Timers.
- PCEP Authentication: PCEP Authentication is the security mechanisms used to verify the identity of PCEP peers and ensure the integrity of the communication channel. It prevents unauthorized entities from participating in or disrupting PCEP sessions. For more information, see PCEP authentication, on page 63.
- PCC-Centric Redundancy: A high-availability model for PCEP that centralizes LSP delegation control at the PCC. This model ensures continuous operation and efficient failover/failback of LSPs by managing re-delegation to alternate PCEs when the primary PCE becomes unavailable, and then facilitating a return to the original PCE upon its recovery. For more information, see PCC-Centric redundancy, on page 66.

Benefits of PCEP

PCEP provides significant advantages for managing and optimizing network paths. These benefits enhance network efficiency, flexibility, and resilience, and include:

- Centralized path optimization: Enables the use of a centralized PCE to compute optimal network paths based on global network knowledge, rather than relying on distributed algorithms at each router.
- Dynamic traffic engineering: Supports real-time adjustment of SRv6 paths in response to changing network conditions, policies, or failures, improving resource utilization and network resilience.
- Stateful control and delegation: Allows stateful PCEs to maintain information about existing paths and take full or partial control of path setup and modification, leading to more intelligent and coordinated path management.
- Simplified operations: Reduces manual configuration and complexity by enabling automation of path computation, provisioning, and optimization through a standardized protocol.
- Support for advanced constraints: Facilitates computation of paths based on diverse constraints, such as bandwidth, latency, disjointness, policy, which may not be supported in traditional distributed routing.
- Enhanced Fault Recovery: Enables rapid re-computation and rerouting of paths in case of network failures or congestion. This improves overall network availability and reliability.
- Security Features: Offers secure session establishment and authentication, such as TCP-AO, helping protect the control plane communications.

Usage Guidelines for Path Computation Element Protocol

PCEP session acceptance conditions

For a configured PCE peer to successfully establish a PCEP session with the PCE, the following conditions must be satisfied:

- The total number of PCEP sessions on the PCE must not exceed its configured limit.
- The Keepalive (KA) interval indicated by the PCC must be acceptable to the PCE.

MSD

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.
- 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.

How PCEP Works

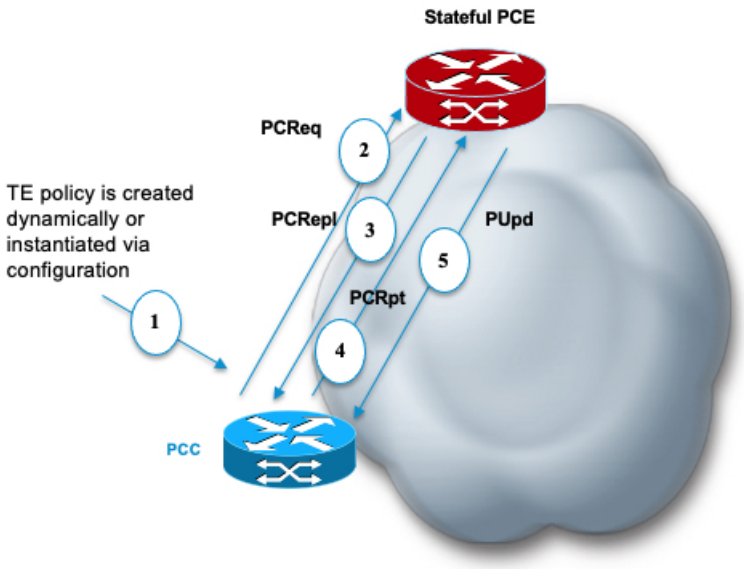
PCEP enables a PCC to request path computations from a PCE.

Summary

The key components involved in this workflow are the PCC and the PCE. The PCC initiates path computation requests and activates SR policies, while the PCE computes optimal paths and maintains policy delegation. Together, these components establish and dynamically maintain SRv6-TE policies, ensuring efficient traffic steering and network adaptability in response to changing conditions.

Workflow

Figure 16: Sample workflow with Stateful PCEP



These stages describe how a sample workflow with Stateful PCEP works:

1. The PCC is configured to instantiate an SRv6-TE policy. It sends a PCEP Path Computation Request (PCReq) to the PCE, requesting a path by specifying path attributes, optimization objectives, and constraints.
2. The PCE stores the request, computes a TE metric shortest-path, and returns the computed SID list in a PCEP Path Computation Reply (PCRepl).
3. The PCC allocates a Binding SID (BSID) and activates the SR Policy using the SID list computed by the PCE. The PCC then sends a Path Computation Report (PCRpt) to the PCE, delegating the SR Policy to the PCE and including the BSID
4. PCE updates paths as needed: The PCE updates the paths when required, for example, following a multi-domain topology change that impacts connectivity, ensuring the SR Policy remains optimized

Configure head-end router as a PCEP Path Computation Client

Before you begin

Ensure that the PCC and PCE addresses are routable to allow the TCP connection (for exchanging PCEP messages) to be established between them

Procedure

Step 1 Enable the SR-TE head-end router as a PCEP client (PCC) with 2 PCEP servers (PCE) with different precedence values.

Example:

```
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
```

Step 2 Enable PCEP reporting for all policies in the node.

Example:

```
Node1(config-sr-te-pcc)# report-all
Node1(config-sr-te-pcc)# exit
```

Step 3 Set the maximum SID Depth (MSD).

Example:

```
Node1(config-sr-te)# srv6
Node1(config-sr-te-srv6)# maximum-sid-depth 5
Node1(config-sr-te-srv6)# exit
```

Step 4 Enable SR-TE related syslogs.

Example:

Node1(config-sr-te)# **logging** Node1(config-sr-te-log)# **policy status** Node1(config-sr-te-log)# **exit** Node1(config-sr-te)#

Step 5 Use the **show running-config** command to review the current configuration.

Example:

```
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
  !
 !
!
```

Step 6 Verify the status and summary of IPv6 PCEP peers for SRv6-TE) on the router.

Example:

```
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
```

Step 7 (optional) Enable ECMP-aware path computation for TE metric to customize the SR-TE path calculation.

Example:

```
Router(config-sr-te)# te-latency
```

Example:

Note
ECMP-aware path computation is enabled by default for IGP and LATENCY metrics.

What to do next

- (optinal) Configure PCEP Authentication
- (optinal) Configure PCEP-Related Timers

- (optinal) Configure PCEP Redundancy Type

PCEP authentication

The PCEP session authentication is a security mechanism that

- establishes a secure and trusted communication channel between a PCC and a PCE
- ensures that only authorized devices can participate in PCEP sessions, and
- protects against unauthorized access and data tampering.

Methods of PCEP authentication

PCEP authentication can be achieved using one of two primary methods:

- TCP Message Digest 5 (MD5) Authentication: This method uses a clear text or encrypted password for authentication. Segments lacking a Message Authentication Code (MAC) that matches the configured password are rejected.
- TCP Authentication Option (TCP-AO): TCP-AO uses Message Authentication Codes (MACs), which provides these benefits:
  - 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, protecting connections by deriving traffic keys from the MKT and coordinating changes between endpoints. Segments lacking a MAC that matches the configured key chain are rejected.

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 Authentication

Procedure

Step 1 Configure PCEP Authentication using using either TCP MD5 or TCP-AO.

- Configure TCP Message Digest 5 (MD5) Authentication. Specify if the password is encrypted or clear text.

```
Router(config-sr-te-pcc)# pce address ipv6 ipv6-PCE-address[password {clear | encrypted} LINE]
```

- Configure TCP Authentication Option (TCP-AO). 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]
```

Step 2 Verify the PCEP Authentication

PCEP-Related Timers

PCEP related timers are configurable parameters that

- govern the behavior and stability of PCEP sessions and the management of delegated SR policies
- ensure proper session maintenance, and
- manage the lifecycle of SR policies initiated or delegated by a PCE.

Types of PCEP-Related Timers:

- Keepalive timer: This timer specifies the frequency (in seconds) at which keepalive messages are sent from the PCC to its PCE peers. These messages confirm the liveness of the PCEP session. Default: 30 seconds
- Deadtimer: This timer defines how long (in seconds) remote PCE peers will wait before declaring a PCEP session down if no PCEP messages are received from the PCC. It acts as a session timeout. Default: 120 seconds
- Delegation timeout: This timer determines the maximum duration (in seconds) a delegated SR policy can remain active on the PCC without an active connection to its delegating PCE. Default: 60 seconds.
- Initiated orphans timer: This timer specifies the amount of time (in seconds) a PCE-initiated SR policy will remain delegated to a PCE peer that has become unreachable by the PCC. It allows for a grace period for the PCE to reconnect. Default: 180 seconds
- Initiated state timer: This timer defines the duration (in seconds) a PCE-initiated SR policy will remain programmed and active in forwarding on the head-end, even when it is not currently delegated to any PCE (e.g., after the orphan timer expires). Default: 600 seconds.

How PCE-Initiated SR Policy Timers Operate

Summary

The key components involved in this process are:

- Path Computation Element (PCE): The entity that initiates and delegates SR policies, for example, PCE A.
- Segment Routing Policy: The network path definition managed by the PCE, for example, Policy P.
- Head-end Router (PCC): The device that receives, programs, and manages the delegated SR policy, for example, Head-end N.
- PCEP Timers: Specifically, the `initiated orphans` and `initiated state` timers that govern the policy's lifecycle during PCE unreachability.

The process involves the head-end router using initiated orphans and initiated state timers to either preserve the SR policy through re-delegation or remove it from forwarding if no PCE takes ownership within defined timeframes.

Workflow

These are the stages for PCE-initiated SR policy timers:

1. Policy Instantiation and Initial Delegation: PCE A instantiates SR Policy P at Head-end N. Head-end N then delegates Policy P to PCE A and programs it into its forwarding plane.
2. PCE Unreachability Detection: If Head-end N detects that PCE A is no longer reachable, head-end N starts both the initiated orphans and initiated state timers for SR Policy P.
3. Re-delegation Grace Period: 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).
4. Policy Removal: 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.

Configure PCEP-related timers

Set timer intervals that govern PCEP session behavior and management of PCE-initiated SR policies.

Adjusting these timers allows tailored session management for specific network performance needs. You can configure the timers independently and in any order.

Procedure

Step 1 Use the **timers keepalive** command to specify how often keepalive messages are sent from the PCC to its peers.

The range is from 0 to 255 seconds; the default value is 30.

Example:

```
Router(config-sr-te-pcc)# timers keepalive 30
```

Step 2 Use the **timers deadtimer** command topecify 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.

Example:

```
Router(config-sr-te-pcc)# timers deadtimer 120
```

Step 3 Use the **timers delegation-timeout** command to specify how long a delegated SR policy can remain active without an active connection to its delegating PCE.

The range is from 0 to 3600 seconds; the default value is 60.

Example:

```
Router(config-sr-te-pcc)# timers delegation-timeout 60
```

Step 4 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.

Example:

```
Router(config-sr-te-pcc)# timers initiated orphans 180
```

Step 5 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.

Example:

```
Router(config-sr-te-pcc)# timers initiated state 600
```

PCC-Centric redundancy

The PCC-Centric Redundancy is a high-availability model for PCEP that

- centralizes LSP delegation control at the PCC
- ensures continuous operation and efficient failover/failback of LSPs, and
- manages re-delegation to alternate PCEs when the primary PCE becomes unavailable, and then facilitates a return to the original PCE upon its recovery.

How the PCC-Centric Redundancy Model Works

Summary

The key components involved in this process are:

- PCC: The device managing the LSP delegation.
- PCE: The entity that computes and delegates LSPs.
- Label switched path (LSP): The network path being delegated and managed.
- Delegation fallback timer: A specific timer that governs the return of an LSP to its original PCE.

The PCC-centric redundancy model maintains high availability for Label Switched Paths (LSPs) in PCEP environments by managing the delegation of LSPs between primary and alternate Path Computation Elements (PCEs). The process includes automatic failover to an alternate PCE when the original becomes unreachable and graceful restoration to the primary PCE once it recovers, coordinated by a delegation fallback timer.

Workflow

These stages describe the PCC-Centric Redundancy Model.

1. Initial LSP delegation: After an LSP is created, the PCC automatically delegates it to the specific PCE that computed the LSP.

2. PCE disconnection handling: If the original PCE (to which the LSP was delegated) becomes disconnected or unreachable, the PCC automatically re-delegates the LSP to another available PCE.
3. Original PCE reconnection and fallback: If the original PCE reconnects and becomes reachable again, a "delegation fallback timer" is initiated by the PCC. Once this timer expires, the LSP is re-delegated back to the original PCE, even if that PCE has a worse preference than the currently active PCE. This action ensures that LSPs eventually return to their primary, originating PCE.

Configure PCEP Redundancy Type

This task describes how to enable the PCC-centric high-availability model for PCEP, which modifies the default LSP delegation behavior to enhance redundancy.

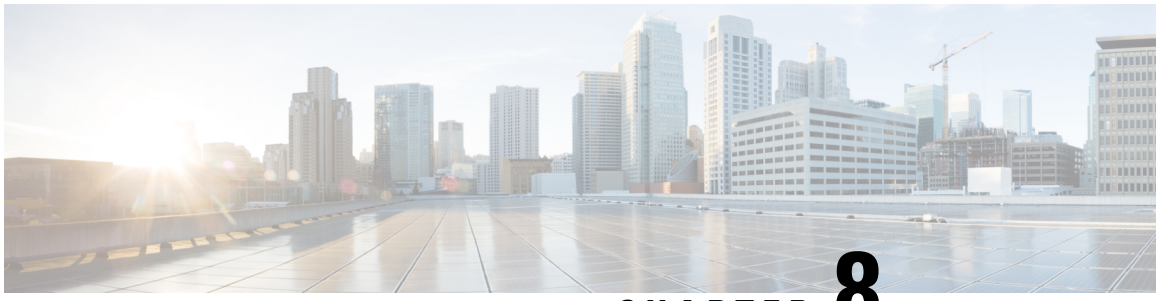
Procedure

Step 1 Run the **redundancy pcc-centric** command to nable PCC-centric high-availability model.

Example:

```
Router(config-sr-te-pcc)# redundancy pcc-centric
```

Step 2 Verify the PCEP Redundancy Type.



CHAPTER 8

SRv6 Microloop Avoidance for Network Resilience

SRv6 microloop avoidance is a SRv6 feature designed to prevent brief, transient packet loops that occur during network topology changes. These microloops can cause packet loss and jitter, but this feature detects such possibilities and temporarily steers traffic onto loop-free paths, ensuring stable forwarding during network convergence. This significantly enhances network stability, performance, and the overall user experience.

SRv6 microloop avoidance

A SRv6 microloop avoidance is a SRv6 feature that

- detects if microloops are possible following a topology change
- creates temporary loop-free SR-TE policy paths to the destination using a list of segments, and
- replaces the temporary paths with regular forwarding paths after a Routing Information Base (RIB) update delay timer expires.

Key concepts of SRv6 microloop avoidance

- Microloops: Amicroloop is a brief packet loop that occurs in the network following a topology change (link down, link up, or metric change events), is caused by the non-simultaneous convergence of different nodes in the network, and results in packet loss, jitter, and out-of-order packets if traffic is looped between nodes.
- Temporary SR-TE Policy Paths: Loop-free paths to a destination, constructed using a list of segments, that are temporarily installed by the feature to avoid microloops.
- RIB update delay timer: A configurable timer that dictates how long the temporary microloop avoidance policy is used before the network transitions to regular, natively converged forwarding paths.
- Loop-free forwarding: The primary goal and outcome of the feature, ensuring that traffic continues to be forwarded without being caught in transient loops during network topology changes and convergence.

Benefits of SRv6 microloop avoidance

SRv6 microloop avoidance provides significant value by mitigating the detrimental effects of microloops, a long-standing challenge in IP networks. This feature offers substantial benefits for network stability, performance, and user experience within SRv6 deployments.

The feature provides these key benefits:

- Addresses a fundamental network challenge: SRv6 microloop avoidance resolves the issue of transient packet loops that occur immediately after topology changes due to non-simultaneous router convergence.
- Prevents service degradation: SRv6 microloop avoidance eliminates packet loss, jitter, and out-of-order packets caused by microloops, which directly impact application performance, especially for real-time services.
- Ensures loop-free forwarding during convergence: The feature proactively detects potential microloops and temporarily steers traffic onto specially constructed, loop-free SRv6-TE policy paths, ensuring continuous and stable forwarding during network convergence.
- Improves network stability and predictability: By preventing microloops, the network's behavior during topology changes becomes more predictable and stable, reducing transient issues and simplifying troubleshooting.
- Enhances application performance: By preventing microloop-induced packet impairments, it directly contributes to a higher quality of service for all applications, particularly sensitive ones.
- Leverages SRv6 capabilities: The solution seamlessly integrates into existing SRv6 deployments by using SRv6 Segment Identifiers (SIDs) to construct and enforce temporary microloop-avoidant paths.
- Offers local behavior with global benefits: The feature operates as a local behavior, meaning that individual nodes can implement it to protect their own forwarding, contributing to overall network stability without requiring universal deployment.

Limitations of SRv6 microloop avoidance

- 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.

How SRv6 microloop avoidance works

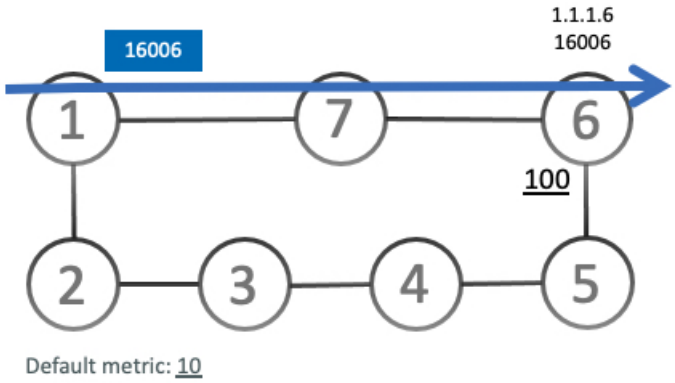
Summary

The SRv6 Microloop Avoidance process involves a router detecting potential microloops following a topology change. The Interior Gateway Protocol (IGP) then computes and temporarily installs an SRv6 microloop-avoidant path using a Segment Identifier (SID) list. After the network converges and a configurable RIB update delay expires, the IGP replaces this temporary path with the native post-convergence forwarding path.

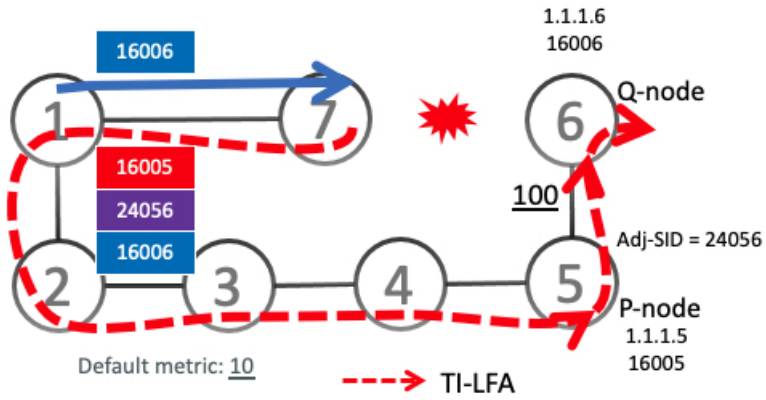
Workflow

These stages describe how SRv6 Microloop Avoidance works in a network:

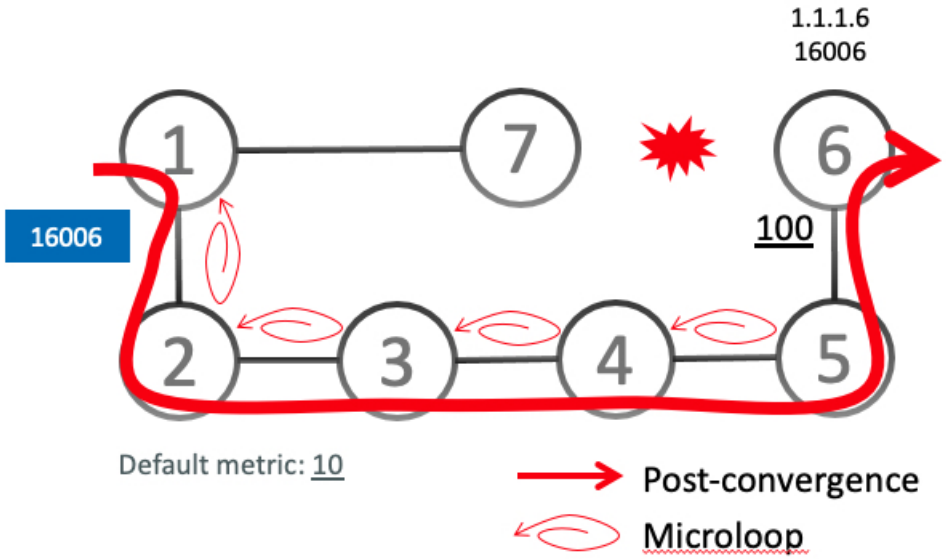
1. Initial network state: The network operates in a stable state. Node1 sends traffic to Node6 (prefix SID 16006) via Node7. Node7 is configured with TI-LFA to protect traffic to Node6.



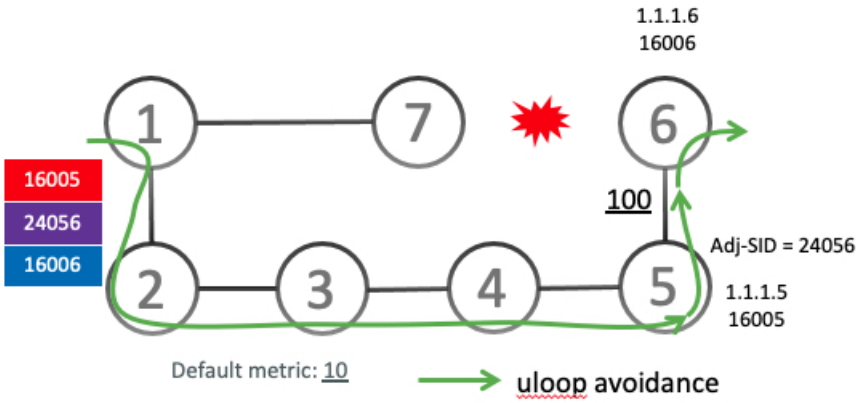
2. Topology change and failure detection: A link failure occurs, for example, between Node6 and Node7. All nodes are notified of this topology change. Node7's TI-LFA pre-computes a backup path for traffic to Node6 (prefix SID 16006), which would steer traffic toward Node5 (prefix SID 16005) and then via the link between Node5 and Node6 (adj-SID 24056).



3. Microloop detection : A router with the SRv6 microloop avoidance feature detects if microloops are possible for a destination on the post-convergence path following the topology change. For example, if Node2 converges before Node3, Node3 might send traffic back to Node2 as the shortest IGP path to Node6, creating a microloop between Node2 and Node3.

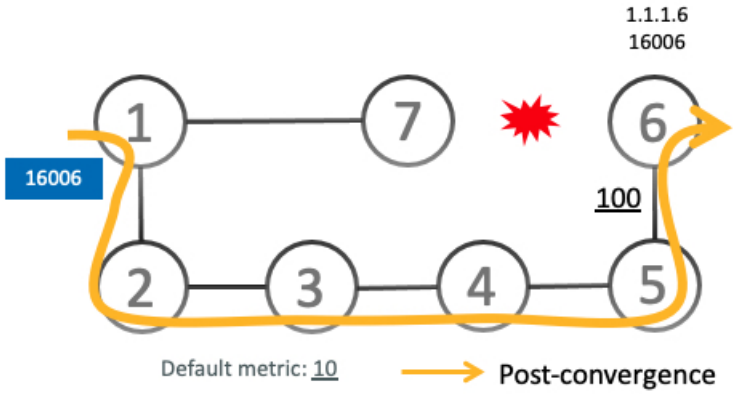


4. Microloop-avoidant path computation and installation: If a node determines that a microloop could occur on the new topology (In this example, on Node1 for traffic to Node6), the IGP computes a microloop-avoidant path. This path is constructed to steer traffic to the destination loop-free over a temporary path. For Node6, this path might be {16005, 24056, 16006}. The IGP then updates the forwarding table and temporarily installs the SID-list imposition entries associated with this microloop-avoidant path for destinations with possible microloops.



5. Network convergence and temporary path usage: All nodes in the network converge and update their forwarding tables. Traffic for destinations prone to microloops is steered using the temporary microloop-avoidant SRv6 SID list.

6. Transition to native forwarding: After the RIB update delay timer expires, the IGP updates the forwarding table again. The microloop-avoidant SRv6 SID list is removed, and traffic now natively follows the post-convergence path.



Configure SRv6 IS-IS microloop avoidance

Use this procedure to configure SRv6 IS-IS Microloop Avoidance and set the Routing Information Base (RIB) update delay value.

Before you begin

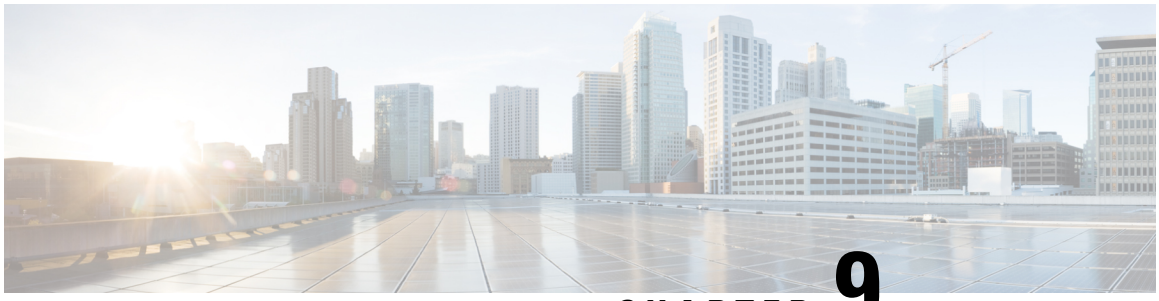
Complete the SRv6 configuration before performing these steps.

Procedure

Configure SRv6 IS-IS microloop avoidance.

Example:

```
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
```



CHAPTER 9

Traffic Engineering in IPv6 Networks Using SRv6-TE

This chapter introduces Segment Routing over IPv6 Traffic Engineering (SRv6-TE), a solution designed to provide granular control and flexibility in managing network traffic.

SRv6-TE enables administrators to steer traffic across IPv6 networks according to specific policies and requirements, facilitating the creation of explicit paths for applications demanding particular Quality of Service (QoS) levels, such as low latency or high bandwidth.

SRv6-TE offers significant advantages over traditional methods like MPLS RSVP-TE, including enhanced scalability, reduced complexity, optimized resource utilization through ECMP, and seamless integration with existing IPv6 infrastructure. It supports advanced traffic engineering capabilities, dynamic path computation, and explicit path configurations, making it essential for modern, agile networks.

• Segment Routing over IPv6 Traffic Engineering (SRv6-TE), on page 75

Segment Routing over IPv6 Traffic Engineering (SRv6-TE)

Segment Routing over IPv6 Traffic Engineering (SRv6-TE) is a traffic engineering solution that

- enables you to steer traffic across a network based on specific policies and requirements and provides greater control over how traffic flows through the network
- 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, and
- leverages the concept of source routing, where the source node determines 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.

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Traffic Engineering	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
SRv6 Traffic Engineering	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700);Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is now supported on: • 8212-32FH-M • 8212-48FH-M • 88-LC1-52Y8H-EM • 88-LC1-36EH • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E

Feature Name	Release Information	Feature Description
SRv6 Traffic Engineering	Release 7.10.1	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: • `Cisco-IOS-XR-segment-routing-ms-cfg` (see GitHub, YANG Data Models Navigator)

Why SRv6-TE is Essential for Modern Networks

SRv6-TE provides enhanced control and flexibility in steering traffic across a network based on specific policies and requirements. This solution eliminates the need for intermediate nodes to maintain per-application and per-flow state. Only the head-end nodes at the network edge maintain state, while the remaining nodes forward packets based on instructions encoded in the packet header. This source routing approach improves scalability and reduces complexity.

Key benefits of SRv6-TE include:

- Explicit traffic steering: Enables the creation of explicit paths for traffic flows that require specific Quality of Service (QoS) levels, such as low latency or high bandwidth.
- Source routing efficiency: The source node calculates and encodes the path directly in the packet header, which offloads path computation from intermediate routers.
- Optimized resource utilization: Utilizes network bandwidth more effectively than traditional MPLS RSVP-TE by leveraging Equal-Cost Multi-Path (ECMP) within each segment.
- Simplified network management: Embeds forwarding paths within packets, which reduces overhead and makes the network more responsive to changes.
- Support for advanced traffic engineering: Supports flexible algorithms and policies for intents like low latency, disjointness, affinity inclusion/exclusion, and Shared Risk Link Group (SRLG) exclusion, often with Path Computation Element (PCE) assistance.
- Seamless IPv6 integration: Leverages existing IPv6 infrastructure for easier deployment.

Comparison between SRv6-TE and MPLS RSVP-TE

Table 19: SRv6-TE vs. traditional MPLS RSVP-TE

Feature	SRv6-TE	Traditional MPLS RSVP-TE
State Maintenance	Only head-end nodes maintain state	All nodes along the path maintain per-flow state
Routing Paradigm	Source routing with path encoded in packet header.	Signaling-based path setup with RSVP protocol.
Bandwidth Utilization	More efficient via ECMP within segments	Less efficient, limited ECMP usage
Traffic Engineering Flexibility	Supports advanced policies including flexible algorithms and explicit paths.	Supports explicit paths but with more overhead.
Integration	Native IPv6-based, leverages IPv6 infrastructure	MPLS-based, requires MPLS infrastructure

Methods of traffic engineering with SRv6

Traffic engineering over SRv6 can be accomplished in these 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-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

An SRv6-TE policy is a traffic engineering mechanism that

- defines specific end-to-end paths using lists of micro-segment IDs (uSIDs)
- enables you to steer packets along chosen routes instead of default IGP paths, and
- is uniquely identified by its head-end, color, and end-point attributes.

Components of SRv6-TE policy

An SRv6-TE policy is uniquely identified by an ordered list of three components:

- Head-end: The node where the SRv6-TE policy is instantiated.
- Color: A numerical value that distinguishes between multiple policies configured for the same node pair (Head-end and End-point). Every policy between the same node pairs requires a unique color value.
- 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.

Each SRv6-TE policy utilizes one or more candidate paths. A candidate path can consist of a single SID-list or a set of weighted SID-lists for Weighted Equal-Cost Multi-Path (WECMP). A SID list can be dynamically computed by a Path Computation Element (PCE) or explicitly configured by a user. For more information, see SRv6-TE Policy Path Types.

Candidate paths

The candidate path is a category of routing path that

- consists of a single segment list or a set of weighted SID-lists
- serves as one of the multiple possible routes within an SR Policy, and
- enables load sharing and failover by allowing traffic to be distributed or switched among the SID-lists.

Characteristics of candidate path

A candidate path has these 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. The protocol of the source is not relevant in the path selection logic.

Types of candidate path

An SR Policy manages multiple candidate paths, but only one candidate path is selected and installed in the RIB/FIB as the active path at any time. Candidate paths can be dynamic (computed) or explicit (manually specified), providing flexibility for traffic engineering and network resilience.

Configure SRv6-TE candidate Paths with weighted SID lists

Use this procedure to define SRv6-TE policies that utilize multiple segment lists with assigned weights for weighted load balancing.

Procedure

Step 1 Set the SRv6 SID format and define the first SRv6 segment list.

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
```

Step 2 Configure the SRv6-TE policy with candidate paths and weighted SID lists.

Example:

```
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
```

Step 3 Verify the configuration with show running configuration.

Example:

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

Explicit Paths

An SRv6 TE explicit path is a traffic engineering route in IPv6 networks that:

- uses a predefined list of Segment Identifiers (SIDs)
- determines the precise forwarding path for packets, giving direct control over how traffic moves through the network, and
- allows segment lists to include uSIDs, uSID carriers, or a mix of both.

Benefits of explicit paths

Explicit paths provide several advantages in traffic engineering:

- Precise traffic control: You can specify the exact route that packets take, ensuring traffic follows your preferred path through the network.
- Predictable performance: By defining the path, you avoid unexpected routing changes, leading to consistent and reliable application performance.
- Enhanced policy enforcement: You can meet specific service requirements—such as low latency, security, or bandwidth guarantees—by choosing the path that best fits your policy.
- Improved resource utilization: You can balance traffic loads or avoid congested links by selecting less utilized routes, optimizing network efficiency.
- Deterministic failover: In case of a failure, you can design backup explicit paths, ensuring quick and predictable recovery.

Guidelines for SRv6-TE explicit path

Address Adj-SID preference

When configuring explicit paths using IP addresses of links, consider the SRv6-TE process's Adj-SID preferences.

The SRv6-TE process prefers the protected Adj-SID of the link if one is available. Furthermore, it prefers a manual-protected Adj-SID over a dynamic-protected Adj-SID. You can configure the path to prefer protected or unprotected Adj-SIDs, or to use only protected or unprotected Adj-SIDs, based on your specific network requirements.

Enable Segment List SID validation

Enable SRv6-TE explicit segment list SID validation. This allows the head-end node to validate the SIDs in an explicit SRv6-TE segment list against the SR-TE topology database.

Compose segment lists

Compose segment lists using uSIDs, uSID carriers, or a combination of both.

Configure SRv6-TE policy with explicit path

Procedure

Step 1 Create a segment list with SRv6 uSIDs.

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_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::
```

Step 2 Create the SRv6-TE policy.

Example:

```
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
```

Step 3 View the show run configuration.

Example:

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

Step 4 Verify the SR-TE policy configuration.

Example:

```
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
```

SRv6-TE explicit segment list SID validation

SRv6-TE explicit segment list SID validation is a process that evaluates whether an SR policy's candidate path's explicit segment list is valid and therefore usable. The head-end node performs this validation by checking if the specified hops are present in the SR-TE topology database.

Segment list SID validation

When enabled, this validation occurs at two key times:

- Before programming the SR policy.
- After an IGP topology change.

If the validation process identifies issues:

- The segment list is declared invalid if validation fails.
- An SR policy candidate path is declared invalid if it contains no valid segment lists.
- An SR policy is declared invalid if it has no valid candidate paths.

Usage Guidelines for SRv6 explicit segment list SID validation

Enable or disable segment list SID validation

Enable segment list SID validation 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 head-end (for example, in a multi-domain case), you can disable the topology check for that specific segment list.

Distribute SR-TE topology from appropriate sources

Distribute the SR-TE topology from the correct source for accurate path computation. For intra-domain paths, the SR-TE topology should be distributed from an Interior Gateway Protocol (IGP). For multi-domain paths, use BGP-LS for distribution.

Ensure SRv6 segment lists are not empty

Ensure that the SRv6 segment list in an explicit candidate path of an SRv6 policy is not empty. An empty segment list will prevent the policy from steering traffic correctly.

Maintain consistent data plane types within segment lists

Ensure all segments within a segment list share the same data plane type. All segments in a given segment list must consistently be either SRv6 or SR-MPLS.

Validate SIDs against local parameters

All SIDs in a segment list are validated against the local SID block and SID format. Top SID validation occurs by performing path resolution using the top SID. A segment list is also validated against the Maximum Segment Depth (MSD).

Enable SID validation globally for all SRv6 explicit segment lists

Before you begin

Verify if the SR-TE topology is available on the headend PCC router using the **show segment-routing traffic-eng topology** command.

Procedure

Step 1 Enable SID validation globally.

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)# 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
```

Step 2 Verify the running configuration.

Example:

```
segment-routing
 traffic-eng
  segment-lists
   srv6
    topology-check
   !
   segment-list p1_r8_1
    srv6
     no topology-check
    !
   !
  !
 !
!
```

Step 3 Verify the SID validation globally for all SRv6 explicit segment lists.

Example:

```
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 a type of SRv6-TE path that

- is based on an optimization objective and a set of constraints
- is computed by the head-end to generate a SID-list, and
- automatically recomputes when the network topology changes.

Optimization objectives for dynamic path computation

An optimization objective defines the primary goal for computing a dynamic path. This objective guides the path computation process to find the most suitable path based on criteria such as minimizing latency, maximizing bandwidth, or finding the shortest path.

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: Computes the path with the least number of hops.
- IGP metric: Computes the path based on the Interior Gateway Protocol (IGP) metric. For more information, refer to the Implementing IS-IS and Implementing OSPF chapters in the Routing Configuration Guide for Cisco 8000 Series Routers
- TE metric: Computes the path based on the Traffic Engineering (TE) metric. See the section for information about configuring TE metrics. See the Configure Interface TE Metrics task.
- Delay (latency): Computes the path based on link latency. See the chapter for information about measuring delay for links or SRv6 policies.

See How Optimization Objectives Determine Dynamic Paths.

Constraints for dynamic path computation

Constraints are rules or conditions applied during the path computation process that restrict the possible routes a dynamic path can take. These allow the head-end router to compute a path that adheres to specific network policies or requirements.

- 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. For more information, see the Named Interface Link Admin Groups and SRv6-TE Affinity Maps section.

Note When performing path computation on a PCE, configuring both affinity and disjoint-path constraints simultaneously is not supported.

- 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). For details and usage guidelines, see the Segment Protection-Type Constraint section.

Guidelines of SRv6 policy with dynamic path

- An SRv6-TE policy combines a dynamic path with specific optimization objectives and constraints.
- If you do not specify a customized per-policy locator and Binding SID (BSID) behavior, the policy uses the global locator and BSID behavior. Similarly, if you do not specify a customized per-policy source address, the policy uses the local IPv6 source address.
- Disjoint-path and affinity constraints are mutually exclusive and cannot be configured simultaneously.

Configure SRv6 policy with dynamic path

Use this procedure to configure an SRv6-TE policy that uses a dynamic path, including optimization objectives and affinity constraints.

Procedure

Step 1 Configure an SRv6-TE policy that uses a dynamic path.

Example:

This 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
```

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
```

Step 2 View the running configuration of SRv6 policy with dynamic path.

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.

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

This 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.

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

Step 3 Verify the SRv6 policy with dynamic path.

Example:

```
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
```

Configure interface TE metrics

Use this procedure to assign a Traffic Engineering (TE) metric to specific interfaces, which influences dynamic path computation in SRv6-TE policies.

Procedure

Step 1 Use the **metric** use the **metric** *value* command. command to configure the TE metric for the interface.

The *value* range is from 0 to 2147483647.

Example:

```
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 100
```

Step 2 Verify the configured TE metrics for the interfaces.

Example:

```
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
```

Named Interface Link Admin Groups and SRv6-TE Affinity Maps

Named interface link admin groups and SRv6-TE affinity maps are SRv6 TE features that

- provide a simplified and more flexible means of configuring link attributes and path affinities
- allow the use of descriptive color names instead of hexadecimal numbers for attributes, and
- enable the definition of granular constraints for SRv6-TE policies.

Traditional TE scheme

In the traditional Traffic Engineering (TE) scheme, links are configured with attribute-flags. These flags are flooded as TE link-state parameters using Interior Gateway Protocols (IGPs), such as Open Shortest Path First (OSPF).

Enhanced configuration with affinity maps

Named interface link admin groups and SRv6-TE affinity maps allow you to assign, or map, up to 32 color names for affinity and attribute-flag attributes. This approach replaces the use of 32-bit hexadecimal numbers, making configurations more intuitive. After these mappings are defined, the attributes can be referred to by their corresponding color name directly in the CLI.

Defining Constraints

Using these affinity maps, you can define constraints with increased flexibility. Constraints can be specified using these arguments, where each statement can contain up to 10 distinct colors.

- `include-all`
- `include-any`
- `exclude-any`

Configuration scheme coexistence

You can configure affinity constraints using either traditional attribute flags or the Flexible Name Based Policy Constraints scheme. However, if configurations for both schemes exist simultaneously, only the configuration pertaining to the new, name-based scheme is applied to the SRv6-TE policies.

Configure Named Interface Link Admin Groups and SRv6-TE Affinity Maps

Use this procedure to configure Named Interface Link Admin Groups and SRv6-TE Affinity Maps, providing a flexible way to define link attributes and path affinities for SRv6-TE policies.

Procedure

Step 1 Assign affinity to interfaces on routers with interfaces that have an associated admin group attribute.

To assign affinity to interfaces, use the **segment-routing traffic-eng interface interface affinity name** command . This configuration is applicable to any router (SRv6-TE head-end or transit node) with colored interfaces.

Example:

```
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
```

Step 2 Define affinity maps.

To define the 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.

Example:

```
Router(config-sr-te)# affinity-map
Router(config-sr-te-affinity-map)# name brown bit-position 1
```

Step 3 Verify the show running configuration.

Example:

```
segment-routing
 traffic-eng
  interface HundredGigE0/0/0/0
   affinity
    name brown
   !
  !
  affinity-map
   name brown bit-position 1
  !
end
```

Segment Protection-Type Constraint

Segment protection-type behavior is a feature that

- introduces the ability to control whether protected or unprotected segments are used
- influences the encoding of the SID-list of an SRv6 policy candidate path, and
- determines whether failures along the path are protected by TI-LFA.

Segment Types

The types of segments that can be used when building a SID-list include uSIDs and adjacency SIDs (uA SIDs).

- Prefix SID: A global segment that represents a prefix identifying a specific node. A prefix SID is programmed with a backup path computed by the Interior Gateway Protocol (IGP) using Topology-Independent Loop-Free Alternate (TI-LFA).
- Adjacency SID: A local segment that represents 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 within a SID-list to forward a packet along a path traversing specific nodes and/or over specific interfaces between nodes. The specific type of segment used when encoding the SID-list will determine whether failures along the path would be protected by TI-LFA. This capability allows operators to offer either unprotected or protected services over traffic engineered paths, depending on their service offerings.

Segment protection-type constraint behaviors

These 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.

Limitations of segment protection-type constraint

- 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.

Configure Segment Protection-Type Constraint

Procedure

Use the **constraints segments protection** command to configure the segment protection-type behavior.

Example:

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 is a feature that

- allows operators to customize Interior Gateway Protocol (IGP) shortest path computation
- enables forwarding beyond traditional link-cost-based shortest paths using custom SRv6 locators, and
- provides automatically computed traffic-engineered paths to any destination reachable by the IGP.

Network Slices with Flexible Algorithm

Consider an SR domain with two distinct network slices, each assigned a `/32` uSID Locator Block:

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)

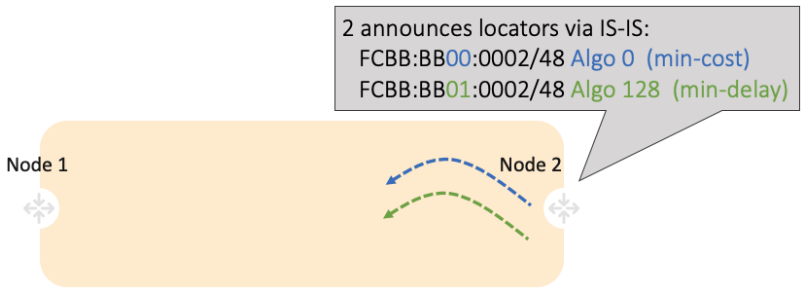
- Min-Cost Slice — FCBB:BB**00**/32
- Min-Delay Slice — FCBB:BB**01**/32

SR node2 gets a Shortest-Path Endpoint uSID (uN) from each slice:

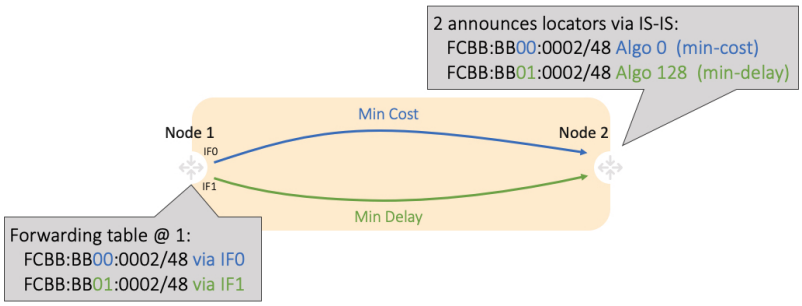
- uN min-cost of Node2 — FCBB:BB**00**:0002/48
- uN min-delay of Node2 — FCBB:BB**01**:0002/48

Node2 announces locators via IS-IS:

- FCBB:BB**00**:0002/48 Algo **0** (min-cost)
- FCBB:BB**01**:0002/48 Algo **128** (min-delay)



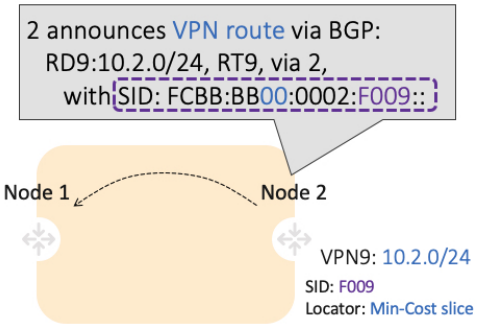
IS-IS in Node1 computes shortest paths for each locator and programs them in the FIB:



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

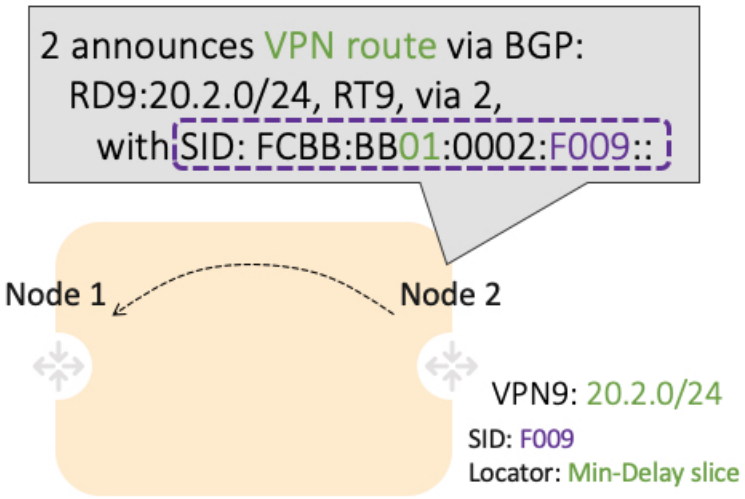


Use-case: VPN over Min-Delay Slice (Control Plane Behavior)

- Intuitive uSID program for routes advertised by Node2:
  - Within the Min-Delay Slice (FCBB:BB**01**)
  - Follow the shortest-path to Node2 (**0002**)

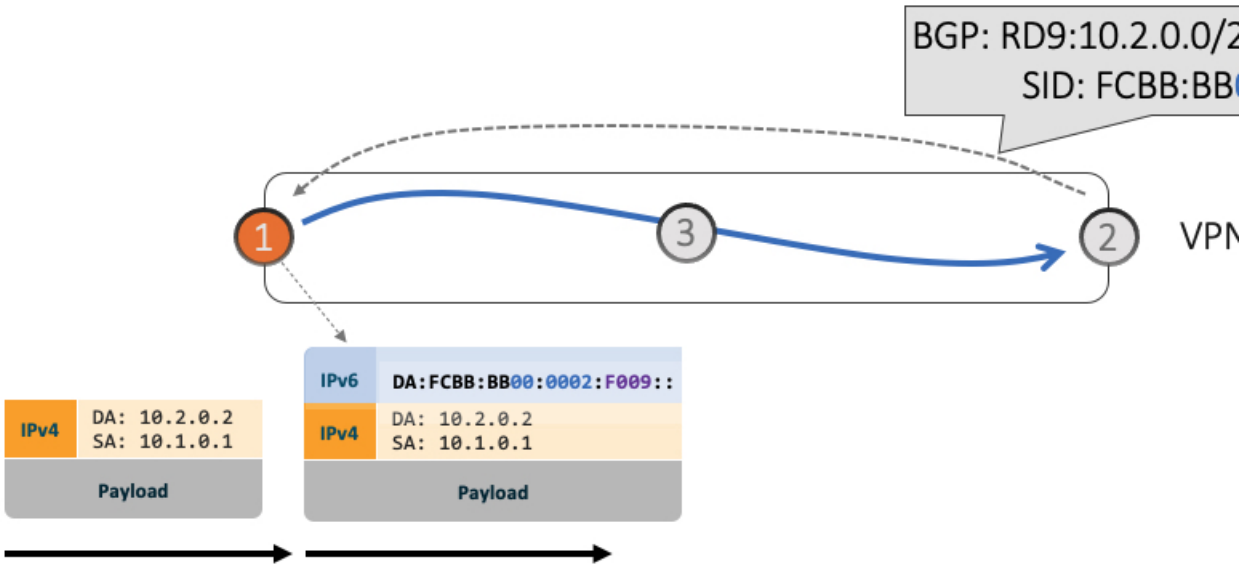
- Execute VPN9 decapsulation at Node2 (**F009**)

- Hardware Efficiency
  - Egress PE 2 processes multiple uSIDs with a single /64 lookup
  - FCBB:BB**01**:**0002**:**F009**/64



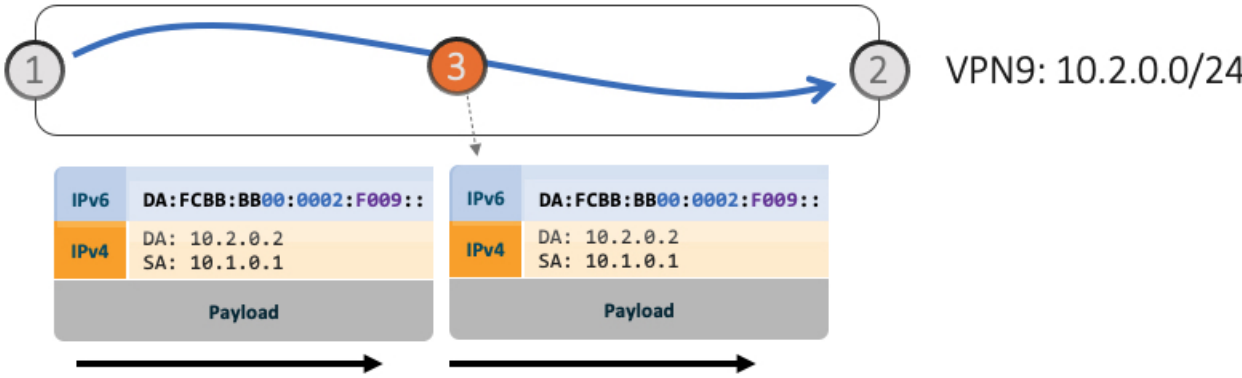
Use-case: VPN over Min-Cost Slice (Data Plane Behavior)

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**

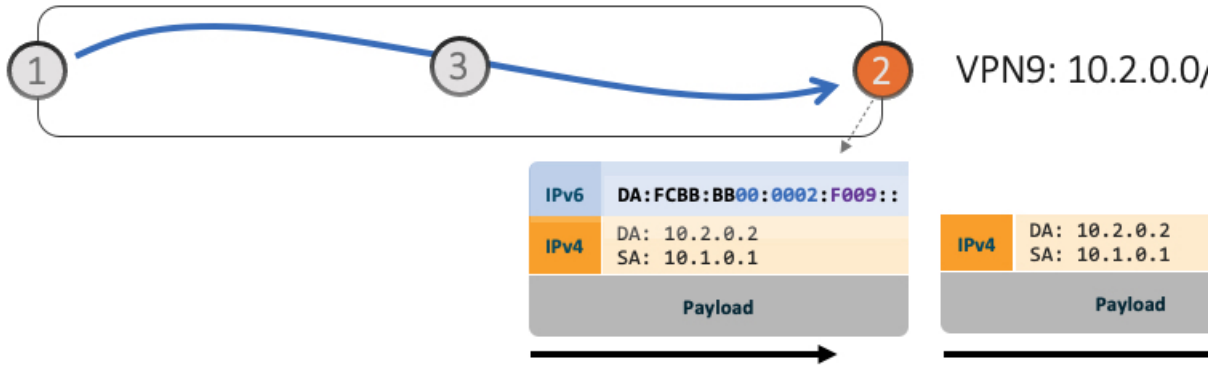


4. SRv6 allows for seamless deployment where any transit node (SRv6-capable or not) simply routes based on a /48 longest prefix match lookup.

   For example, transit node (Node 3) forwards traffic along the Algo 0 (min-cost) shortest path for the remote prefix FCBB:BB00:0002::/48.

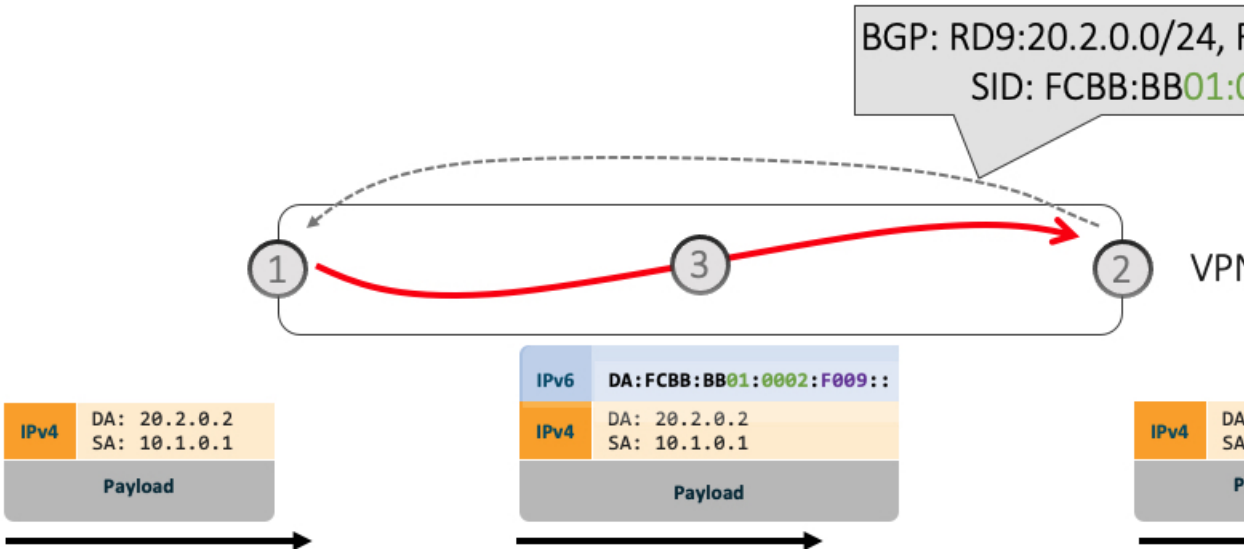


5. Egress PE (Node 2) matches local SID FCBB:BB00:0002:F009/64. Node 2 applies “VPN Decaps” behavior into VRF9 by removing the IPv6 encapsulation and looking up the payload’s DA on the corresponding VPN table.



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.



SRv6 policy counters

Policy counters are metrics used in networking that

- measure and report traffic statistics for specific policies,
- enable network administrators to monitor and manage network performance, and
- support capacity planning by providing detailed usage insights.

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 policy counters (POL.CP.SL.INT.E)	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q200, P100]) Network administrators can now monitor and manage network performance, capacity planning, and traffic engineering by reviewing the policy counters (POL.CP.SL.INT.E) in SRv6-TE. This feature is enabled by default.

SRv6 policy traffic counters

POL.CP.SL.INT.E stands for Per-SR-policy, per Candidate-Path, per-Segment-List, per-interface, egress traffic counter. This counter in SRv6 is used to measure and report traffic statistics for specific paths and interfaces within a network policy. It provides granular insights into the traffic flow through different segments and interfaces of a network policy, enabling effective network management and optimization.

To optimize hardware resources, you can use the **hw-module profile cef te-tunnel highscale-no-ldp-over-te** command to enable high-scale MPLS-TE tunnel support in CEF while disabling LDP-over-TE. For more information, see *MPLS Configuration Guide for Cisco 8000 Series Routers*.

Benefits of SRv6 policy counters POL.CP.SL.INT.E

The benefits of SRv6 policy counters POL.CP.SL.INT.E are:

- Network optimization: By analyzing the detailed traffic data, network operators can make informed decisions to optimize the network, such as rerouting traffic to avoid congestion.
- Troubleshooting: In case of network issues, these counters provide the necessary data to quickly identify and resolve problems.
- Policy validation: Ensuring that the implemented SRv6 policies are functioning as intended and making adjustments as needed based on the counter data.

Verify SRv6 policy counters

The purpose of this task is to retrieve SRv6 policy counter information.

Procedure

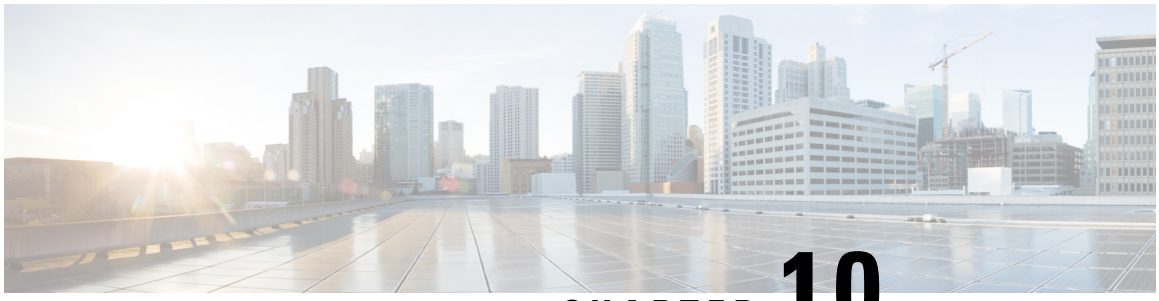
Run the following show commands to view the policy counter details.

Example:

Note that SRv6 policy prefixes will have a /64 address

```
Router#show segment-routing traffic-eng forwarding policy color 2
Fri Sep  5 12:12:46.083 UTC

SR-TE Policy Forwarding database
--------------------------------

Color: 2, End-point: 2::1
  Name: srte_c_2_ep_2::1
  Binding SID: fccc:cc06:1:e008::
  Active LSP:
    Candidate path:
      Preference: 100 (configuration)
      Name: po_r2_inetv6
    Segment lists:
      SL[0]:
        Name: r1_r2_remote_uN
        SL ID: 0xa000003
        Paths:
          Path[0]:
            Outgoing Interfaces: HundredGigE0/0/0/19
            Next Hop: fe80::2
            Switched Packets/Bytes: 1002000/296592000
            FRR Pure Backup: No
            ECMP/LFA Backup: No
            SID stack (Top -> Bottom): {}
          Path[1]:
            Outgoing Interfaces: Bundle-Ether1201
            Next Hop: fe80::2
            Switched Packets/Bytes: 998000/295408000
            FRR Pure Backup: No
            ECMP/LFA Backup: No
            SID stack (Top -> Bottom): {}
          Path[2]:
            Outgoing Interfaces: Bundle-Ether1301
            Next Hop: fe80::3
            Switched Packets/Bytes: 0/0
            FRR Pure Backup: Yes
            ECMP/LFA Backup: Yes
            SID stack (Top -> Bottom): {fccc:cc00:3:e001::/64}

  Policy Packets/Bytes Switched: 2000000/592000000


Router#show cef ipv6 accounting
fccc:cc06:1:e008::/64
Accounting: 2000000/592000000 packets/bytes output (per-prefix-per-path mode)
 via fe80::2/128, HundredGigE0/0/0/19
  path-idx 0
  next hop fe80::2/128
  Accounting: 1002000/296592000 packets/bytes output
 via fe80::2/128, Bundle-Ether1201
  path-idx 1
  next hop fe80::2/128
  Accounting: 998000/295408000 packets/bytes output
```

```
via fe80::3/128, Bundle-Ether1301
 path-idx 2
 next hop fe80::3/128
 Accounting: 0/0 packets/bytes output
```



CHAPTER 10

SRv6 Layer 3 VPN Services and Global Routing

Segment Routing over IPv6 (SRv6) provides a scalable, flexible, and programmable framework for traffic engineering and service delivery within modern network architectures. This chapter functions as a foundational guide, detailing the principles and implementation procedures for core Layer 3 VPN (L3VPN) and global routing services leveraging SRv6.

SRv6 Layer 3 Services

SRv6 Layer 3 services are a category of advanced IP routing and VPN capabilities that

- utilize SRv6 SIDs to steer traffic within a network
- enable scalable, flexible, and integrated IP routing solutions by encapsulating payloads with an outer IPv6 header addressed to an SRv6 Service SID, and
- facilitate seamless interconnection of Provider Edge (PE) nodes and VPN formation.

These services support IPv4 and IPv6 L3VPNs, BGP signaling, redundancy modes, VRF-to-VRF route leaking, and dual-stack deployment in mixed IPv4 and IPv6 environments.

Key features of SRv6 L3 services include:

- Support for dual-stack L3VPN services (IPv4 and IPv6) enabling flexible deployment in mixed IP environments.
- Use of BGP for signaling SRv6 Service SIDs, supporting iBGP, eBGP, OSPF, and static PE-CE protocols.
- Support for Equal-Cost Multi-Path (ECMP) and Unequal Cost Multipath (UCMP) routing.
- Interworking capabilities between MPLS L3VPN and SRv6 L3VPN services.
- Mechanisms for VRF-to-VRF route leaking within the SRv6 core.

- Path Maximum Transmission Unit (MTU) discovery for SRv6 encapsulated packets to optimize packet forwarding and avoid fragmentation.

SRv6 BGP-Based Services

The BGP in SRv6 Services is a routing protocol extension that

- enables the signaling and distribution of SRv6 SIDs
- supports advanced VPN and routing services over an SRv6 network, and
- facilitates the encapsulation and interconnection of provider edge routers using SRv6 service SIDs.

Extending BGP for SRv6-based overlay services

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)
- 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 for SRv6 BGP-based services

- These SRv6 BGP-based services are supported:
  - IPv4 Layer-3 VPNs

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

- uDT4 and uDT6 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 these 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.

Configure SRv6 BGP-based services

Procedure

Step 1 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.

Example:

```
Node1(config)# router bgp 100 segment-routing srv6
```

Step 2 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.

Example:

```
Node1(config)# router bgp 100 segment-routing srv6 locator Node1-locator
```

IPv4 L3VPNs over SRv6

IPv4 L3VPNs over SRv6 is an SRv6 L3VPN service that

- provides Layer 3 VPN connectivity specifically for IPv4 traffic
- utilizes an SRv6 network as the underlying transport, and

- enables enterprise IPv4 internet connectivity.

This service allows operators to establish Layer 3 VPNs for IPv4 traffic using a SRv6 infrastructure. It leverages advanced routing capabilities to ensure efficient and scalable connectivity, particularly for enterprise environments requiring internet access through VPNs. It supports VPNv4 services, including route leaking between Global Routing Tables (GRT) and Virtual Routing and Forwarding (VRF) instances.

SRv6 VPN BGP route leaking

SRv6 VPN BGP route leaking is a BGP routing mechanism that

- enables the controlled exchange of routing information between Global Routing Tables (GRT) and Virtual Routing and Forwarding (VRF) instances within an SRv6 network
- facilitates VPNv4 services, and
- is essential for providing enterprise IPv4 internet connectivity by allowing specific routes to be shared across different routing contexts.

Table 21: Feature History Table

Feature Name	Release	Description
SRv6 VPN BGP Route Leaking	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SRv6 VPN BGP Route Leaking	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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 service allows operators to establish Layer 3 VPNs for IPv4 traffic using a SRv6 infrastructure. It leverages advanced routing capabilities to ensure efficient and scalable connectivity, particularly for enterprise environments requiring internet access through VPNs. It supports VPNv4 services, including route leaking between Global Routing Tables (GRT) and Virtual Routing and Forwarding (VRF) instances.

Key concepts of SRv6 VPN BGP route leaking

- Global Routing Table (GRT): The Global Routing Table (GRT) is the main routing table within a router, containing all routes that are not associated with a specific VPN or VRF. It is used for forwarding traffic that is not part of any VPN.
- Virtual Routing and Forwarding (VRF) Instance: A VRF instance is a virtual router within a physical router, allowing multiple independent routing tables to coexist on the same device. Each VRF instance maintains its own routing table, forwarding table, and interfaces, providing logical separation for different VPNs or customer networks.

Per-VRF-46 allocation mode

Per-VRF-46 allocation mode is an SRv6 SID allocation mode that

- is configured within BGP under an address family
- assigns a specific uDT46 SID to various types of routes when advertised to remote peers, and
- aggregates multiple routes into a single entity for forwarding decisions, improving network efficiency and scalability.

Table 22: Feature History Table

Feature Name	Release	Description
Per-Prefix SRv6 Locator Assignment	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release	Description
Per-Prefix SRv6 Locator Assignment	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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.

Dual-stack L3VPN services for SRv6 Micro-SID on IPv4 and IPv6

Dual-stack L3VPN services is an SRv6 L3VPN capability that

- supports dual-stack (VPNv4/VPNv6) VRFs
- enables SRv6 L3VPN service for both IPv4 (uDT4) and IPv6 (uDT6) on the same interface, sub-interface, or VRF, and
- allows operators to simultaneously and independently access IPv4 and IPv6 without protocol translation, ensuring high processing efficiency and zero information loss.

Table 23: Feature History Table

Feature Name	Release	Description
Dual-Stack L3VPN Services (IPv4, IPv6) (SRv6 Micro-SID)	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Dual-Stack L3VPN Services (IPv4, IPv6) (SRv6 Micro-SID)	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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.

This feature is designed to provide comprehensive L3VPN connectivity over an SRv6 network for environments that require seamless handling of both IPv4 and IPv6 traffic.

Configure SRv6-based IPv4 L3VPN

To enable SRv6-based L3VPN services for IPv4 traffic by configuring SRv6 under BGP, specifying locators, and defining SID allocation modes.

SRv6-based L3VPNs allow for flexible and scalable IPv4 VPN services over an SRv6 network. The assignment of SRv6 locators and SID allocation modes can be configured at various levels within the BGP configuration.

Procedure

Step 1 Assign an SRv6 locator under BGP.

- 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 configuration

```
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.

  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 configuration

```
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
```

- 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:

  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, with per-CE 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-ce
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-ce
   !
  !
 !
```

```
!
```

- Use Case 4: Assigning SRv6 Locator for a specific prefix

  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.

  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)#
```

  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 configured under BGP is used to allocate the SID. If the default locator is not configured, then the SID will not be allocated.
  - Per-CE allocation mode always uses the default locator configured under BGP to allocate the SID.

Step 2 Verify that the local and received SIDs have been correctly allocated under VPNv4 and specific VRF (vrf_cust1).

Example:

```
Node1# show bgp vpnv4 unicast local-sids
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 50
BGP NSR Initial initsync version 18 (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            Local Sid                                Alloc mode   Locator
Route Distinguisher: 8:8
```

```
*>i8.8.8.8/32         NO SRv6 Sid                                 -           -
* i                   NO SRv6 Sid                                 -           -
Route Distinguisher: 1.1.1.1:0 (default for vrf vrf_cust1)
*> 1.1.1.0/24         fc00:0:1:40::                               per-vrf     locator1
*> 2.2.2.0/24         fc00:8:1:40::                               per-vrf     locator2
*> 3.3.3.0/24         fc00:9:1:40::                               per-vrf     locator4
*> 4.4.4.0/24         fc00:9:1:41::                               per-ce      locator4
*> 1.1.1.5/32         NO SRv6 Sid                                 -           -
*> 3.3.3.3/32         NO SRv6 Sid                                 -           -
*>i8.8.8.8/32         NO SRv6 Sid                                 -           -
```

```
Node1# show bgp vpnv4 unicast received-sids
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 50
BGP NSR Initial initsync version 18 (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                          Received Sid
Route Distinguisher: 8:8
*>i8.8.8.8/32         10.1.1.2                          fc00:0:2:42::
* i                   2400:2020:42:2fff::1
                                                        fc00:0:2:42::
Route Distinguisher: 1.1.1.1:0 (default for vrf vrf_cust1)
*> 1.1.1.0/24         11.1.1.2                          NO SRv6 Sid
*> 2.2.2.0/24         11.1.1.2                          NO SRv6 Sid
*> 3.3.3.0/24         11.1.1.2                          NO SRv6 Sid
*> 4.4.4.0/24         11.1.1.2                          NO SRv6 Sid
*> 1.1.1.5/32         11.1.1.2                          NO SRv6 Sid
*> 3.3.3.3/32         13.2.2.2                          NO SRv6 Sid
*>i8.8.8.8/32         10.1.1.2                          fc00:0:2:42::
```

```
Node1# show bgp vrf vrf_cust1 local-sids
BGP VRF vrf_cust1, state: Active
BGP Route Distinguisher: 1.1.1.1:0
VRF ID: 0x60000004
BGP router identifier 1.1.1.1, local AS number 1
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000013   RD version: 37
BGP main routing table version 37
BGP NSR Initial initsync version 18 (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           Local Sid                                 Alloc mode   Locator
Route Distinguisher: 1.1.1.1:0 (default for vrf vrf_cust1)
*> 1.1.1.0/24         fc00:0:1:40::                             per-vrf      locator1
*> 2.2.2.0/24         fc00:8:1:40::                             per-vrf      locator2
*> 3.3.3.0/24         fc00:9:1:40::                             per-vrf      locator4
*> 4.4.4.0/24         fc00:9:1:41::                             per-ce       locator4
*> 1.1.1.5/32         NO SRv6 Sid                               -            -
*> 3.3.3.3/32         NO SRv6 Sid                               -            -
```

```
*>i8.8.8.8/32         NO SRv6 Sid                                 -            -

Node1# show bgp vrf vrf_cust1 received-sids
BGP VRF vrf_cust1, state: Active
BGP Route Distinguisher: 1.1.1.1:0
VRF ID: 0x60000004
BGP router identifier 1.1.1.1, local AS number 1
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000013   RD version: 37
BGP main routing table version 37
BGP NSR Initial initsync version 18 (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                            Received Sid
Route Distinguisher: 1.1.1.1:0 (default for vrf vrf_cust1)
*> 1.1.1.0/24         11.1.1.2                            NO SRv6 Sid
*> 2.2.2.0/24         11.1.1.2                            NO SRv6 Sid
*> 3.3.3.0/24         11.1.1.2                            NO SRv6 Sid
*> 4.4.4.0/24         11.1.1.2                            NO SRv6 Sid
*> 1.1.1.5/32         11.1.1.2                            NO SRv6 Sid
*> 3.3.3.3/32         13.2.2.2                            NO SRv6 Sid
*>i8.8.8.8/32         10.1.1.2                            fc00:0:2:42::
```

Configure per-VRF-46 label allocation mode

Procedure

Assign SRv6 locator

- 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
```

- 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
```

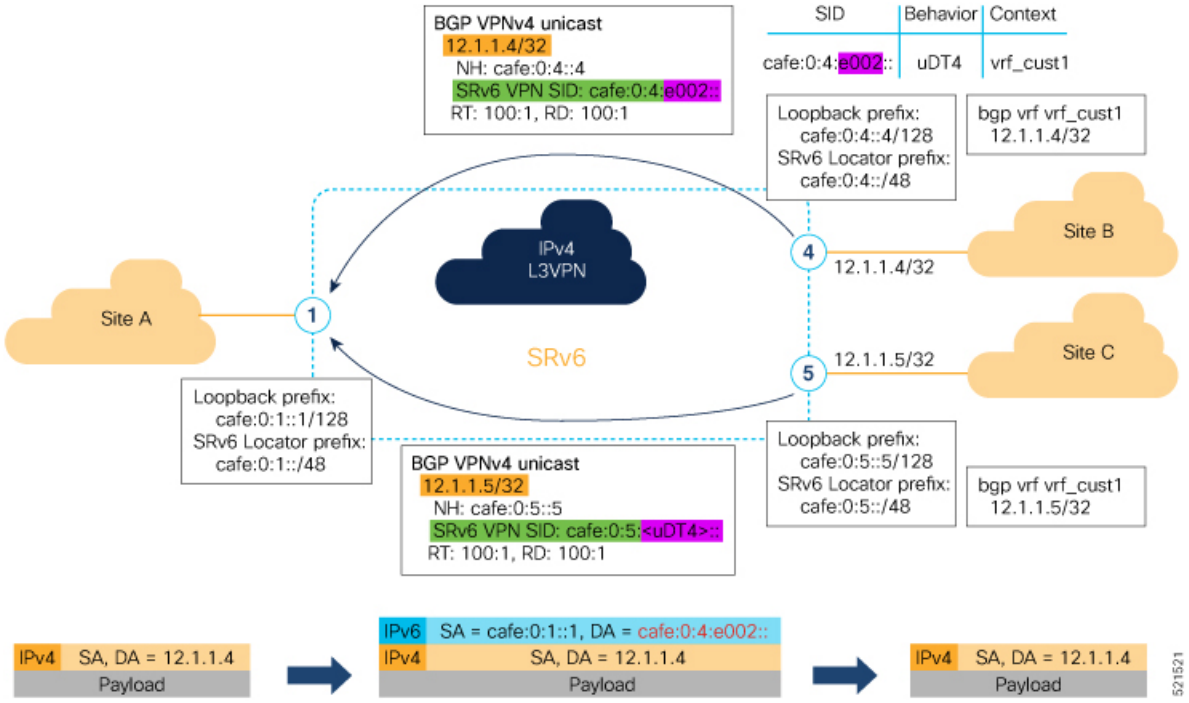
- Configure the SRv6 locator for an individual VRF for both IPv4 and IPv6 address families, with per-VRF-46 label allocation mode:

```
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
```

Verify the SRv6 based L3VPN configuration

To confirm that the SRv6-based L3VPN services for IPv4 traffic are correctly configured and operating as expected. This task helps ensure that SRv6 locators, SID allocation modes, and routing information are properly established.

This verification task is typically performed after configuring SRv6-based IPv4 L3VPNs. The examples provided in this task illustrate a common VPNv4 scenario where router Node1 acts as the Ingress Provider Edge (PE), and routers Node4 and Node5 function as Egress PEs.



Procedure

Step 1 Verify the SRv6 SID information using the **show segment-routing srv6 sid** command.

Example:

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, and uDX4 represents Endpoint with decapsulation and IPv4 cross-connect.

```
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
```

Step 2 Verify the detailed SRv6 SID information for a specific SID using the **show segment-routing srv6***SID-prefix***detail** command.

Example:

```
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)
```

Step 3 Verify the BGP VPNv4 unicast summary using the **show bgp vpnv4 unicast** commands on Egress PE.

Example:

```
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
```

Step 4 Verify the BGP VPNv4 unicast routes for a specific Route Distinguisher (RD) using the **show bgp vpnv4 unicast rd***route-distinguisher prefix* command on Ingress PE.

Example:

Step 5 Verify the SRv6 based L3VPN configuration using the **show route vrf** commands.

Example:

```
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.
```

Step 6 Verify CEF information for a VRF using the **show cef vrf** commands.

Example:

```
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
```

Step 7 Verify the BGP prefix information for VRF instances using the **show bgp vrf** commands:

Example:

```
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
```

IPv6 L3VPN services over SRv6 networks

IPv6 L3VPN is an SRv6 L3 service that

- provides IPv6 Layer 3 Virtual Private Networks (VPNv6) over an SRv6 network
- enables egress Provider Edge (PE) routers to signal an SRv6 SID with the BGP overlay service route, and
- allows ingress PEs to encapsulate IPv4/IPv6 payloads in an outer IPv6 header, using the SRv6 service SID as the destination address.

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: IPv6 L3VPN	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

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.

Restrictions of SRv6-based IPv6 L3VPN

SRv6 locator assignment and allocation guidelines

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF, or per-prefix.
- Per-VRF allocation mode is supported (uDT6 behavior)

Supported Network Capabilities

- 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.
- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.
- Per-CE allocation mode is not supported (uDX6 behavior)

Configure SRv6-based IPv6 L3VPN

To enable SRv6-based L3VPN in BGP, you must activate SRv6, specify the locator, and set the SID allocation mode. SRv6 locators can be assigned at the service, address-family, or global BGP level, with inheritance rules determining which locator is used for SID allocation.

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.

Procedure

Step 1 Enable SRv6-based L3VPN by activating SRv6 under BGP.

Specify the appropriate SRv6 locator, and configure the SID allocation mode according to your network requirements.

- Use Case 1: Assign 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: Assign SRv6 locator for all VRFs.

  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: Assign SRv6 locator for a specific VRF.

  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
```

This example shows how to configure the SRv6 locator for an Individual VRF, with per-CE 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-ce
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-ce
   !
  !
 !
!
end
```

Step 2 Verify that the local and received SIDs have been correctly allocated under VPNv6 and specific VRF (vrf_cust6).

Example:

```
Node1# show bgp vpnv6 unicast local-sids

BGP router identifier 1.1.1.1, local AS number 1
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 50
BGP NSR Initial initsync version 18 (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            Local Sid                                 Alloc mode   Locator
```

```
Route Distinguisher: 8:8
*>i3008::8:8:8:8/128  NO SRv6 Sid                                  -           -
* i                    NO SRv6 Sid                                  -           -
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::1:1:1:1/128  fc00:0:1:40::                                per-vrf     locator1
*> 3001::2:2:2:2/128  fc00:8:1:40::                                per-vrf     locator2
*> 3001::3:3:3:3/128  fc00:9:1:40::                                per-vrf     locator4
*> 3001::4:4:4:4/128  fc00:9:1:41::                                per-ce      locator4
*> 3001::5:5:5:5/128  NO SRv6 Sid                                  -           -
*> 3001::12:1:1:5/128 NO SRv6 Sid                                  -           -
*>i3008::8:8:8:8/128  NO SRv6 Sid                                  -           -
```

```
Node1# show bgp vpnv6 unicast received-sids

BGP router identifier 1.1.1.1, local AS number 1
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 50
BGP NSR Initial initsync version 18 (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                            Received Sid
Route Distinguisher: 8:8
*>i3008::8:8:8:8/128  10.1.1.2                            fc00:0:2:42::
* i                    2400:2020:42:2fff::1
                                                          fc00:0:2:42::
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::1:1:1:1/128  11.1.1.2                            NO SRv6 Sid
*> 3001::2:2:2:2/128  11.1.1.2                            NO SRv6 Sid
*> 3001::3:3:3:3/128  11.1.1.2                            NO SRv6 Sid
*> 3001::4:4:4:4/128  11.1.1.2                            NO SRv6 Sid
*> 3001::5:5:5:5/128  11.1.1.2                            NO SRv6 Sid
*> 3001::12:1:1:5/128 13.2.2.2                            NO SRv6 Sid
*>i3008::8:8:8:8/128  10.1.1.2                            fc00:0:2:42::
```

```
Node1# show bgp vrf vrf_cust6 local-sids
BGP VRF vrf_cust6, state: Active
BGP Route Distinguisher: 1.1.1.1:0
VRF ID: 0x60000004
BGP router identifier 1.1.1.1, local AS number 1
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000013   RD version: 37
BGP main routing table version 37
BGP NSR Initial initsync version 18 (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            Local Sid                                 Alloc mode   Locator
Route Distinguisher: 8:8
*>i3008::8:8:8:8/128  NO SRv6 Sid                                  -           -
* i                    NO SRv6 Sid                                  -           -
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::1:1:1:1/128  fc00:0:1:40::                                per-vrf     locator1
*> 3001::2:2:2:2/128  fc00:8:1:40::                                per-vrf     locator2
```

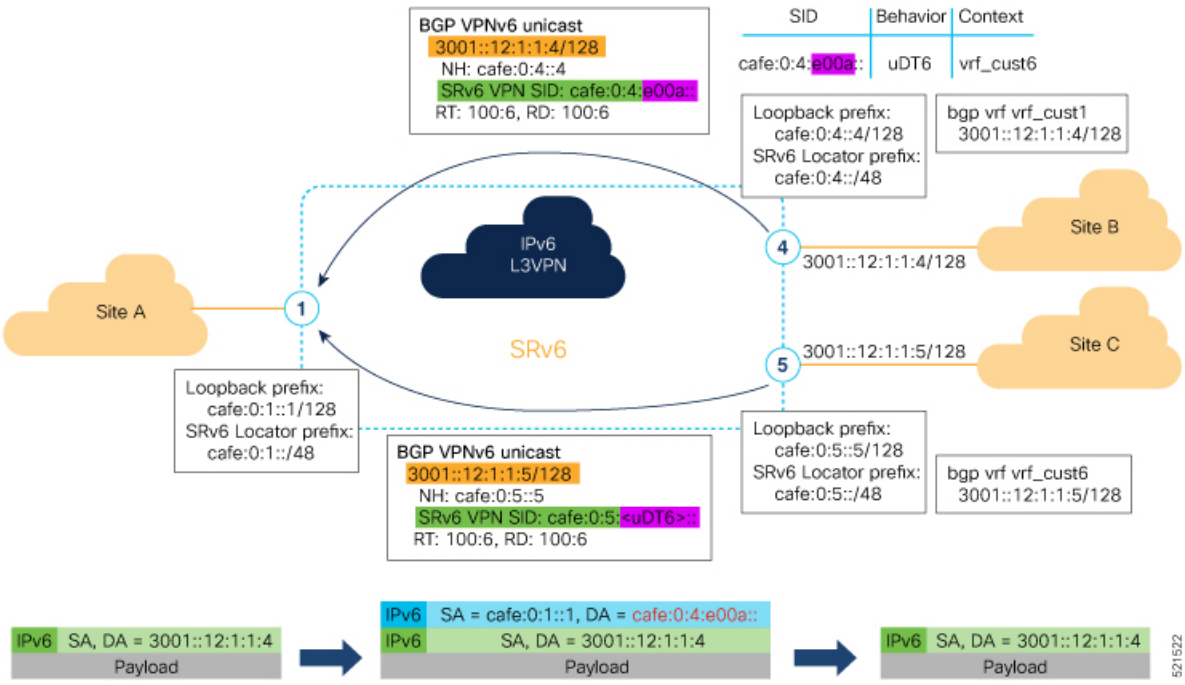
```
*> 3001::3:3:3:3/128  fc00:9:1:40::                                per-vrf      locator4
*> 3001::4:4:4:4/128  fc00:9:1:41::                                per-ce       locator4
*> 3001::5:5:5:5/128  NO SRv6 Sid                                  -            -
*> 3001::12:1:1:5/128 NO SRv6 Sid                                  -            -
*>i3008::8:8:8:8/128  NO SRv6 Sid                                  -            -


Node1# show bgp vrf vrf_cust6 received-sids
BGP VRF vrf_cust6, state: Active
BGP Route Distinguisher: 100:6
VRF ID: 0x60000004
BGP router identifier 1.1.1.1, local AS number 1
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000013   RD version: 37
BGP main routing table version 37
BGP NSR Initial initsync version 18 (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                            Received Sid
Route Distinguisher: 100:6 (default for vrf vrf_cust6)
*> 3001::1:1:1:1/128  11.1.1.2                            NO SRv6 Sid
*> 3001::2:2:2:2/128  11.1.1.2                            NO SRv6 Sid
*> 3001::3:3:3:3/128  11.1.1.2                            NO SRv6 Sid
*> 3001::4:4:4:4/128  11.1.1.2                            NO SRv6 Sid
*> 3001::5:5:5:5/128  11.1.1.2                            NO SRv6 Sid
*> 3001::12:1:1:5/128 13.2.2.2                            NO SRv6 Sid
*>i3008::8:8:8:8/128  10.1.1.2                            fc00:0:2:42::
```

What to do next

Verify SRv6-based IPv6 L3VPN configuration, on page 130

Verify SRv6-based IPv6 L3VPN configuration

This example involves a VPNv6 scenario where Node1 (Ingress PE) and Node4/Node5 (Egress PEs) are configured for SRv6-based L3VPN. The provided command sequence demonstrates how to verify SRv6 L3VPN configurations for a specific VRF, using per VRF label allocation mode.



Procedure

Step 1 Observe the uDT6 SID associated with the IPv6 L3VPN, where uDT6 behavior represents Endpoint with decapsulation and IPv6 table lookup.

Example:

```
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
```

Step 2 Verify the SRv6 based L3VPN configuration using the **show bgp vpnv6 unicast** commands on the Ingress PE.

Example:

```
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
```

Step 3 Verify the BGP prefix information for VRF instances.

Example:

```
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
```

Step 4 Verify the current routes in the Routing Information Base (RIB).

Example:

```
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.
```

Step 5 Verify the current IPv6 Cisco Express Forwarding (CEF) table.

Example:

```
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
```

IPv4 BGP Global SRv6 services

IPv4 BGP global is a BGP service that operates within an IPv4 network, and is supported by SRv6-based services through the implementation of uDX4, uDT4, and uDT46 SRv6 functions at the PE node.

Key concepts

- IPv4 global BGP: Refers to BGP routing specifically for IPv4 addresses within the global routing table, typically used for internet-wide connectivity rather than within a Virtual Routing and Forwarding (VRF) instance.
- uDX4, uDT4, and uDT46 SRv6 Functions: These are specific SRv6 Segment Identifiers (SIDs) or behaviors that define how a packet is processed at an SRv6-enabled node.
  - uDX4 (Endpoint with decapsulation and IPv4 lookup): This behavior indicates that the PE node decapsulates the SRv6 header and then performs an IPv4 lookup in its routing table to forward the packet.
  - uDT4 (Endpoint with decapsulation and IPv4 table lookup): This behavior is similar to uDX4, specifically indicating a lookup in an IPv4 routing table.
  - uDT46 (Endpoint with decapsulation and IPv4/IPv6 table lookup): This behavior supports decapsulation and a lookup in either an IPv4 or IPv6 routing table, enabling dual-stack forwarding.

Restrictions of IPv4 BGP Global SRv6 Services

- 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.
-

Configure BGP global IPv4 Over SRv6 with Per-CE SID allocation mode

Procedure

Configure BGP global IPv4 over SRv6 with per-CE SID allocation.

Example:

```
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
  !
 !
!
```

Configure per-VRF-46 label allocation mode

Procedure

Assign SRv6 locator

- 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
```

- 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
```

- 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
```

IPv6 BGP global SRv6 services

IPv6 BGP global is a SRv6 feature that extends the support of SRv6-based BGP services to include IPv6 global BGP, and implements uDT6 and uDT46 SRv6 functions at the PE node.

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: BGP Global IPv6	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
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.

Key concepts

- IPv6 global BGP: Refers to BGP routing specifically for IPv6 addresses within the global routing table, typically used for internet-wide connectivity.
- uDT46 (Endpoint with decapsulation and IPv4/IPv6 table lookup): An SRv6 behavior that supports decapsulation and a lookup in either an IPv4 or IPv6 routing table, enabling dual-stack forwarding.
- SID allocation mode: Defines how SRv6 SIDs are assigned for prefixes.
  - per-VRF: Specifies that the same label is used for all routes advertised from a unique Virtual Routing and Forwarding (VRF) instance.
  - per-VRF-46: Specifies that the same service SID (uDT46 behavior) is used for all routes advertised from a unique VRF, supporting both IPv4 and IPv6.
  - per-CE (Per-Customer Edge): An allocation mode where a unique SID is allocated for each Customer Edge device.
  - route-policy: Uses a route policy to determine the SID allocation mode and locator for a given prefix.

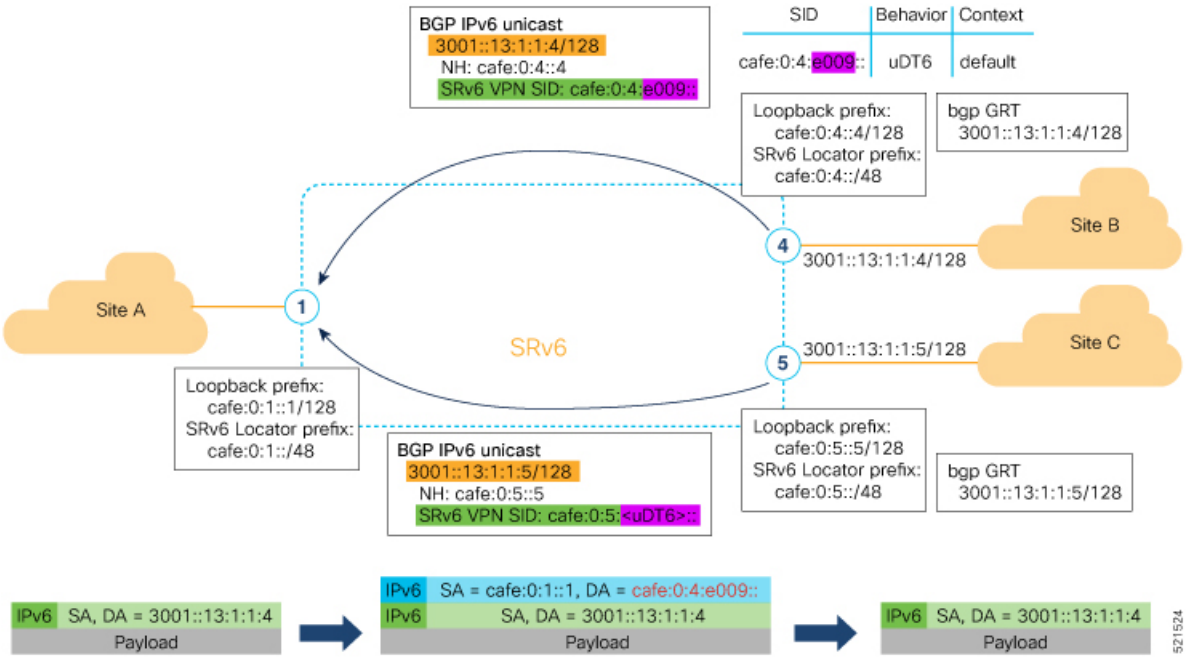
Restrictions of IPv6 BGP global SRv6 services

- 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.

Configure IPv6 BGP global SRv6 services

The figure shows a IPv6 BGP global scenario.

Figure 17: IPv6 BGP global scenario



Procedure

Step 1 Configure BGP global IPv6 over SRv6.

- 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
```

Step 2 Verify the BGP global IPv6 configuration.

The sequence of commands included correspond to router Node1 acting as Ingress PE, and routers Node4 and Node5 acting as Egress PEs.

a) Verify the BGP global IPv6 configuration using the **show bgp ipv6 unicast** commands.

Example:

```
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):
```

b) Verify the current routes in the Routing Information Base (RIB):

Example:

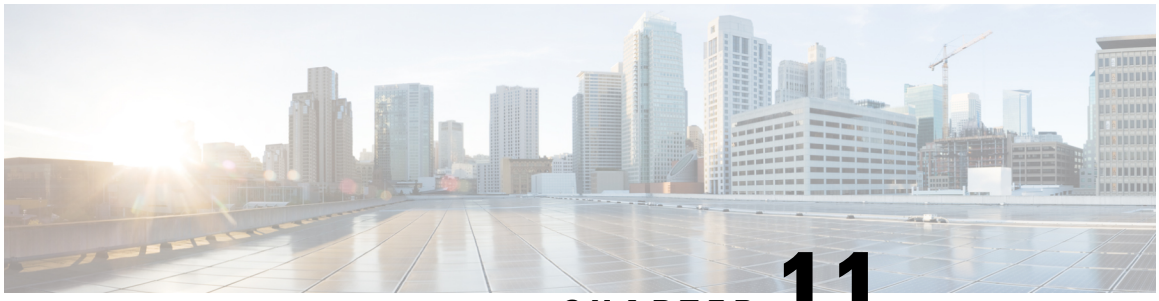
```
Node1# show route ipv6 3001::13:1:1:4/128
Fri Jan 29 19:53:26.839 UTC

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.
```

c) Verify the current IPv6 Cisco Express Forwarding (CEF) table:

Example:

```
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
```



CHAPTER 11

Advanced SRv6 Layer 3 features

As modern networks demand greater agility, efficiency, and seamless coexistence with diverse technologies, SRv6 offers a rich set of advanced functionalities to meet these challenges.

This chapter delves into specialized mechanisms that allow SRv6 to integrate smoothly into existing network infrastructures, providing flexible migration paths and operational continuity. We will examine features designed to optimize resource utilization and streamline traffic management across complex network segments. Furthermore, the chapter will cover advanced tools for network visibility and diagnostics, enabling deeper insights into network behavior and performance. By understanding these advanced features, you will gain the expertise to leverage SRv6 for highly adaptable, resilient, and future-proof network deployments, ensuring interoperability and maximizing operational effectiveness.

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

BGP Signaling for co-existence of IP routes with or without SRv6 SID is an SRv6 feature that supports the coexistence of IP routes with or without SRv6 SID over an SRv6-enabled core network, enabling the integration of SRv6 capabilities into existing network infrastructures without completely replacing IP routing.

SRv6 relax-SID is a BGP encapsulation type that allows the advertisement of prefixes with or without SRv6 SID over the same BGP session.

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
BGP Signaling for co-existence of IP routes	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
BGP Signaling for co-existence of IP routes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
BGP Signaling for co-existence of IP routes	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100]; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) 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**

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.

Procedure

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

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.

```
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 Provider Edge Lite support

SRv6 Provider Edge (PE) Lite support is an SRv6 feature that

- provides VPN de-multiplexing-only behaviors (End.DT4/DT6/DT46) at an SRv6 PE node
- allows for a lightweight-PE implementation by not supporting VPN encapsulation, and
- allows for a lightweight-PE implementation (no VPN encapsulation), and
- leverages SRv6 programmability to steer SRv6-encapsulated traffic across an SR-MPLS backbone after performing a VPN lookup.

Benefits of SRv6 Provider Edge Lite support

- Lightweight-PE implementation: It allows for a simplified PE role by only supporting VPN de-multiplexing behaviors and not VPN encapsulation.
- Efficient traffic steering: It effectively steers SRv6-encapsulated traffic across an SR-MPLS backbone after performing a VPN lookup.
- High availability: Achieved through the use of Anycast SRv6 locators, ensuring that traffic can still be handled by other nodes if a specific SRv6 PE lite node fails.
- SLA-Driven transport: It enables the provision of desired transport Service Level Agreements (SLAs) to applications by leveraging BGP VPN overlay routes with color extended communities and SR-TE policies.
- Interoperability: It facilitates the carrying of SRv6-encapsulated traffic over IP-only metro domains and SR-MPLS backbones, bridging different network technologies and domains.

How SRv6 Provider Edge Lite processes traffic

SRv6 Provider Edge (PE) Lite is a mechanism that leverages SRv6 programmability to seamlessly steer service traffic from an IPv6 domain across an SR MPLS (non-SRv6) backbone. This process ensures that traffic adheres to specified Service Level Agreements (SLAs) even when traversing heterogeneous network segments.

Summary

SRv6 Provider Edge (PE) Lite is a mechanism that leverages SRv6 programmability to seamlessly steer service traffic from an IPv6 domain across an SR MPLS (non-SRv6) backbone. This process ensures that traffic adheres to specified Service Level Agreements (SLAs) even when traversing heterogeneous network segments.

Workflow

These stages describe how SRv6 PE Lite steers traffic across an SR MPLS backbone:

1. Control Plane Setup and Signaling:
    - SRv6 PE Lite nodes are configured with SRv6 locators and explicit service de-multiplexing end-point behaviors (e.g., End.DT46 functions).

- Data center gateways advertise prefixes into the backbone via multiprotocol BGP, including color extended communities to indicate desired transport SLAs.
- SRv6 PE Lite nodes acting as SR-TE head-ends establish SR policies associated with specific colors and egress PE end-points for steering.

2. Ingress Data Center Gateway Encapsulation:
   - Data center gateways receive traffic destined for a remote data center.
   - The Data Center Gateway encapsulates this traffic into an outer IPv6 header, where the destination address is an SRv6 network program (e.g., `FCBB:BB00:100:FE01::`) determined by a gateway controller for a desired transport SLA.
3. Metro Domain Transport:

   Transit nodes in the IP-only metro domain perform a longest-prefix-match lookup for the outer IPv6 destination address and forward the packet along the shortest path to the SRv6 PE Lite node.
4. SRv6 PE Lite Decapsulation and VPN Lookup:
   - The SRv6 PE Lite node receives the encapsulated traffic.
   - The SRv6 PE Lite node removes the SRv6 encapsulation and performs a lookup of the original encapsulated packet's IP destination address in the routing table of the corresponding MPLS VPN (e.g., VRF 200).
5. SRv6 PE Lite Label Imposition and Steering:
   - Based on the VPN lookup and associated SLA, the SRv6 PE Lite node imposes the necessary MPLS VPN label and additional transport labels/SIDs.
   - This steers the traffic over the native LSP path for best-effort traffic or over an SR policy path for SLA-specific traffic.
6. SR MPLS Backbone Transport

   Transit nodes in the SR MPLS backbone forward the traffic based on the imposed MPLS labels or SIDs, directing it towards the egress PE.
7. Egress PE Decapsulation and Delivery:
   - The Egress PE receives the traffic from the backbone.
   - The Egress PE performs final decapsulation and VPN table lookup, then delivers the original packet to its destination in the remote data center.

Configuration for SRv6 PE Lite Node

Procedure

Step 1 Configure SRv6.

Example:

```
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
   !
  !
 !
!
```

Step 2 Configure IGP instance in core with SR MPLS enabled and prefix SID assigned to Loopback0.

Example:

```
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
  !
 !
!
```

Step 3 Configure interface Loopback0.

Example:

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

Step 4 Configure the SR policy.

Example:

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

Step 5 Configure the VRF (dual-stack IPv4/IPv6).

Example:

```
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
!
```

Step 6 Configure BGP.

Example:

```
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
  !
 !
!
```

Explicit End DT46 SRv6 SIDs

Explicit End.DT46 SRv6 SIDs is a SRv6 feature that allows network administrators to configure specific Segment Identifiers (SIDs) associated with SRv6-based L3VPN/Internet BGP services, ensuring these SIDs are persistent over reloads and restarts, unlike dynamically allocated SIDs.

Table 27: Feature History Table

Feature Name	Release	Description
Support for End.DT46 SRv6 Endpoint Behavior	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Support for End.DT46 SRv6 Endpoint Behavior	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Support for Explicit End.DT46 SRv6 SIDs	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release	Description
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.

Multiple explicit uDT46 IDs allocation examples

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.

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
```

Configure explicit SRv6 uSID allocation start range

Use this task to modify the starting value for the explicit Local ID Block (LIB) and explicit Wide-Local ID Block (W-LIB) ranges used in SRv6 uSID allocation.

SRv6 micro-SIDs (uSIDs) utilize Local ID Blocks (LIB) and Wide-Local ID Blocks (W-LIB) for allocation. The explicit LIB and W-LIB ranges are designated for manually assigned SIDs. Modifying the start of these explicit ranges can impact the number of available IDs in the dynamic LIB and W-LIB ranges. The default explicit LIB start value is

`0xfe00`, and the default explicit W-LIB start value is `0xfff7`.

Before you begin

- The `usid-f3216` SID format must be configured or enabled by default.

Procedure

Step 1 Use the **show segment-routing srv6 manager** command to display the default LIB and W-LIB start values.

Example:

```
Router# 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)

. . .
```

Step 2 Use the **segment-routing srv6 formats format usid-f3216 usid wide-local-id-block explicit start** command to modify the start value for the explicit W-LIB.

Example:

```
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
```

Step 3 Use the **show segment-routing srv6 manager** command tto display the configured explicit LIB and W-LIB starting values.

Example:

```
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 explicit End DT46 SRv6 SIDs

This task explains how to configure explicit End.DT46 SRv6 Segment Identifiers (SIDs) to enable L3VPN/Internet BGP services with persistent SIDs.

Procedure

Step 1 Configure explicit uDT46 IDs allocated from the LIB or W-LIB range.

Example:

```
Router(config)# segment-routing
Router(config-sr)# srv6
Router(config-srv6)# static
Router(config-srv6-static)# endpoint
Router(config-srv6-static-endpoint)# sid fcbb:bb00:10:fe0a:: behavior end-udt46
Router(config-srv6-static-sid)# allocation-context vrf VRF-A
Router(config-srv6-static-sid)# forwarding
Router(config-srv6-static-sid)# exit
Router(config-srv6-static-endpoint)# sid fcbb:bb00:11:fff7:d:: behavior end-udt46
Router(onfig-srv6-static-sid)# allocation-context vrf VRF-D
```

Step 2 Use the **router static address-family ipv6 unicast** command toonfigure explicit uDT46 IDs allocated from the LIB or W-LIB range,

Example:

```
RP/0/RP0/CPU0:ios(config)# router static
RP/0/RP0/CPU0:ios(config-static)# address-family ipv6 unicast
RP/0/RP0/CPU0:ios(config-static-afi)# fcbb:bb00:10:fe0a::/64 segment-routing srv6 endpoint behavior
 uDT46 vrf VRF-A
RP/0/RP0/CPU0:ios(config-static-afi)# fcbb:bb00:100:fe0a::/64 segment-routing srv6 endpoint behavior
 uDT46 vrf VRF-A
```

```
RP/0/RP0/CPU0:ios(config-static-afi)# fcbb:bb00:11:fff7:d::/80 segment-routing srv6 endpoint behavior
 uDT46 vrf VRF-D
RP/0/RP0/CPU0:ios(config-static-afi)# fcbb:bb00:100:fff7:d::/80 segment-routing srv6 endpoint behavior
 uDT46 vrf VRF-D
```

Step 3 View the running configuration.

Example:

```
router static
 address-family ipv6 unicast
  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
  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
```

SRv6 SID Information in BGP-LS Reporting

SRv6 SID Information in BGP-LS Reporting is a SRv6 feature that

- enhances BGP Link-State (BGP-LS) by extending its topology reporting capabilities
- adds the capability to report comprehensive SRv6 SID Network Layer Reachability Information (NLRI), and
- enables detailed reporting of SRv6 domain topologies.

Network Layer Reachability Information (NLRI)

NLRI is the information about network destinations (prefixes, SIDs, etc.) and how to reach them. It's essentially the routing information that a router advertises to its peers, allowing other routers to build their forwarding tables and determine paths through the network.

For example, NLRI is used to describe various types of SRv6-specific information reported by BGP-LS. These 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

You can use these commands to ping or trace an SRv6 Segment Identifier (SID):

- Ping a specific SID: `ping B:k:F::`
- Traceroute a specific SID: `traceroute B:k:F::`
- Ping or traceroute a SID using a list of packed carriers: `ping <destination SID> via srv6-carriers <list of packed carriers>traceroute <destination SID> via srv6-carriers <list of packed carriers>`

Configure SRv6 SID Information in BGP-LS Reporting

Use this task to configure IS-IS to distribute SRv6 link-state data, enabling BGP-LS to report comprehensive SRv6 Segment Identifier (SID) Network Layer Reachability Information (NLRI) within the domain.

Distributing IS-IS SRv6 link-state data to BGP-LS is essential for network visibility and for enabling diagnostic tools like SRv6 ping and traceroute to utilize the reported SID information. This configuration allows BGP-LS to build a detailed topology view that includes SRv6-specific elements.

Procedure

Configure the distribution of link-state data for the specified instance ID.

Example:

```
Router(config)# router isis 200
Router(config-isis)# distribute link-state instance-id 200
```

Path maximum transmission unit discovery for SRv6 encapsulated packets

Path maximum transmission unit (MTU) discovery for SRv6 encapsulated packets is a network mechanism that

- prevents silent packet loss and ensures efficient data transmission in SRv6 networks
- dynamically adjusts encapsulated packet sizes to match the path MTU, and
- enables routers to send ICMP error messages to the source when oversized packets are detected, addressing the critical limitation that SRv6-enabled routers do not support packet fragmentation.

Table 28: 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 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) 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. This feature is now supported on: • 8011-4G24Y4H-I
Path MTU discovery for SRv6 Packets on Ingress Provider Edge (PE) Routers, Egress (PE) Routers, and P Role Transit Nodes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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.

VRF-to-VRF route leaking in SRv6 core

VRF-to-VRF route leaking is an SRv6 feature that

- enables communication between separate Virtual Routing and Forwarding (VRF) instances
- selectively shares specific routes between these VRFs while maintaining isolation for others, and
- uses configured import and export route targets in each VRF to control the route exchange.

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
VRF-to-VRF route leaking in SRv6 core	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: Q200, P100])(select variants only*) VRF-to-VRF route leaking enables sharing of routes between VRFs while maintaining their isolation. This feature allows the source VRF to send leaked routes to remote PEs or Route Reflectors (RRs) across an SRv6 core network, similar to an MPLS core network, enabling communication between different service tenants or administrative domains without compromising VRF isolation. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

SRv6 VRF-to-VRF route leaking capabilities

SRv6 VRF-to-VRF route leaking extends MPLS-based VRF route leaking functionalities to SRv6. This feature leverages SRv6 flexibility for path selection and optimization, allowing you to configure the destination VRF to send leaked routes to a remote Provider Edge (PE) or Route Reflector (RR) across an SRv6 core. It enables communication between VRFs in an SRv6-enabled environment while maintaining control over routing and traffic engineering decisions.

Benefits of VRF-to-VRF route leaking in SRv6

The key benefits of the feature are:

- Improved traffic management: The feature allows routes in the source VRF to use SRv6 SIDs from a best-effort locator and routes in the destination VRF to use SIDs from a low-latency locator. This setup enables differentiated traffic treatment in the SRv6 core.

- Enhanced flexibility: The feature leaks routes between VRFs and advertises them as VPNv4 or VPNv6 or EVPN RT5 prefixes to remote PE routers, providing better flexibility in managing network traffic and inter-VRF communication.
- Scalability: The feature dynamically leaks routes that help to scale the network by automating the redistribution process between VRFs.
- Security and isolation: The feature uses route targets and policies to control route leaking, ensuring that it only occurs between intended VRFs, maintaining both security and isolation.

Limitations for VRF-to-VRF route leaking in SRv6

- VRF-to-VRF route leaking does not support multicast routes.
- The feature supports both SRv6 Full-length SID and Micro-SIDs.
- Depending on the destination VRF configuration, the PE router assigns SIDs to the leaked route based on the SID allocation mode, which can be per-VRF or per-VRF-46.

How SRv6 VRF-to-VRF route leaking works

VRF-to-VRF route leaking is an SRv6 feature that enables communication between separate VRF)instances by selectively sharing specific routes while keeping others isolated.

Summary

The SRv6 VRF-to-VRF route leaking process involves several key actors and components:

- Source VRF: The Virtual Routing and Forwarding instance that initiates the route import.
- Destination VRF: The Virtual Routing and Forwarding instance that receives the imported prefix, allocates an SRv6 SID, and advertises the prefix.
- SRv6 SID: The unique Segment Identifier allocated to the imported prefix for routing within the SRv6 core.
- Remote neighbors: Other devices in the SRv6 network that receive the advertised prefixes and route traffic accordingly.
- SRv6 core network: The underlying network infrastructure where SRv6 routing and traffic engineering occur.
- BGP VPNv4/VPNv6/EVPN RT5: The BGP extensions used by the destination VRF to advertise the prefixes and SIDs.

The SRv6 VRF-to-VRF route leaking process involves VRFs importing and advertising prefixes with associated SRv6 SIDs, allowing remote neighbors to efficiently route traffic to the intended destinations within the SRv6 core.

Workflow

These stages describe how SRv6 VRF-to-VRF route leaking works:

1. Route import: A source VRF imports a prefix from another destination VRF, which belongs to a different service. This action creates a route leak between them.
2. SID allocation The destination VRF allocates a unique SRv6 SID to the newly imported prefix. This SID ensures that traffic destined for the imported prefix is correctly routed within the SRv6 core. The SID allocation type depends on the destination VRF configuration (either per-VRF or per-VRF-46).
3. Prefix advertisement: The destination VRF advertises the imported prefix, along with its associated SRv6 SID, to remote neighbors in the SRv6 network. This advertisement occurs through BGP VPNv4, VPNv6, or EVPN RT5.
4. Routing within the SRv6 core: Remote neighbors receive this information and can now route traffic to the imported prefix using the SRv6 SID. The SRv6 SID facilitates efficient routing and traffic engineering, ensuring that traffic reaches the correct VRF and its ultimate destination.

Configure VRF-to-VRF route leaking in SRv6

Procedure

Step 1 Run the **export route-policy** command to configure and attach route leaking in the source VRF.

- Configure the static export Route Target to leak all prefixes to the destination VRF. In the below configuration, the leaked Route Target is 1:12.

```
Router(config)#vrf vrf-be
Router(config-vrf)#address-family ipv4 unicast
Router(config-vrf-af)#import route-target 1:10
Router(config-vrf-af)#export route-target 1:10
Router(config-vrf-af)#export route-target 1:12
Router(config-vrf-af)#commit
```

- Configure a route policy that attaches appropriate Route Target to the leaked prefixes to leak specific prefixes to the destination VRF. Apply the IF condition in the Route Policy Language (RPL) to leak specific prefixes.

```
Route(config)#prefix-set allowed-leaked-route
Route(config-pfx)#192.168.1.0/32
Router(config-pfx)#end-set
Router(config)#route-policy export-policy
Router(config-rpl)#if destination in allowed-leaked-route then
Router(config-rpl-if)#set extcommunity rt 1:12
Router(config-rpl-if)#endif
Router(config-rpl)#end-policy
Router(config)#commit
Router(config)#vrf vrf-be
Router(config-vrf)#address-family ipv4 unicast
Router(config-vrf-af)#import route-target 1:10
Router(config-vrf-af)#export route-target 1:10
Router(config-vrf-af)#export route-policy export-policy
Router(config-vrf-af)#commit
```

Step 2 Configure the destination VRF to import routes from the source VRF.

Example:

```
Router(config)#vrf vrf-ef
Router(config-vrf)#address-family ipv4 unicast
```

```
Router(config-vrf-af)#import route-target 2:2
Router(config-vrf-af)#import route-target 1:12
Router(config-vrf-af)#export route-target 2:2
```

Step 3 Run the **show running-config** command to verify the running configuration.

Example:

```
vrf vrf-be
  address-family ipv4 unicast
    import route-target 1:10
    export route-target 1:10
    export route-targer 1:12
  !
!
vrf vrf-ef
  address-family ipv4 unicast
    import route-target 2:2
    import route-target 1:12
    export route-target 2:2
  !
!
```

Step 4 Run the **import from vrf advertise-as-vpn** command to forward the imported routes to a remote PE or VPN RR peer through configuration.

Example:

```
Router(config)#vrf vrf-ef
Router(config-vrf)#address-family ipv4 unicast
Router(config-vrf-af)#import from vrf advertise-as-vpn
Router(config-vrf-af)#commit
```

Step 5 Run the **show bgp vrf** command to verify the route leaking from the source VRF vrf-be.

In the below show output, the source VRF vrf-be leaks the Route Target 1:12.

Example:

```
Router#show bgp vrf vrf-be 192.168.1.0/32 detail
Mon Aug 19 14:06:22.668 UTC
BGP routing table entry for 192.168.1.0/32, Route Distinguisher: 1.1.1.1:11
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker            3434714      3434714
    SRv6-VPN SID: fc00:1:4:fff0:7d1::/80
    Format: f3216
    Alloc Mode/Locator ID: per-vrf/1
    Flags: 0x00143001+0x01000000+0x00000000
Last Modified: Aug 19 09:53:33.351 for 04:12:49
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Flags: 0x3000000005040003+0x00, import: 0x31f
  Advertised to update-groups (with more than one peer):
    0.2
  4
    100.4.0.1 from 100.4.0.1 (193.0.0.1), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate
      Received Path ID 0, Local Path ID 1, version 3434714
      Extended community: RT:1:10 RT:1:12
```

The described show output indicates that the destination VRF vrf-ef imports the prefix from the source VRF vrf-be, as shown by the **imported** flag. The output also includes details of the source VRF. A non-zero value in the **Flags** field confirms that the prefix is imported.

Example:

```
Router#show bgp vrf vrf-ef 192.168.1.0/24 detail
Mon Aug 19 14:08:07.102 UTC
BGP routing table entry for 192.168.1.0/24, Route Distinguisher: 1.1.1.1:21
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker            3440133      3440133
    SRv6-VPN SID: fc00:2:4:fff0:7d1::/80
    Format: f3216
    Alloc Mode/Locator ID: per-vrf/2
    Flags: 0x00103001+0x01000000+0x00000000
Last Modified: Aug 19 10:48:24.351 for 03:19:42
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Flags: 0x3100000005040003+0x00, import: 0x080
  Advertised to update-groups (with more than one peer):
    0.2
  4
    100.4.0.1 from 100.4.0.1 (193.0.0.1), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 3440133
      Extended community: RT:1:10 RT:1:12 RT:1:20
      Source AFI: VPNv4 Unicast, Source VRF: vrf-be, Source Route Distinguisher: 1.1.1.1:11
```

The below show output is an example of the output on remote PE:

```
Router#show bgp vpnv4 unicast rd 1.1.1.1:21 192.168.1.0/32 detail
Fri Dec 13 13:21:29.136 PST
BGP routing table entry for 192.168.1.0/32, Route Distinguisher: 1.1.1.1:21
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                 690          690
    Flags: 0x00040028+0x00000000+0x00000000
Last Modified: Dec 13 13:14:37.000 for 00:06:52
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x2000000025060005+0x00, import: 0x31f
  Not advertised to any peer
  10
    192::1 (metric 30) from 192::4 (192.168.0.1), if-handle 0x00000000
      Received Label 0x7d10
      Origin IGP, metric 0, localpref 100, valid, internal, best, group-best, import-candidate,
not-in-vrf
      Received Path ID 1, Local Path ID 1, version 690
      Extended community:RT:2:2
      Originator: 192.168.0.1, Cluster list: 192.168.1.4
      PSID-Type:L3, SubTLV Count:1, R:0x00,
       SubTLV:
 T:1(Sid information), Sid:fcc00:2:4:fff0::(Transposed), F:0x00, R2:0x00, Behavior:61, 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
```

Dual-stack with SRv6 unicast and IPv4 multicast core

Dual-stack with SRv6 unicast and IPv4 multicast core is an SRv6 core feature that

- uses SRv6 for unicast traffic and IPv4 for multicast communication
- enables a device to simultaneously support multiple Internet Protocol (IP) versions within a network stack, typically IPv4 and IPv6, and
- allows devices to support both IPv4 and IPv6 unicast addresses concurrently, facilitating the transition to IPv6 while maintaining compatibility with existing IPv4 networks and applications.

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
Dual-stack with SRv6 unicast and IPv4 multicast core	Release 25.2.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) This feature introduces dual-stack support, enabling SRv6 for unicast traffic and IPv4 for multicast communication. The dual-stack approach combines the strengths of both IPv4 and SRv6 protocols to simplify routing, enhance interoperability, and improve overall network efficiency. SRv6 provides precise control over unicast traffic paths, while IPv4 ensures reliable support for multicast traffic. *This feature is supported on 8712-MOD-M.

Traffic distribution in dual-stack environments using SRv6 and IPv4

In a dual-stack environment with SRv6 and IPv4 configuration, SRv6 is utilized for unicast traffic, where packets are sent from a single sender to a single receiver. On the other hand, IPv4 is used for multicast communication, which involves sending packets from one sender to multiple receivers.

Benefits of dual-stack with SRv6 unicast and IPv4 multicast

The key benefits of the feature are:

- Combines the benefits of both IPv4 and SRv6 protocols, by providing a streamlined and optimized network experience for both unicast and multicast communication.
- Facilitates seamless interoperability between IPv4 and IPv6 protocols.
- Enables devices to use the optimal protocol for a specific network scenario based on the availability and performance of IPv4 and IPv6 services.
- Provides a simple solution for data transfer from IPv4 to SRv6 without requiring complex tunneling or translation mechanisms.
- Preserves the end-to-end connectivity and security of IPv6 while ensuring compatibility with the existing IPv4 infrastructure and applications.

Limitations for dual-stack with SRv6 and IPv4

- The dual-stack support is available on MVPN GRE-based profiles such as profile 0 and profile 11.
- The dual-stack supports only SRv6 micro-segments (uSIDs).
- The dual-stack uses Customer Edge (CE) label allocation for SRv6.

Enable dual-stack with SRv6 unicast and IPv4 multicast core

This section includes only the BGP configuration that is required to enable dual-stack with SRv6 unicast and an IPv4 multicast core.

Procedure

Step 1 Run the **router bgp** *as-number* **segment-routing srv6** command to enable SRv6 globally.

The *as-number* range is 1–65535.

Example:

```
Router(config)#router bgp 101
Router(config-bgp)#nsr
Router(config-bgp)#mvpn
Router(config-bgp)#bgp router-id 10.10.10.1
Router(config-bgp)#bgp graceful-restart
Router(config-bgp)#segment-routing srv6
```

Step 2 Run the **router bgp** command to configure IPv4 for multicast communication.

Example:

```
Router(config-bgp)#address-family ipv4 multicast
Router(config-bgp-af)#redistribute connected
```

Step 3 Run the **router bgp** command to configure SRv6 for unicast communication.

The **address-family ipv6 unicast** and **segment-routing srv6** configurations indicate that SRv6 is used for unicast communication.

Example:

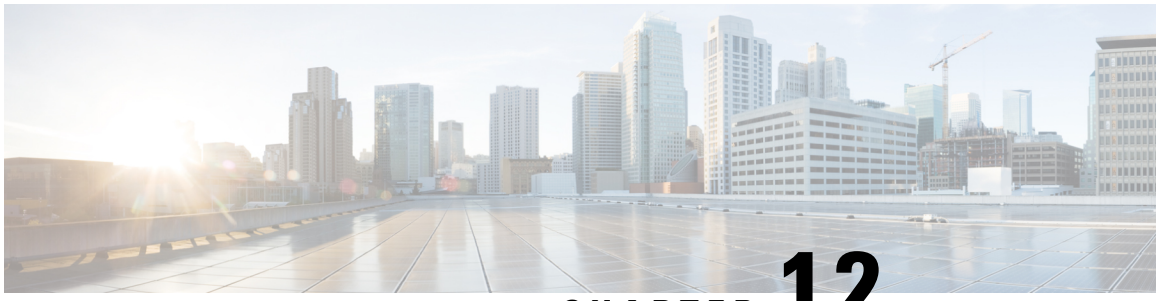
```
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#segment-routing srv6
Router(config-bgp-af)#redistribute connected
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 mvpn
Router(config-bgp)#address-family ipv4 mvpn
```

Step 4 Run the **show running-config** command to verify the running configuration.

Example:

```
router bgp 101
 nsr
 mvpn
 bgp router-id 10.10.10.1
 bgp graceful-restart
 segment-routing srv6
```

```
 !
 address-family ipv4 multicast
  redistribute connected
 !
 address-family ipv6 unicast
  segment-routing srv6
  !
  redistribute connected
 !
 address-family ipv4 mvpn
 !
 address-family ipv6 mvpn
 !
!
```



CHAPTER 12

SRv6-Based Layer 2 and Integrated VPN Services

IPv4 L3VPN active-standby redundancy using port-active mode

An IPv4 L3VPN active-standby redundancy using port-active mode is a Segment Routing IPv6 (SRv6) service that

- provides all-active per-port load balancing for multihoming
- determines traffic forwarding based on specific interfaces rather than per-flow across multiple Provider Edge (PE) routers, and
- enables faster convergence by using designated forwarder (DF) election through modulo calculation, where byte 10 of the Ethernet Segment Identifier (ESI) is used to detect the active PE router and bring down the standby PE router's interface

Enhanced multihoming efficiency with active-standby redundancy using port-active mode

This feature ensures high availability and efficient traffic management in multihomed IPv4 L3VPN networks. By leveraging per-port active-standby redundancy, it provides robust load balancing and rapid failover, minimizing service disruption. The port-active mode simplifies traffic forwarding decisions based on specific interfaces and accelerates convergence using DF election, enhancing overall network resilience and operational efficiency.

Benefits of active-standby redundancy using port-active mode

- Load balancing performed per interface (port), simplifying forwarding decisions.
- DF election based on modulo calculation of the ESI to identify the active PE.
- Standby PE interfaces disabled to avoid forwarding loops and conflicts.
- Rapid failover and convergence by re-electing the DF upon link or port failure.

- Enhanced network efficiency, reliability, and resilience in multihomed IPv4 L3VPN deployments.

Restrictions of active-standby redundancy using port-active mode

- This feature can only be configured on bundle interfaces.
- When an EVPN Ethernet segment is configured with port-active load-balancing mode, you must not configure ACs of that bundle on bridge domains that have an EVPN instance (EVI) configured.
- EVPN Layer 2 bridging service is incompatible with port-active load-balancing mode and therefore cannot be used together.

How SRv6 services for L3VPN active-standby redundancy using port-active mode work

Summary

The SRv6 Services for L3VPN Active-Standby Redundancy using Port-Active Mode process ensures high availability and efficient traffic management. PE routers exchange EVPN ES routes via BGP, then individually determine an ordered list of PEs. A modulo calculation designates one PE as the active forwarder (DF), while others disable their bundles. In the event of a failure, the active DF withdraws its route, prompting a re-election and quick resumption of forwarding duties by a newly elected DF.

Workflow

These stages describe how SRv6 services for IPv4 L3VPN active-standby redundancy using port-active mode works:

1. PE routers exchange EVPN ES routes. All PE routers connected to the multihomed ES exchange EVPN ES routes across BGP.
2. Each PE router creates an ordered list and assigns an ordinal.
   - Each PE router creates an ordered list of all PE IP addresses connected to the ES.
   - Each PE router assigns itself an ordinal based on its position in this list.
3. A modulo calculation determines the Designated Forwarder (DF).
   - Using the ordinals, a modulo calculation determines which PE becomes the Designated Forwarder (DF) for the ES.
   - For the modulo calculation, byte 10 of the Ethernet Segment Identifier (ESI) is used.
4. The elected DF forwards traffic and non-DF PEs disable bundles.
   - The PE elected as DF actively forwards traffic for the ES.
   - All other PEs (non-DF) disable their respective bundles for that ES.
5. The active DF PE withdraws its ES route upon failure.

   Upon a link or port failure, the active DF PE withdraws its ES route.

6. A new DF election is triggered, and a new PE resumes forwarding.
   - This withdrawal triggers a new DF election among all PEs servicing the ES.
   - A new PE is elected as DF and resumes forwarding duties.

Configure SRv6 services L3VPN active-standby redundancy using port-active mode

Configure SRv6 services to provide L3VPN active-standby redundancy using port-active mode under an Ethernet Segment (ES).

Procedure

Step 1 Configure Ethernet link bundle interface.

Example:

```
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
```

Step 2 Configure the physical interface to join the bundle.

Example:

```
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
```

Step 3 Configure the BGP address family session for EVPN.

Example:

```
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
```

Step 4 Running configuration active-standby redundancy using port-active mode.

Example:

```
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
  !
 !
!
```

Step 5 Verify the SRv6 services L3VPN active-standby redundancy using port-active mode configuration.

a) Verify ethernet-segment details on active DF router.

Example:

```
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
```

b) Verify bundle Ethernet configuration on active DF router.

Example:

```
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
```

c) Verify ethernet-segment details on standby DF router.

Example:

```
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
```

d) Verify bundle configuration on standby DF router.

Example:

```
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
```

IPv4 L3VPN active-active redundancy using port-active mode

An IPv4 L3VPN active-active redundancy using port-active mode is an SRv6 service that

- provides active-active connectivity to a CE device in a Layer 3 VPN (L3VPN) deployment
- allows the CE device, whether Layer 2 or Layer 3, to connect to redundant PE routers over a single Link Aggregation Control Protocol (LACP) link aggregation group (LAG) port, and
- augments Layer 3 local route learning with remote route-synchronization programming to ensure complete route awareness across redundant PEs despite bundle hashing distributing ARP or IPv6 Network Discovery packets unevenly

Key concepts

- Bundle hashing: A method that distributes traffic across multiple links in a LAG based on packet header fields, which can cause packets like ARP or IPv6 ND to be sent to different redundant routers.
- Remote route-synchronization programming: A mechanism that synchronizes Layer 3 routing information between redundant PEs to provide full route awareness

Route synchronization between service PEs

Route synchronization between service PE devices is essential to minimize service interruptions for both unicast and multicast traffic following a failure on a redundant service PE. The synchronization process uses specific EVPN route types for Layer 3 route synchronization:

- EVPN route-type 2: Synchronizes ARP tables.
- EVPN route-type 7/8: Synchronizes IGMP JOIN and LEAVE messages.

In scenarios where a Layer 3 CE router connects to redundant PEs, it may establish an IGP adjacency on the bundle port. This adjacency forms with only one of the redundant PEs, meaning that IGP customer routes are present only on that PE. To synchronize Layer-3 customer subnet routes (IP prefixes) across redundant PEs, EVPN route-type 5 is used. This route type carries the ESI, Ethernet Tag (ETAG), and the gateway address (prefix next-hop address) to ensure consistent routing information.

This synchronization mechanism ensures seamless failover and continuity of Layer 3 services in redundant PE environments

Consistent route synchronization across VRFs and Ethernet segments

Consistent route synchronization across Virtual Routing and Forwarding instances (VRFs) and Ethernet segments ensures optimized redundancy and efficient route management in the network. This synchronization maintains uniform routing information, which enhances network stability and simplifies the management of redundant paths and failover mechanisms.

- Synchronization across VRFs prevents routing inconsistencies between isolated routing domains.
- Ethernet segment synchronization supports redundancy by coordinating route updates among connected devices.
- The approach optimizes route management by reducing conflicts and improving convergence times.

- Gratuitous ARP (GARP) and IPv6 Network Advertisement (NA) replay are not necessary for CE devices connected to redundant PE devices through a single Link LAG port.
- This configuration example enables Layer 3 route synchronization for routes learned on Ethernet segment subinterfaces:

```
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 L3 EVPN services

An SRv6 L3 EVPN services is a network service that

- uses EVPN Route Type 5 (RT5) to advertise EVPN routes with IP prefixes
- provides end-to-end Layer 3 connectivity, and
- supports carrying L3VPN routes within the L2VPN EVPN RT5 address family instead of VPNv4 or VPNv6 unicast address families across an SRv6 core (EVPN over SRv6 underlay).

Interworking and BGP session for EVPN RT5 over SRv6 and MPLS cores

Interworking between EVPN RT5 over SRv6 core and EVPN RT5 over MPLS core is supported through the L3 EVPN/SRv6 and L3 EVPN/MPLS interworking gateway.

Similarly, interworking between EVPN RT5 over SRv6 core and L3VPN over MPLS core is enabled through the L3 EVPN/SRv6 and L3VPN/MPLS interworking gateway.

Limitations of SRv6 L3 EVPN services

This topic provides supported configurations for interworking EVPN RT5 with SRv6 and MPLS core.

Supported interworking

- Interworking between EVPN RT5 over an SRv6 core and EVPN RT5 over an MPLS core is supported.
- Interworking between EVPN RT5 over an SRv6 core and L3VPN over an MPLS core is supported.

BGP session configuration for route reflectors

- BGP does not support configuring both VPNv4/v6 address families and EVPN RT5 address families on the same BGP session simultaneously.

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.

VRF route advertisement

- BGP advertises all VRF routes using either the 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.

Supported behaviors for EVPN RT5 over SRv6

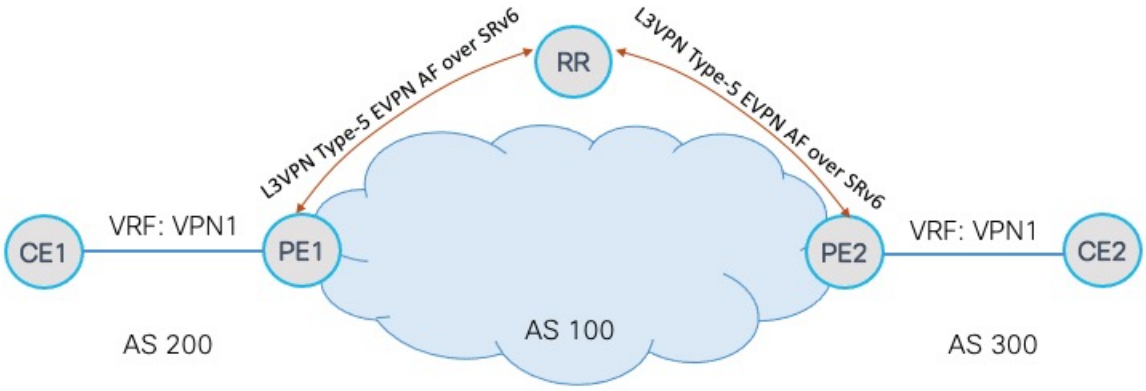
- IPv4, IPv6, and IPv4/IPv6 (dual stack) L3 EVPN over SRv6
- uDT4
- uDT6
- uDT46
- Automated Steering to Flex-Algo (BGP per-VRF locator Flex-Algo (per-prefix))
- Automated Steering to SRv6 Policy (ODN/AS)

Configure SRv6-based L3 EVPN

Enable and configure SRv6-based Layer 3 EVPN to support dual-stack IPv4/IPv6 environments.

-

Figure 18: Configuration Example: Dual Stack L3 EVPN over SRv6



Procedure

Step 1 Configure the VRF with dual-stack IPv4 and IPv6 address families.

Example:

```
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)#
```

Step 2 Configure the SRv6 locator for an individual VRF, with per-VRF label allocation mode.

Example:

```
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
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
```

Step 3 Running configuration of SRv6-based L3 EVPN

Example:

```
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
   !
  !
  neighbor 1.1.1.1
   remote-as 200
   address-family ipv4 unicast
   !
  !
  neighbor 3333::3
   remote-as 200
   address-family ipv6 unicast
   !
  !
 !
!
```

SRv6 service support

Segment Routing over IPv6 (SRv6) enables advanced network programmability and service chaining by steering packets through a network based on instructions encoded as IPv6 addresses, known as Segment IDs (SIDs). Modern SRv6 deployments leverage SID lists and SID databases (such as W-LIB) to support flexible and scalable network services, including both Layer 2 and Layer 3 offerings.

SRv6 services can utilize either remote or local SIDs, depending on service requirements and deployment models. The integration with the writable SID Library (W-LIB) further enhances service delivery by efficiently managing SID resources and enabling dynamic service provisioning.

These SRv6 services are supported:

- L2 and L3 services with remote SIDs from W-LIB
- L3 services with local SIDs from W-LIB

These capabilities enable operators to deliver versatile and scalable network services using SRv6, leveraging both remote and local SIDs managed by W-LIB.

L2 and L3 services with remote SIDs from W-LIB

An L2 and L3 service with remote SIDs from Wide Local ID Block (W-LIB) is a SRv6 feature that

- uses SRv6 to enable a headend node to receive and install remote Segment Identifiers (SIDs) with wide (32-bit) functions
- supports both Layer and Layer service steering through these remote SIDs, and
- facilitates precise packet forwarding and service function chaining by embedding service SIDs into packet headers to steer traffic along specific paths including the intended service functions

This capability is enabled by default and allows the source node to insert a service SID into the packet header, which uniquely identifies a specific service function and directs the packet through the network accordingly.

Table 31: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: L2 and L3 Services with Remote SIDs from Wide Local ID Block	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SRv6 Services: L2 and L3 Services with Remote SIDs from Wide Local ID Block	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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.

How the L2 and L3 services using remote SIDs from W-LIB work

Summary

The SRv6 headend node, source node, and receiver are the key components involved in SRv6 services with remote SIDs from W-LIB. Together, these components manage the identification, insertion, signaling (through transposition and SRv6 SID Structure Sub-Sub-TLV), and reassembly of SRv6 Service SIDs. This collaborative effort ensures precise Layer 2 and Layer 3 traffic steering for service function chaining and optimizes the compression of service prefix NLRIs in BGP update messages.

Workflow

These stages describe how the L2 and L3 services using remote SIDs from W-LIB work.

1. The source node inserts an SRv6 Service SID into the packet header to identify and steer the packet through the intended service function path.

2. The headend node signals the Service SID by transposing variable parts of the SRv6 SID value (function, argument, or both) into existing label fields, which optimizes compression of service prefix NLRIs in BGP updates.
3. The SRv6 SID Structure Sub-Sub-TLV (SSTLV) carries appropriate length fields that enable the receiver to accurately reassemble the split SRv6 Service SID parts. The Transposition Offset specifies the bit position, and the Transposition Length specifies how many bits are extracted from the SRv6 SID and placed into the high-order bits of the label field.
4. The receiver reassembles the split SRv6 Service SID parts based on the SSTLV information.
5. The headend node steers the packet along the specific path that includes the required service function, using the reassembled Service SID.

Remote W-LIB uSID structure with SRv6 SID SSTLV

A remote W-LIB uSID is defined by specific parameters within the SRv6 SID SSTLV that describe the composition and transposition of the SID value and label.

The example uSID fcbb:bb00:0200:fff0:0001:: is characterized 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:

From these parameters:

- The SID value is constructed as fcbb:bb00:0200:fff0::, which includes the block, node, and function parts without the transposed argument.
- The Label value extracted through transposition is 0x0001, representing the transposed bits from the original SID.

Interpret BGP VPNv4 route table output for a prefix learned from multiple egress PEs

Understand the details of BGP paths for a VPNv4 prefix learned from three egress Provider Edge (PE) routers

This task helps network engineers analyze BGP route entries, including path attributes and Segment Routing (SR) information, to troubleshoot or verify VPNv4 routing

Procedure

View the 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

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

Example:

```
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 0x40020
      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
```

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

L3 services with local SIDs from W-LIB

L3 services with local SIDs from W-LIB are network services that

- use SRv6 service SIDs to identify specific service functions
- steer packets along defined paths including those service functions, and
- enable flexible and efficient service delivery by allocating uSIDs from the W-LIB space.

Table 32: Feature History Table

Feature Name	Release	Description
SRv6-Services: L3 Services with Local SIDs from W-LIB	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SRv6-Services: L3 Services with Local SIDs from W-LIB	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

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.

SRv6 service SID and BGP uSID allocation

An SRv6 service SID uniquely identifies a specific service function within a network. The source node inserts this service SID into the packet header to steer the packet along a defined path that includes the targeted service function. This mechanism provides enhanced flexibility and control over packet processing, enabling efficient service delivery across the network.

Key points regarding uSID allocation and BGP transposition include:

- By default, BGP instructs the SID-Manager to allocate uSIDs from the LIB space only. When enabled, BGP can enforce uSID allocation from the W-LIB space.
- For VPN services, BGP performs transposition of the service SID into the label part of the NLRI as specified in IETF RFC 9252. In the LIB implementation, the 16-bit function is transposed to the NLRI label field.
- In the W-LIB implementation, BGP transposes the last 16 bits of the 32-bit W-LIB function to the NLRI label field for VPNv4 and VPNv6 routes.
- There is no transposition for BGPv4/BGPv6 global routes.

Support for Cisco Silicon One ASICs

- 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.

Configure L3 services with local SIDs from W-LIB

Enable the allocation and advertisement of an SRv6 service SID using the wide-local-id-block (W-LIB) mode to support L3 services

The W-LIB allocation mode applies precedence rules at various configuration levels within BGP to control how uSIDs are allocated and advertised.

Procedure

Step 1 Apply W-LIB uSID allocation globally under BGP

Example:

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

Step 2 Apply W-LIB uSID allocation at the IPv4 and IPv6 address family levels under BGP.

Example:

```
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
```

Step 3 Apply W-LIB uSID allocation for all VPNv4 and VPNv6 address families.

Example:

```
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
```

Step 4 Apply W-LIB uSID allocation at the VRF level for IPv4 and IPv6 address families.

Example:

```
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
```

Step 5 Use the show commands to verify the W-LIB uSID allocation.

a) Verify the W-LIB uSID allocation.

Example:

```
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
```

b) Verify the advertised SRv6 W-LIB uSID for the default VRF.

Example:

```
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
```

c) Verify the advertised SRv6 W-LIB uSID for a specific VRF (foo).

Example:

```
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
```

Static SRv6 pseudowire

Static SRv6 pseudowire (PW) gives the flexibility to

- configure single-homing static SRv6 pseudowire between two Provider Edge (PE) routers, in an SRv6 core plane, and
- extends Virtual Private Wire Service (VPWS) capabilities by incorporating SRv6.

Table 33: Feature History Table

Feature Name	Release Information	Feature Description
Static SRv6 pseudowire	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100, Q200] (select variants only*); Modular Systems (8800 [LC ASIC: P100]); Centralized Systems (8600 [ASIC: Q200]) You can now create a virtual connection between two endpoints over an IPv6 network using SRv6. Unlike dynamic pseudowires, which rely on signaling protocols to establish and maintain connections, static pseudowires are manually configured and do not require a control plane for setup. This enhancement allows operators to extend the existing Virtual Private Wire Service (VPWS) capabilities by incorporating SRv6, providing improved flexibility and scalability for service providers. * This feature is supported on: • 8201-32FH

Key aspects of SRv6 static pseudowire

This feature enhances the existing Virtual Private Wire Service (VPWS) by incorporating SRv6 capabilities, which previously supported only MPLS.

This enhancement enables the configuration and management of static SRv6 pseudowires, thereby providing enhanced flexibility and scalability for service providers. Additionally, it addresses the specific requirements of customers who require static configuration for their network services.

- Static configuration: Unlike dynamic pseudowires that rely on signaling protocols like LDP or BGP, static pseudowires are manually configured. This allows for more control and customization in specific network scenarios.
- Pseudowire (PW): A pseudowire is a mechanism that emulates the properties of a traditional telecommunications circuit over a packet-switched network. It allows for the transport of Layer 2 frames over an IP/MPLS network.

The benefits of having an option to configure pseudowire in a static SRv6 core plane gives businesses an edge over their competitors, as listed.

- Enhanced flexibility: By using SRv6, service providers can define more granular and flexible paths for their pseudowires, improving network efficiency and performance.
- Scalability: SRv6 allows for a scalable solution that can handle a large number of pseudowires without the need for complex BGP signaling protocols.
- Interoperability: The feature is designed to work across various platforms and planes making it versatile for different network environments.

Configure Static SRv6 pseudo-wire

The purpose of this task is to configure static SRv6 pseudo-wires. To configure this, the network operators must define the local and remote SRv6 SIDs for the pseudo-wire endpoints. This involves specifying the segment routing configuration and ensuring that the pseudo-wire is correctly mapped to the desired network path.

Procedure

Step 1 Define the locator.

Example:

```
Router(config)#segment-routing
Router(config-sr)#srv6
Router(config-srv6)#formats
Router(config-srv6-fmts)#format usid-f3216
Router(config-srv6-fmts)#usid local-id-block explicit start 0xee00 (default start is 0xfe00 - optional
 required if you if more than 256 )
Router(config-srv6-fmts)#exit

Router(config-srv6)#encapsulation
Router(config-srv6)#source-address 1::1
Router(config-srv6)traffic-class propagate

Router(config-srv6)#locators
Router(config-srv6-locators)#locator locator0
Router(config-srv6-locators)#micro-segment behavior unode psp-usd
Router(config-srv6-locators)#prefix fccc:cc00:1::/48
```

Step 2 Define the Sub-interface.

Example:

```
interface Bundle-Ether101.40010001 l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
!
```

Step 3 Define the xconnect group.

Example:

```
Router(config)#l2vpn
Router(config-l2vpn)#xconnect group xg4001
Router(config-l2vpn-xc)#p2p vpws-400100150
Router(config-l2vpn-xc-p2p)#interface Bundle-Ether101
Router(config-l2vpn-xc-p2p)#neighbor segment-routing srv6 static local fccc:cc00:1:ee96:: remote
fccc:cc00:2:ee96::
```

Note
The **remote fccc:cc00:2:ee96::** SID is the remote device SID. You must configure this on both ends of the PE. However the local and remote SIDs interrchange

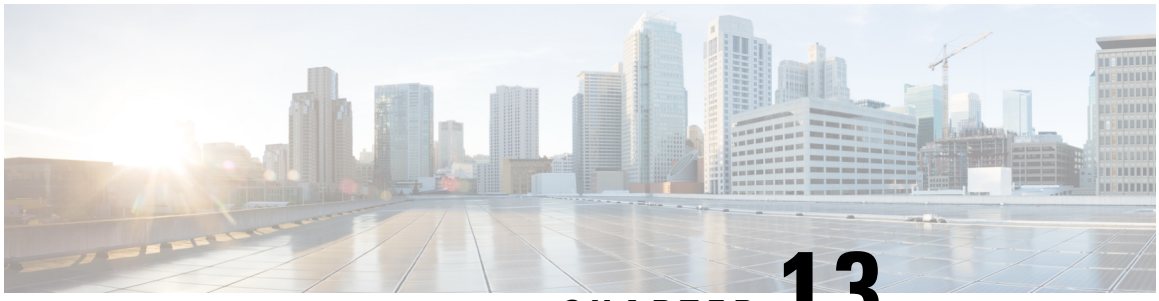
Step 4 Run the following show commands to view the defined locator, evpn, L2vpn, xconnect and evpn internal ID details.

Example:

```
Router#show segment-routing srv6 locator locator0 sid  fccc:cc00:1:ee01:: detail

SID                         Behavior          Context                          Owner
 State  RW
--------------------------  ----------------  -------------------------------  ------------------
 -----  --
fccc:cc00:1:ee01::          uDX2              pw_id:3221225475                 l2vpn_srv6
 InUse  Y
  SID Function: 0xee01
  SID context: { static pw_id=3221225475 }
  Locator: 'locator0'
  Allocation type: Explicit
  Created: Dec 16 15:38:33.852 (00:05:19 ago)


Router# show evpn segment-routing srv6 detail

Configured default locator: None
Configured default SID Function Length: 16 bits
EVIs with unknown locator config: 0
VPWS with unknown locator config: 0
Global SID Function Length: 16 bits
No SRv6 locators in use

Router#show l2vpn xconnect group xg4001 xc-name vpws-40010001 det

Group xg4001, XC vpws-40010001, state is up; Interworking none
Decoupled mode: Disabled
  AC: Bundle-Ether101.40010001, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 9186; XC ID 0xc0000003; interworking none
    Statistics:
      packets: received 7109476, sent 5069183
      bytes: received 910012144, sent 648854934
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor ::ffff:10.0.0.4, PW ID 2684354565, state is up
    PW class not set, XC ID 0xc0000003
    Encapsulation SRv6, protocol none
    PW type Ethernet, control word unknown, interworking none
    PW backup disable delay 0 sec

    Ignore MTU mismatch: Disabled
    Transmit MTU zero: Disabled
    Reachability: Up
    Nexthop type: Internal ID ::ffff:10.0.0.4

      SRv6              Local                       Remote
      ----------------  --------------------------- --------------------------
      uDX2              fccc:cc00:1:ee01::          fccc:cc00:2:ee01::
      AC ID             0                           0
      MTU               9200                        9200
      Locator           locator0                    N/A
      Locator Resolved  Yes                         N/A
      SRv6 Headend      H.Encaps.L2.Red             N/A
    Statistics:
      packets: received 5069183, sent 7109476
      bytes: received 648854934, sent 910012144


Router#show evpn internal-id
```

```
VPN-ID     Encap  Ethernet Segment Id         EtherTag     Internal ID
---------- ------ -------------------------- ----------   -----------
0          SRv6   fccc:cc00:2:ee01::         13421820     ::ffff:10.0.0.4
   Summary pathlist (ID 0x000000000000001a):
 0x05000005     fccc:cc00:2:ee01::                        fccc:cc00:2:ee01::

0          SRv6   fccc:cc00:2:ee02::         13421820     ::ffff:10.0.0.2
   Summary pathlist (ID 0x000000000000000a):
 0x05000002     fccc:cc00:2:ee02::                        fccc:cc00:2:ee02::
```

```
Router#show evpn internal-id  detail

VPN-ID     Encap  Ethernet Segment Id         EtherTag     Internal ID
---------- ------ -------------------------- ----------   -----------
0          SRv6   fccc:cc00:2:ee01::         13421820     ::ffff:10.0.0.4
   Path resolved: TRUE
   Path Internal ID: ::ffff:10.0.0.4
    IP-Tunnel (ID:0x0000000000000019)
                fccc:cc00:2:ee01::                        fccc:cc00:2:ee01::
                  Path Version:1, Originating PE:::
   Summary pathlist (ID 0x000000000000001a):
 0x05000005     fccc:cc00:2:ee01::                        fccc:cc00:2:ee01::

0          SRv6   fccc:cc00:2:ee02::         13421820     ::ffff:10.0.0.2
   Path resolved: TRUE
   Path Internal ID: ::ffff:10.0.0.2
    IP-Tunnel (ID:0x0000000000000009)
                fccc:cc00:2:ee02::                        fccc:cc00:2:ee02::
                  Path Version:1, Originating PE:::
   Summary pathlist (ID 0x000000000000000a):
 0x05000002     fccc:cc00:2:ee02::                        fccc:cc00:2:ee02::
```



CHAPTER 13

SRv6-MPLS L3 Service Interworking Gateways

This chapter focuses on the crucial role of SRv6-MPLS L3 service interworking gateways in bridging disparate network domains, ensuring seamless service continuity between SRv6 and MPLS infrastructures. It details how these gateways facilitate the extension of L3 EVPN services, supporting migration strategies and enabling communication across domains with potentially different Segment Identifier (SID) formats. The chapter covers the operational mechanisms, benefits, and configuration of interworking gateways, including scenarios for dual-connected Provider Edges (PEs) and route re-origination for optimized scalability and interoperability.

SRv6 MPLS L3 service interworking gateway

An SRv6 MPLS L3 service interworking gateway is a gateway that

- extends L3 services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane
- enables SRv6 L3VPN domains to interwork with existing MPLS L3VPN domains and supports migration from MPLS L3VPN to SRv6 L3VPN, and
- provides both transport and service termination at the gateway node.

Table 34: Feature History Table

Feature Name	Release	Description
Identical Route Distinguisher (RD) for Interworking Gateways between MPLS and SRv6 Domains	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release	Description
Identical Route Distinguisher (RD) for Interworking Gateways between MPLS and SRv6 Domains	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8700 [ASIC: K100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SRv6/MPLS L3 Service Interworking Gateway (SRv6 Micro-SID)	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release	Description
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.

Gateway mechanism

The gateway performs these key actions to enable interworking:

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

Virtual Routing and Forwarding (VRFs) on the gateway node are configured with two sets of route targets (RTs): MPLS L3VPN RTs and SRv6 L3VPN RTs (referred to as stitching RTs).

To understand the detailed control-plane and data-plane behaviors for specific traffic flows, refer to these process topics:

- How SRv6-to-MPLS Control-Plane and MPLS-to-SRv6 Data-Plane traffic works
- How MPLS-to-SRv6 control-plane and SRv6-to-MPLS data-plane traffic works, on page 199

How MPLS-to-SRv6 control-plane and SRv6-to-MPLS data-plane traffic works

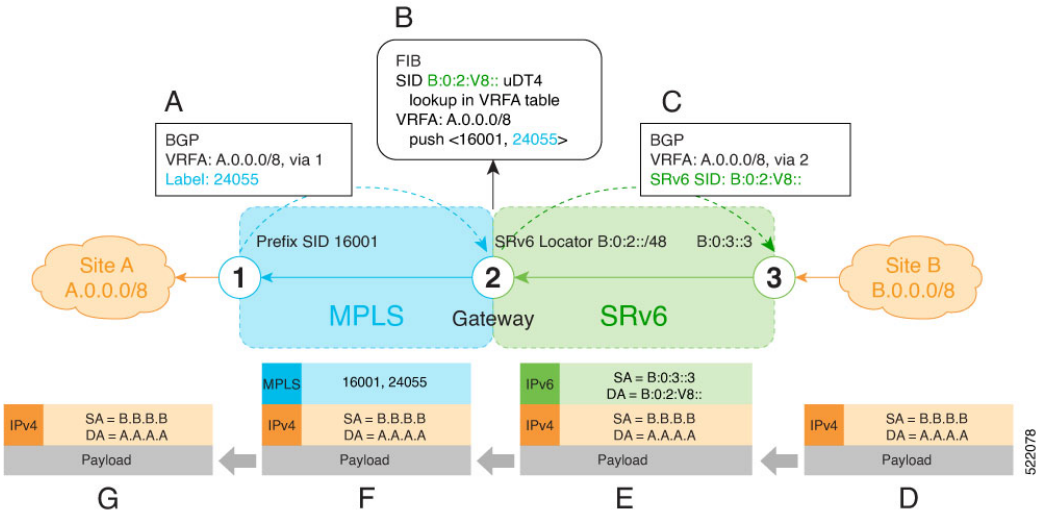
This process describes the control-plane behaviors in the MPLS-to-SRv6 direction for traffic that flows in the SRv6-to-MPLS data-plane direction.

Summary

The process involves BGP L3VPN updates exchanged between Node 1 (MPLS domain) and Node 2 (gateway), which then re-originates updates into the SRv6 domain. Subsequently, traffic from Site B is encapsulated by Node 3, processed by Node 2, and finally forwarded by Node 1 to Site A.

Workflow

Figure 19: Topology for MPLS-to-SRv6 control-plane and SRv6-to-MPLS data-plane traffic flow



These stages describe how the SRv6/MPLS L3 service interworking gateway works, covering both control plane and data plane flows.

1. (A) As illustrated by A in the figure, 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. Refer the part A in the figure.
2. (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.

3. (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.
4. (D) Site B sends traffic to an IPv4 prefix (A.A.A.A) of Site A.
5. (E) Node 3 Encapsulates the payload in an outer IPv6 header with destination address (DA) equal to the uDT4 function (B:0:2:V8::).
6. (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)

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

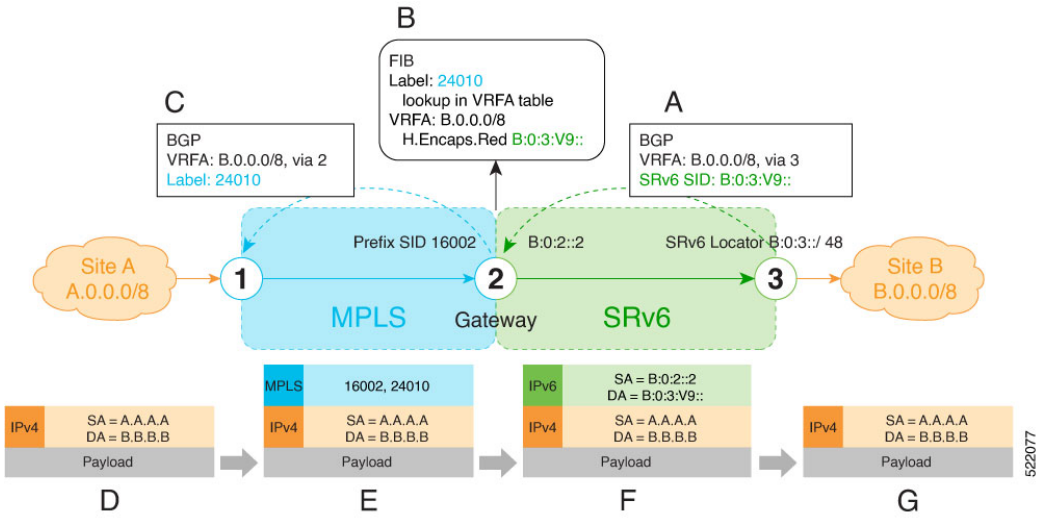
How SRv6-to-MPLS Control-Plane and MPLS-to-SRv6 Data-Plane Traffic Works

This process describes the flow of control plane information from an SRv6 domain to an MPLS domain, and the subsequent forwarding of data traffic from the MPLS domain to the SRv6 domain through an SRv6/MPLS L3 Service Interworking Gateway. This interworking ensures seamless service continuity and efficient traffic routing across different network segments.

Summary

In this process, an L3VPN PE in the SRv6 domain (Node 3) advertises a prefix. The SRv6/MPLS L3 Service Interworking Gateway (Node 2) imports this advertisement, programs its FIB, and re-originates the prefix into the MPLS domain. Subsequently, traffic originating from the MPLS domain (Site A via Node 1) is encapsulated by Node 1, processed and re-encapsulated by the gateway (Node 2) for the SRv6 domain, and finally forwarded by Node 3 to its destination (Site B

Workflow



These stages describe how the SRv6-to-MPLS control plane and MPLS-to-SRv6 data plane interworking process functions:

1. (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, as shown in A of Figure 1.

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.

2. (B) Node 2 (gateway) imports the BGP L3VPN update and programs its FIB, as shown in B of Figure 1.
   - 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.

3. (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, as shown in C of Figure 1.
4. D: Site A sends traffic to an IPv4 prefix (B.B.B.B) of Site B, as shown in D of Figure 1.
5. (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), as shown in E of Figure 1.
6. (F) Node 2 performs the following actions, as shown in F of Figure 1.
   - 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::)

Interworking gateways for L3 EVPN SRv6 and L3VPN MPLS

L3 EVPN Interworking between SRv6 and MPLS is a SRv6 networking solution that

- extends L3 EVPN services between SRv6 and MPLS domains
- provides service continuity on both the control plane and data plane, and
- allows for migration from MPLS L3 EVPN to SRv6 L3 EVPN.

Restrictions of interworking gateways for L3 EVPN SRv6 and L3VPN MPLS

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

How interworking gateways for L3 EVPN SRv6 and L3VPN MPLS operate

The L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway acts as a central network device. It provides both transport and service termination at the gateway node, enabling the extension of L3 EVPN services between MPLS and SRv6 domains.

For this interworking, VRFs on the gateway node are configured with two sets of route targets (RTs):

- L3 EVPN/MPLS RTs
- L3 EVPN/SRv6 RTs (also known as stitching RTs)

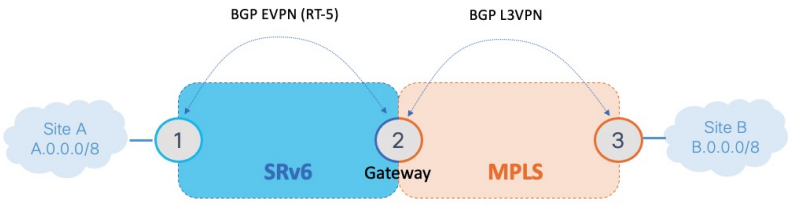
Summary

The key components involved in the interworking process are:

- Gateway node: The central device that performs the interworking functions, connecting and translating between the SRv6 and MPLS domains.
- SRv6 L3 EVPN domains: Network segments utilizing SRv6 for transport and EVPN for L3 service delivery.
- MPLS L3VPN domains: Traditional MPLS networks providing Layer 3 VPN services.
- Virtual Routing and Forwarding instances: Logical routing tables configured on the gateway node to isolate and manage different VPN services.
- Route targets (RTs): BGP extended communities used to control the import and export of VPN routes between domains, including stitching RTs for interworking. Stitching is the overarching process by which the gateway node integrates the control and data planes of the two distinct domains, often involving specific route targets and re-origination mechanisms.
- Border Gateway Protocol) The routing protocol used to exchange VPN routing information between the gateway and the respective domains.

The L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway facilitates seamless service continuity between SRv6 and MPLS domains. It achieves this by importing and re-originating service routes in the control plane and by performing necessary encapsulation, decapsulation, and lookup operations in the data plane. This process supports traffic flow in both directions and enables migration scenarios.

Workflow



These stages describe how interworking gateways for L3 EVPN SRv6 and L3VPN MPLS operate:

1. The gateway prepares for route exchange.
   - 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)
2. The gateway performs control plane interworking.

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

3. The gateway performs data plane interworking.
   - MPLS-to-SRv6 Traffic Forwarding:
     - The gateway pops the MPLS L3 EVPN label.
     - The gateway looks up the destination prefix.
     - The gateway pushes the appropriate SRv6 encapsulation (uDT4/H.Encaps.Red).
     - The gateway forwards traffic to the SRv6 domain.
   - SRv6-to-MPLS Traffic Forwarding:
     - The gateway removes the outer IPv6 header.
     - The gateway looks up the destination prefix.
     - The gateway pushes the L3 EVPN and next-hop MPLS labels.
     - The gateway forwards traffic to the MPLS domain.

Configure interworking gateways for L3 EVPN SRv6 and L3VPN MPLS

Use this procedure to enable seamless Layer 3 service interworking between an SRv6 L3 EVPN domain and an MPLS L3VPN domain on a gateway node.

This task configures a gateway node to act as an interworking point, allowing SRv6 L3 EVPN domains to exchange Layer 3 services with existing MPLS L3VPN domains. The configuration involves setting up SRv6 parameters, defining VRF route targets for stitching, and configuring BGP peering for route exchange and re-origination.

Procedure

Step 1 Enable SRv6 with locator and encapsulation parameters.

Example:

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

```
!
```

Step 2 Configure the VPNv4/VPNv6 VRF with route targets.

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

Example:

```
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
  !
 !
!
```

Step 3 Configure BGP for SRv6 and MPLS interworking.

Example:

```
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
 !
!
!
```

Layer 3 service gateway for interconnecting SRv6 domains

A Layer 3 service gateway is a mechanism that

- extends L3 services between two distinct SRv6 domains
- enables seamless service continuity on both control and data planes for SRv6-based networks, and
- facilitates route re-origination and summarization between SRv6 VPNv4 and VPNv6 address families.

Table 35: Feature History Table

Feature Name	Release	Description
Layer 3 service gateway for interconnecting SRv6 domains	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q200, P100]) Optimize network scalability and interoperability by reducing SID resource usage, and enabling seamless integration between distinct SRv6 domains. The Layer 3 service gateway provides a flexible mechanism to extend Layer 3 services across different SRv6 networks, supporting efficient route summarization, cross-locator compatibility, and consistent service continuity on both control and data planes.

Key concepts

- Route re-origination is the process by which a network device, such as a Layer 3 service gateway, receives routes from one network domain and then advertises those routes as if they originated from itself in another domain. The received routes from one address family are imported and updated with a next-hop set to self and a locally generated SID. These routes are then re-advertised either within the same address family or into another compatible address family, appearing as newly originated routes from the gateway. This process is essential for stitching or interworking between different VPN address families, ensuring consistent route distribution and reachability across SRv6 domains.
- Route summarization is the process of consolidating multiple specific routes into a single, more general route before advertising it into another domain. For example, several contiguous prefixes can be aggregated into a broader prefix, reducing the number of routes and SIDs that must be managed and propagated. This optimization applies to both VPNv4 and VPNv6 address families.

Key challenges

Managing large-scale networks while ensuring interoperability between different SRv6 domains involves two major challenges:

- Encap ID consumption optimization: Large SRv6 networks generate numerous unique Segment Identifiers (SIDs), which can exhaust Encap ID resources on remote Provider Edge (PE) devices. This limitation impacts network scalability and efficiency.
- Interconnecting SRv6 domains: Different SRv6 domains may use varying locator formats, such as base format and uSID-based locators, complicating seamless inter-domain connectivity and service continuity.
- Cross-locator format interoperability: Supports both same-format (base-to-base or uSID-to-uSID) and different-format (base-to-uSID or uSID-to-base) domain interconnections, providing flexible integration for heterogeneous SRv6 deployments.

The Layer 3 service gateway addresses key challenges in large-scale SRv6 deployments by optimizing Encap ID consumption through network partitioning and SID summarization, and by interconnecting SRv6 domains that use different locator formats (base format and uSID-based) through Route Policy Language (RPL) based SID allocation.

Feature benefits

The SRv6 Layer 3 service gateway addresses these challenges by:

- Network partitioning and SID summarization: Partitions the network into core and regional domains, enabling SID summarization and route re-origination at the gateway. This reduces the number of unique SIDs propagated into the core, optimizing Encap ID resource usage and simplifying network scaling
- Flexible interconnection across domains: Acts as an effective intermediary to extend layer 3 VPN services across distinct SRv6 domains, ensuring seamless service delivery and integration.
- Interconnecting different locator formats: Supports both base format and uSID locators, enabling interoperability between heterogeneous SRv6 deployments. Route Policy Language (RPL)-based SID allocation ensures that SIDs are allocated from the appropriate locator format for each prefix.
- Robust control and data Plane operations: Facilitates route re-origination, summarization, importing, and re-advertising of service routes for both VPNv4 and VPNv6 address families, improving route management and efficiency. Also supports critical data plane functions such as packet pop/decapsulation, IP lookup, and encapsulation/push operations for smooth packet forwarding.

- Service continuity for IPv4 and IPv6 VPNs: Manages routing information and maintains seamless service continuity across both control and data planes for SRv6-based networks, supporting both IPv4 and IPv6 VPNs.

How SRv6 L3 service gateway works

The SRv6 L3 service gateway provides a robust solution for interconnecting disparate SRv6 domains. It enables efficient network partitioning and ensures seamless data plane connectivity and interoperability between domains that use different SRv6 formats. The process covers both scenarios where the source and destination SRv6 domains use the same SID format and where they use different SID formats.

Summary

The key components involved in the process are:

- SRv6 L3 service gateway: The central network device that acts as a bridge, performing decapsulation, IP lookup, and re-encapsulation to forward traffic between SRv6 domains.
- SRv6 domains: Distinct network segments that utilize SRv6 technology and are interconnected by the gateway.
- Traffic (data plane): The IP packets and their associated SRv6 headers that flow through the gateway, undergoing transformation.
- Regional domains: Specific SRv6 domains representing partitioned parts of a larger L3VPN network, connected to a core domain through the gateway.
- Core domain: The central SRv6 domain in a partitioned L3VPN network, connected to regional domains through the gateway.
- Base format SRv6 locators: A specific format of SRv6 locators used within certain SRv6 domains.
- uSID-based SRv6 locators: Another specific format of SRv6 locators used within different SRv6 domains.
- Route Policy Language (RPL): RPL is a mechanism used by the gateway to determine which locator format (base or uSID) is assigned to each prefix. RPL selects the appropriate SID format for each prefix, but only one SID can be allocated per prefix at the gateway.

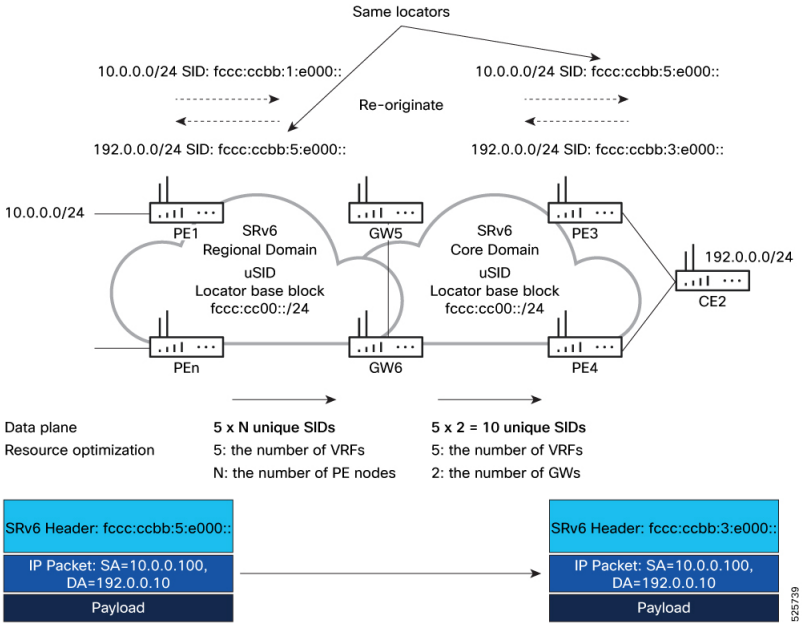
The SRv6 L3 service gateway, acting as a central network device, interconnects disparate SRv6 domains. It achieves this by performing control plane SID re-origination, summarization, and format translation, alongside data plane packet decapsulation, IP lookup, and re-encapsulation. This comprehensive process ensures efficient network partitioning, seamless data plane connectivity, and interoperability across domains with varying SRv6 SID formats.

Workflow

These stages describe SRv6 layer 3 service gateway interconnect domains:

1. Control plane stage: Re-origination and summarization of SIDs (Use case 1 - Regional to Core):
    - Regional to core: When VPN prefixes flow from a regional domain to the core, the gateway re-originates them using local per-VRF SIDs.

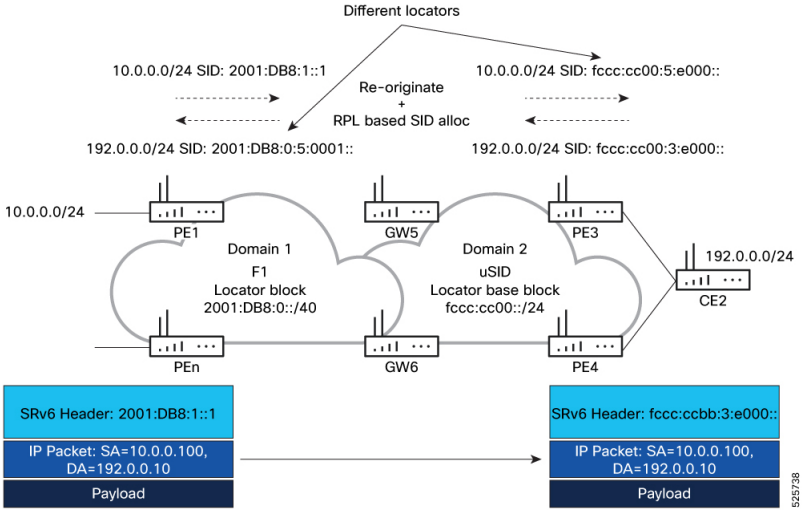
- Core to regional: When VPN prefixes flow from the core to a regional domain, the gateway processes and forwards it into the appropriate regional domain, re-originating SIDs as needed for that regional domain.



2. Control plane stage: Format translation between domains (Use case 2):

   When prefix crosses between domains using different SRv6 formats (for example, base format to uSID), the gateway utilizes RPL-based SID allocation to match prefixes from the source domain and assign a SID from the corresponding locator type of the destination domain.

   - When prefix flows from Domain 1 to 2, the gateway assigns a uSID corresponding to the prefix and advertises it into Domain 2.
   - When prefix flows from Domain 2 to 1, the gateway assigns a Base or F1 format SID for the prefix and advertises it into Domain 1.



3. Data plane stage: Packet processing and forwarding.
    - Traffic arrival: Traffic originating from one SRv6 domain arrives at the SRv6 L3 service gateway.
    - Pop or decapsulation: The gateway decapsulates the incoming SRv6 packet by removing its existing SRv6 header.
    - IP lookup: The gateway performs an IP lookup on the exposed inner IP packet to determine its next hop or final destination within the target SRv6 domain.
    - Encapsulation or Push: Based on the IP lookup, the gateway encapsulates the IP packet with a new SRv6 header that is appropriate for the destination SRv6 domain.
        - Same SID format: The new SRv6 header uses the same SID format as the source domain, maintaining SID consistency across the gateway.
        - Different SID formats: The gateway translates the SID format as needed, for example, from base SID to uSID or vice versa, to match the destination domain's requirements.
    - Traffic forwarding: The newly encapsulated traffic is then pushed out towards the target SRv6 domain.

Configure SRv6 layer 3 service gateway with different SID formats

Use this procedure when the core and regional domains use different SID formats, for example, base in the core and uSID in the regional domain.

Before you begin

Ensure that these tasks are completed:

- Explicitly configure the per-VRF allocation mode.

Use this procedure when the core and regional domains use different SID formats, for example, base in the core and uSID in the regional domain.

Procedure

Step 1 Configure two locators, one using the base format and another using the uSID format.

Example:

```
Router(config)#segment-routing
Router(config-sr)#srv6
Router(config-srv6)#locators
Router(config-srv6-locators)#locator base_locator
Router(config-srv6-locator)#prefix 2001:DB8:0:5::/64
Router(config-srv6-locator)#exit
Router(config-srv6-locators)#locator usid_locator
Router(config-srv6-locator)#prefix fccc:cc00:5::/48
Router(config-srv6-locator)#exit
```

Step 2 Set up a route policy to allocate uSIDs for prefixes received from the base domain and allocate base format SIDs for prefixes received from the uSID domain.

Example:

```
Router(config-bgp-vrf-af)#route-policy alloc_sid_policy
Router(config-rpl)#if destination in core_prefix_set then
Router(config-rpl-if)#set srv6-alloc-mode per-vrf locator base_locator
Router(config-rpl-if)#else
Router(config-rpl-else)#set srv6-alloc-mode per-vrf locator usid_locator
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy

Router(config)#prefix-set core_prefix_set
Router(config-pfx)#192.0.0.0/24
Router(config-pfx)#end-set
```

Step 3 Configure the VRF to import and export route targets for both the core and regional domains.

Example:

```
Router(config)#vrf VRF1
Router(config-vrf)#address-family ipv4 unicast
Router(config-vrf-af)#import route-target
Router(config-vrf-import-rt)#100:1
Router(config-vrf-import-rt)#200:1 stitching
Router(config-vrf-import-rt)#exit
Router(config-vrf-af)#export route-target
Router(config-vrf-export-rt)#100:1
Router(config-vrf-export-rt)#200:1 stitching
Router(config-vrf-export-rt)#exit
Router(config-vrf-af)#exit

Router(config-vrf)#address-family ipv6 unicast
Router(config-vrf-af)#import route-target
Router(config-vrf-import-rt)#100:1
Router(config-vrf-import-rt)#200:1 stitching
Router(config-vrf-import-rt)#exit
Router(config-vrf-af)#export route-target
Router(config-vrf-export-rt)# 100:1
Router(config-vrf-export-rt)#200:1 stitching
Router(config-vrf-export-rt)#exit
Router(config-vrf-af)#exit
Router(config-vrf)#exit
```

Step 4 Configure BGP neighbor for the regional and core domains.

Example:

```
Router(config)#router bgp 65001
Router(config-bgp)# bgp router-id 10.5.5.5
Router(config-bgp)#address-family vpnv4 unicast
Router(config-bgp-af)#segment-routing srv6
Router(config-bgp-af-srv6)#locator locator0
Router(config-bgp-af-srv6)#alloc mode route-policy alloc-sid-policy-ipv4
Router(config-bgp-af-srv6)#exit
Router(config-bgp-af)#exit

Router(config-bgp)#address-family vpnv6 unicast
Router(config-bgp-af)#vrf all
Router(config-bgp-af-vrfall)#segment-routing srv6
Router(config-bgp-af-vrfall-srv6)#locator locator0
Router(config-bgp-af-vrfall-srv6)#alloc mode per-vrf
Router(config-bgp-af-vrfall-srv6)#exit
Router(config-bgp-af-vrfall)#exit
Router(config-bgp-af)#exit

Router(config-bgp)#neighbor-group CORE-DOMAIN
Router(config-bgp-nbrgrp)#remote-as 65001
Router(config-bgp-nbrgrp)#update-source Loopback0
```

```
Router(config-bgp-nbrgrp)#address-family vpnv4 unicast
Router(config-bgp-nbrgrp-af)#import reoriginate stitching-rt
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv4 unicast re-originated
Router(config-bgp-nbrgrp-af)#exit

Router(config-bgp-nbrgrp)#address-family vpnv6 unicast
Router(config-bgp-nbrgrp-af)# import reoriginate stitching-rt
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv6 unicast re-originated
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#exit

Router(config-bgp)#neighbor-group REGIONAL-DOMAIN
Router(config-bgp-nbrgrp)#remote-as 65001
Router(config-bgp-nbrgrp)#update-source Loopback0

Router(config-bgp-nbrgrp)#address-family vpnv4 unicast
Router(config-bgp-nbrgrp-af)#import stitching-rt reoriginate
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv4 unicast re-originated stitching-rt
Router(config-bgp-nbrgrp-af)#exit

Router(config-bgp-nbrgrp)#address-family vpnv6 unicast
Router(config-bgp-nbrgrp-af)#import stitching-rt reoriginate
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv6 unicast re-originated stitching-rt
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#exit

Router(config-bgp)#neighbor 2001:DB8:0:1::1
Router(config-bgp-nbr)#use neighbor-group REGINAL-DOMAIN
Router(config-bgp-nbr)#exit
Router(config-bgp)#neighbor fccc:cc00:3::1
Router(config-bgp-nbr)#use neighbor-group CORE-DOMAIN
Router(config-bgp-nbr)#exit

Router(config-bgp)#vrf VRF1
Router(config-bgp-vrf)#rd auto
Router(config-bgp-vrf)#address-family ipv4 unicast
Router(config-bgp-vrf-af)#segment-routing srv6
Router(config-bgp-vrf-af-srv6)#locator locator-usid
Router(config-bgp-vrf-af-srv6)#alloc mode route-policy alloc-sid-policy-ipv4
Router(config-bgp-vrf-af-srv6)#exit
```

Configure SRv6 layer 3 service gateway with same SID formats

Use this procedure when both the core and regional domains use the same SID format, for example, both base or both uSID.

Before you begin

Ensure that these tasks are completed:

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

- Configure SRv6 under IS-IS
- Configure per-VRF allocation mode

Follow these steps to configure the SRv6 L3 service gateways for seamless interconnection of domains with the same SID format.

Procedure

Step 1 Configure the VRF to import and export route targets for both the core and regional domains.

Example:

```
Router(config)#vrf VRF1
Router(config-vrf)#address-family ipv4 unicast
Router(config-vrf-af)#import route-target
Router(config-vrf-import-rt)#100:1
Router(config-vrf-import-rt)#200:1 stitching
Router(config-vrf-import-rt)#exit
Router(config-vrf-af)#export route-target
Router(config-vrf-export-rt)#100:1
Router(config-vrf-export-rt)#200:1 stitching
Router(config-vrf-export-rt)#exit
Router(config-vrf-af)#exit

Router(config-vrf)#address-family ipv6 unicast
Router(config-vrf-af)#import route-target
Router(config-vrf-import-rt)#100:1
Router(config-vrf-import-rt)#200:1 stitching
Router(config-vrf-import-rt)#exit
Router(config-vrf-af)#export route-target
Router(config-vrf-export-rt)# 100:1
Router(config-vrf-export-rt)#200:1 stitching
Router(config-vrf-export-rt)#exit
Router(config-vrf-af)#exit
Router(config-vrf)#exit
```

Step 2 Configure BGP neighbor for the regional and core domains.

Example:

This example shows how to uSID to uSID VPNv4/v6 gateway:

```
Router(config)#router bgp 65001
Router(config-bgp)# bgp router-id 10.5.5.5
Router(config-bgp)#address-family vpnv4 unicast
Router(config-bgp-af)# vrf all
Router(config-bgp-af-vrfall)#segment-routing srv6
Router(config-bgp-af-vrfall-srv6)#locator locator0
Router(config-bgp-af-vrfall-srv6)#alloc mode per-vrf
Router(config-bgp-af-vrfall-srv6)#exit
Router(config-bgp-af-vrfall)#exit
Router(config-bgp-af)#exit

Router(config-bgp)#address-family vpnv6 unicast
Router(config-bgp-af)#vrf all
Router(config-bgp-af-vrfall)#segment-routing srv6
Router(config-bgp-af-vrfall-srv6)#locator locator0
Router(config-bgp-af-vrfall-srv6)#alloc mode per-vrf
Router(config-bgp-af-vrfall-srv6)#exit
Router(config-bgp-af-vrfall)#exit
```

```
Router(config-bgp-af)#exit

Router(config-bgp)#neighbor-group CORE-DOMAIN
Router(config-bgp-nbrgrp)# remote-as 65001
Router(config-bgp-nbrgrp)#update-source Loopback0
Router(config-bgp-nbrgrp)#address-family vpnv4 unicast
Router(config-bgp-nbrgrp-af)#import reoriginate stitching-rt
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv4 unicast re-originated
Router(config-bgp-nbrgrp-af)#exit

Router(config-bgp-nbrgrp)#address-family vpnv6 unicast
Router(config-bgp-nbrgrp-af)# import reoriginate stitching-rt
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv6 unicast re-originated
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#exit

Router(config-bgp)#neighbor-group REGIONAL-DOMAIN
Router(config-bgp-nbrgrp)#remote-as 65001
Router(config-bgp-nbrgrp)#update-source Loopback0

Router(config-bgp-nbrgrp)#address-family vpnv4 unicast
Router(config-bgp-nbrgrp-af)#import stitching-rt reoriginate
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv4 unicast re-originated stitching-rt
Router(config-bgp-nbrgrp-af)#exit

Router(config-bgp-nbrgrp)#address-family vpnv6 unicast
Router(config-bgp-nbrgrp-af)#import stitching-rt reoriginate
Router(config-bgp-nbrgrp-af)#route-reflector-client
Router(config-bgp-nbrgrp-af)#encapsulation-type srv6
Router(config-bgp-nbrgrp-af)#advertise vpnv6 unicast re-originated stitching-rt
Router(config-bgp-nbrgrp-af)#exit
Router(config-bgp-nbrgrp)#exit

Router(config-bgp)#neighbor fccc:cc00:1::1
Router(config-bgp-nbr)#use neighbor-group REGINAL-DOMAIN
Router(config-bgp-nbr)#exit
Router(config-bgp)#neighbor fccc:cc00:3::1
Router(config-bgp-nbr)#use neighbor-group CORE-DOMAIN
Router(config-bgp-nbr)#exit

Router(config-bgp)# vrf VRF1
Router(config-bgp-vrf)# rd auto
Router(config-bgp-vrf)#address-family ipv4 unicast
Router(config-bgp-vrf-af)#exit
Router(config-bgp-vrf)#address-family ipv6 unicast
Router(config-bgp-vrf-af)#exit
Router(config-bgp-vrf)#exit
Router(config-bgp)#exit
```

SRv6 MPLS Dual-Connected PEs

An SRv6 MPLS dual-connected PE (SRv6 Micro SID) is a network feature that allows a PE router to support IPv4 or IPv6 L3VPN services for a given VRF with both MPLS and SRv6, and establishes an MPLS and SRv6 L3VPN coexistence scenario.

Table 36: Feature History Table

Feature Name	Release	Description
SRv6/MPLS Dual-Connected PE (SRv6 Micro SID)	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.
SRv6/MPLS Dual-Connected PE (SRv6 Micro SID)	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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.

How an SRv6 MPLS dual-connected PE operates

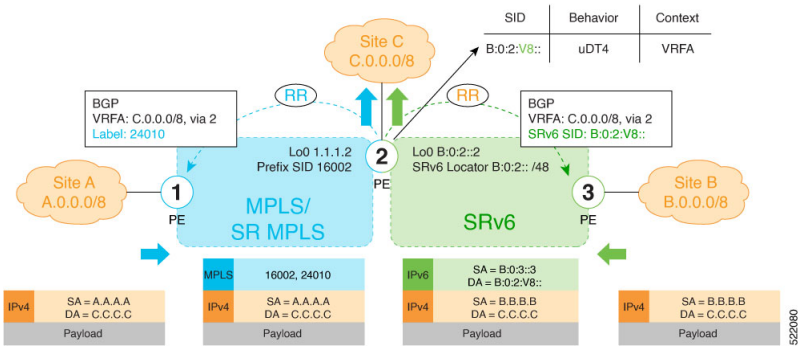
An SRv6 MPLS dual-connected PE (Provider Edge) is a router that supports IPv4 or IPv6 L3VPN services for a given VRF with both MPLS and SRv6. This setup creates an MPLS and SRv6 L3VPN coexistence scenario, sometimes referred to as a dual-connected PE.

Summary

The process involves a PE router configured to support L3VPN services for a VRF using both MPLS and SRv6. This enables the PE to provide connectivity for different sites, such as MPLS/IPv4 L3VPN between one set of sites and SRv6/IPv4 L3VPN between another set of sites, by managing the allocation and advertisement of appropriate labels and SIDs.

Workflow

Figure 20: Deployment scenario for SRv6 MPLS dual-connected PE



These stages describe how an SRv6 MPLS dual-connected PE operates:

1. The PE router is configured for dual-mode L3VPN services.
   - The PE router is set up to support a specific VRF for L3VPN services, enabling both MPLS and SRv6 capabilities.
   - BGP is configured to enable MPLS label allocation and SRv6 SID allocation within the VRF's address family.
2. The PE router allocates identifiers for L3VPN prefixes. For each L3VPN prefix within the configured VRF, the PE router allocates both an MPLS label and an SRv6 SID.
3. The PE router advertises VRF prefixes.

   By default, if a VRF prefix has both an MPLS label and an SRv6 SID, the PE sends the MPLS label when advertising the prefix to the PE.
4. The PE router provides L3VPN connectivity.
   - The dual-connected PE provides MPLS/IPv4 L3VPN services between designated sites (for example, Site A and Site C).
   - The dual-connected PE also provides SRv6/IPv4 L3VPN services between other designated sites (for example, Site B and Site C).

Configure an SRv6 MPLS dual-connected PE

Use this procedure to configure a PE router to support IPv4 or IPv6 L3VPN services for a given VRF with both MPLS and SRv6, enabling an MPLS and SRv6 L3VPN coexistence scenario.

Procedure

Step 1 Enable MPLS label allocation.

Example:

```
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)#
```

Step 2 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.

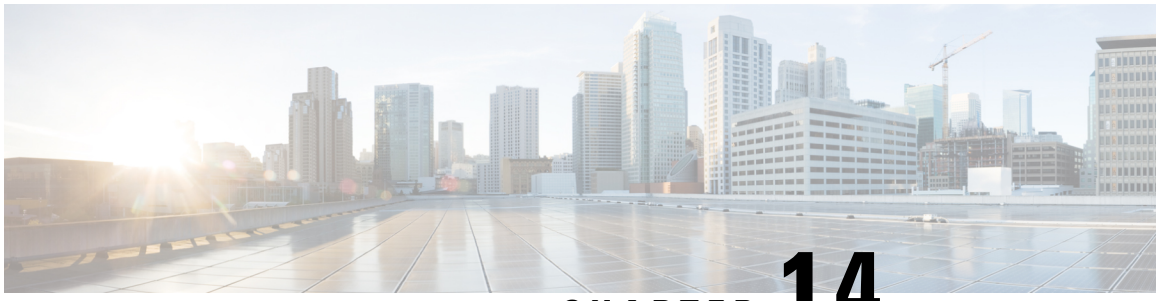
Example:

```
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
```

Step 3 Run the **show running-config** command to verify the configuration.

Example:

```
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
   !
  !
 !
!
```



CHAPTER 14

SRv6 Network Performance Measurement

This chapter is dedicated to Performance Measurement (PM) in SRv6 networks, a vital capability for monitoring network performance and ensuring Service Level Agreement (SLA) compliance. It explores various PM functionalities, including liveness monitoring for both IP endpoints and SR policies, and detailed delay measurement techniques (one-way, two-way, loopback). The chapter also introduces SRv6 path tracing, explaining how it records actual packet paths, measures per-hop delays, and identifies interface loads, providing comprehensive diagnostic tools for network operators.

Performance measurement

Performance measurement (PM) is a SRv6 feature that

- monitors network performance metrics such as packet loss, delay, delay variation, and bandwidth utilization
- provides network operators with information for performance evaluation and Service Level Agreement (SLA) compliance, and
- applies to links and end-to-end Traffic Engineering (TE) Label Switched Paths (LSPs) in service provider networks.

PM functionalities

This table details the functionalities supported by the PM feature for measuring delay across links or SR policies.

Table 37: Functionalities of SRv6 PM

Functionality	Details
Profiles	You can configure different profiles for different types of delay measurements. Delay profile allows you to schedule probe and configure metric advertisement parameters for delay measurement.

Functionality	Details
Protocols	The TWAMP Light from Appendix of RFC 5357 is standardized as Simple TWAMP in RFC 8762. Then it was extended with RFC 8972.
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.

PM probes typically follow the designated Segment Routing Traffic Engineering (SR-TE) path. However, in certain scenarios, the convergence of the PM probes and the SR-TE path may occur at different times. During this convergence period, PM probes may temporarily follow the IGP path and utilize an alternate egress interface until full convergence is achieved.

PM methods

PM is designed to monitor key network metrics, including packet loss, delay, delay variation, and bandwidth utilization. These measurements can be applied to individual links as well as end-to-end Segment Routing Traffic Engineering (SR-TE) Label Switched Paths (LSPs). By using these measurement methods, SRv6 enables comprehensive monitoring and optimization of network performance across various paths and endpoints. These methods are used to assess these metrics:

- Liveness monitoring: Verifies that a specific path, segment, or node is operational and capable of forwarding packets. This essential check for network availability and reliability supports both IP Endpoint liveness monitoring and SR policy liveness monitoring. For more information, see Liveness monitoring, on page 220.
  - IP Endpoint liveness monitoring: Ensures that a particular IP endpoint is reachable and operational within the network.
  - SR policy liveness monitoring: Verifies that traffic can be successfully forwarded along an SR policy path.
- Delay measurement: Measures the latency experienced by data packets as they travel through the network. For more information, see Delay measurement, on page 241.
  - IP Endpoint delay measurement: Tracks the time required for a packet to travel from the source device to a specific IP endpoint.

Liveness monitoring

Liveness is the ability of the network to confirm that a specific path, segment, or node is operational and capable of forwarding packets. It is essential for maintaining network availability and reliability. You can determine liveness for SR Policy and IP Endpoint.

Benefits of liveness monitoring

- Fault detection: You can quickly identify if a device is down, which allows for immediate response and troubleshooting.
- Load balancing: You can identify which network devices are live, so work can be distributed more evenly across the network. This prevents overloading of specific components and improves 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 also helps with maintenance planning, as system administrators can understand which components are operational or offline. They can then plan maintenance and downtime to minimize service disruption.
- Security: Regular liveness checks help maintain network security. Administrators can take proactive steps to mitigate damage and prevent future incidents by identifying unusual activity that might indicate a security breach or attack.

IP endpoint liveness monitoring

IP endpoint liveness is a network monitoring method that

- dynamically measures and verifies the availability of a device identified by an IP address
- sends probes or requests to the endpoint's IP address and awaits responses, and
- determines the endpoint's status based on receiving a response within a specified timeframe.

Table 38: Feature History Table

Feature Name	Release Information	Feature Description
Liveness Monitoring for IP Endpoint over SRv6 Network	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is now supported on: • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Liveness Monitoring for IP Endpoint over SRv6 Network	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) 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. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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)

Key concepts of IP endpoint liveness

- IP endpoint: An IP endpoint is any device in the network identified by an IPv4 or IPv6 address. The endpoint of a probe is defined by this IP address. 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. You can use the **performance-measurement endpoint** command to configure a probe endpoint source and destination addresses on a sender node. When the endpoint is reachable using SRv6, the forwarding stage imposes the SRv6 encapsulation.
- Probe: The probe is the request sent to verify liveness and the probe can be of various packet types, such as Ithat the endpoint responds to. 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*.
- 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. IP Endpoint segment list configuration is not supported under nondefault VRF.
  - VRF-awareness allows operators to deploy probes in these 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.

Usage guidelines for IP endpoint liveness monitoring

- Liveness session without segment list for an endpoint in a non-default VRF is not supported.
- IP Endpoint segment list configuration is not supported under nondefault VRF.
- SR PM endpoint session over BVI interface is not supported.

Supported and unsupported features for liveness monitoring for IP Endpoint over SRv6 Network

Table 39: 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*

How IP endpoint liveness detection works

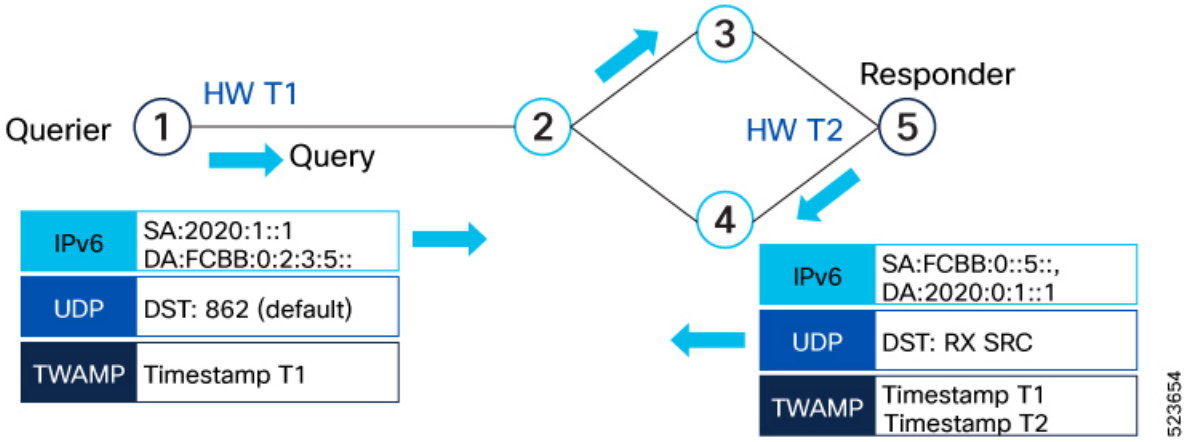
IP endpoint liveness detection leverages the loopback measurement mode to determine if an IP endpoint is active within an SRv6 network.

Summary

The key components involved in this process are the querier, the responder, and the SRv6 transport network. The process describes how the querier initiates and monitors PM TWAMP probe packets to determine the session's status.

Workflow

Figure 21: IP Endpoint Liveness In an SRv6 Network



These stages describe the sequence of events:

1. Probe creation and transmission: The querier creates and transmits the PM TWAMP probe packets based on the endpoint configuration.
2. Packet formation: The system forms the packet to reach the responder and return to the querier node over the SRv6 transport network.
3. Session up declaration: The querier node declares the session active once it receives the probe packet back
4. Session down declaration: If the sender node does not receive the specified number of consecutive probe packets, based on the configured multiplier, it declares the PM session inactive.

Configure IP Endpoint liveness

Procedure

Step 1 Configure basic IP endpoint liveness with a source address, endpoint, and a multiplier for liveness detection.

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)#
```

This 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
```

This 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
```

This 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
```

This 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
```

Step 2 Use the **show performance-measurement endpoint detail** to verify the IP endpoint liveness configuration.

Example:

```
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

SR policy liveness monitoring is a mechanism that

- verifies end-to-end traffic forwarding over an SRv6 policy candidate path
- periodically sends probe messages from the head-end router to the SRv6 Policy's endpoint router, and
- receives these probe messages back from the endpoint router without control-plane dependency.

Table 40: Feature History Table

Feature Name	Release Information	Feature Description
SR Policy Liveness Monitoring on Segment Routing over IPv6 (SRv6)	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.
SR Policy Liveness Monitoring on Segment Routing over IPv6 (SRv6)	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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.

Restrictions for SR policy liveness monitoring

- 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.

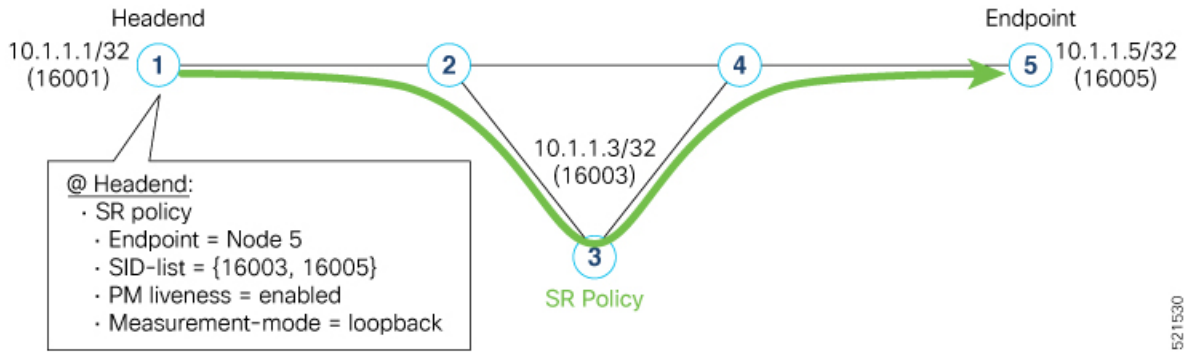
How SRv6 policy liveness detection works

This process describes the workflow for liveness detection over an SRv6 Policy. Consider an SRv6 policy programmed at a head-end node router (for example, Router 1) towards an endpoint node (example, Router 5). This SRv6 policy is enabled for liveness detection using the loopback measurement-mode.

Summary

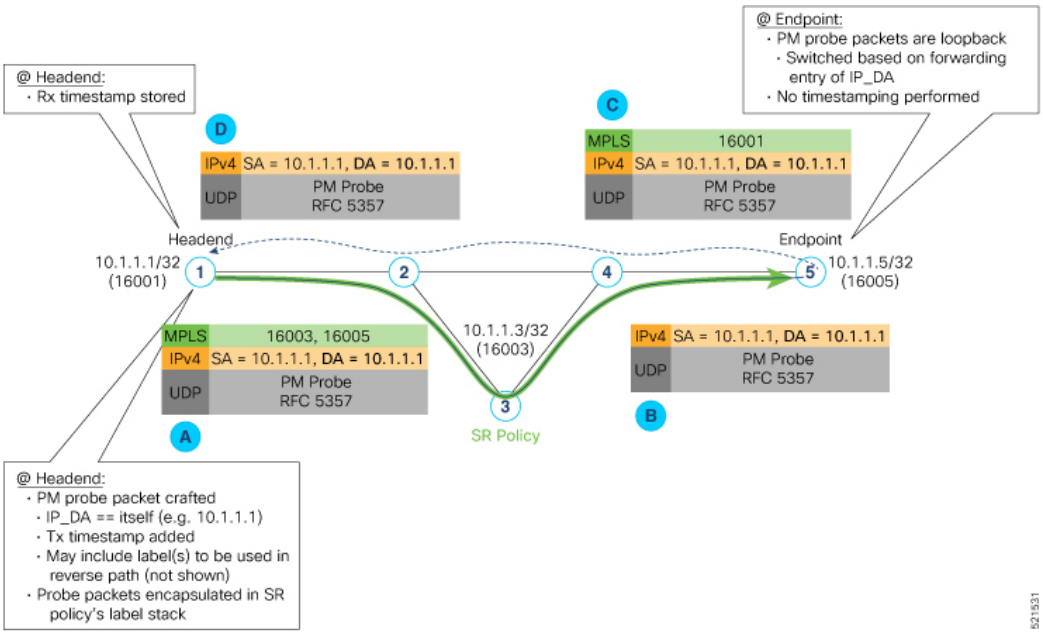
SRv6 policy liveness detection involves a head-end node sending PM probe packets through the network to an endpoint node. The endpoint node processes these packets and sends them back to the head-end. The head-end node then uses the received packets to determine the operational status of the SRv6 policy's candidate path.

Workflow



These stages describe how SRv6 policy liveness detection works:

1. 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 dataplane using the same encapsulation (label stack) of that of the SR policy being monitored.
2. The network delivers the PM probe packets as it would user packet for the SR policy.
3. 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 endpoint 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 endpoint node based on the forwarding entry associated with the top-most reverse path label.
4. Head-end node receives the PM probe packets.
   - A received (Rx) timestamp is 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 triggered.



Configure SR Policy Liveness Monitoring

Configuring SR Policy liveness monitoring involves configuring a performance measurement liveness profile to customize generic probe parameters and enabling liveness monitoring under SR Policy by associating a liveness profile, and customizing SR policy-specific probe parameters.

Procedure

Configure a performance measurement liveness profile to customize generic probe parameters and enable liveness monitoring under SR policy by associating a liveness profile, and customize SR policy-specific probe parameters.

- Configure a default SR-Policy PM liveness-profile

```
Router(config)# performance-measurement
Router(config-perf-meas)# liveness-profile sr-policy default
Router(config-pm-ld-srpolicy)# probe
Router(config-pm-ld-srpolicy-probe)# tx-interval 150000
Router(config-pm-ld-srpolicy-probe)# tos dscp 52
Router(config-pm-ld-srpolicy-probe)# exit
Router(config-pm-ld-srpolicy)# liveness-detection
Router(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

```
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

```
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:

```
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)# liveness-detection
Router(config-sr-te-policy-live-detect)# liveness-profile name sample-profile
Router(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:

```
Router(config)# segment-routing traffic-eng
Router(config-sr-te)# policy BAA
Router(config-sr-te-policy)# performance-measurement
Router(config-sr-te-policy-perf-meas)# liveness-detection
Router(config-sr-te-policy-live-detect)# liveness-profile name sample-profile
Router(config-sr-te-policy-live-detect)# invalidation-action down
Router(config-sr-te-policy-live-detect)# logging session-state-change
Router(config-sr-te-policy-live-detect)# exit
Router(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
```

Segment lists to activate candidate paths for PM Liveness

A minimum active segment lists is an SRv6 PM liveness feature that

- allows configuring the number of active segment lists required for a candidate path to be considered operational
- enables the head-end router to determine a candidate path's up status based on this configured minimum, and
- identifies a candidate path as up only when all segment lists are active if the configured minimum exceeds the total available segment lists in that path.

Table 41: Feature History Table

Feature Name	Release Information	Feature Description
Configure Segment Lists to Activate Candidate Paths in SRv6 for PM Liveness	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.
Configure Segment Lists to Activate Candidate Paths in SRv6 for PM Liveness	Release 24.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

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)

Configure the minimum number of segment lists in SRv6

Use this task to configure the minimum number of active segment lists required for a candidate path to be considered up within an SRv6 policy.

By default, the head-end router identifies a candidate path as operational only when all associated segment lists are active. This feature allows you to define a specific, lower threshold for the number of active segment lists.

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.

Procedure

Step 1 Activate three segment lists to have the PM liveness session up.

Example:

```
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
```

Step 2 Verify the running configuration after applying the minimum active segment list settings.

Example:

```
segment-routing
 traffic-eng
  policy po-103
   performance-measurement
    liveness-detection
     validation-cp minimum-active segment-lists 3
    !
   !
  !
 !
!
```

Step 3 Verify the configuration by displaying the detailed liveness information for the SR policy.

Example:

```
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
      Minimum 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
--------------------------------------------------------------------------------
```

Flow labels in SRv6 Header for PM liveness

Flow labels in SRv6 header for PM liveness is a SRv6 mechanism that utilizes 20-bit fields within the SRv6 header that

- monitors the activeness of multiple paths for a given segment list
- identifies different Equal-Cost Multi-Path (ECMP) paths, and
- is used exclusively with IPv6 probe packets.

Table 42: Feature History Table

Feature Name	Release Information	Feature Description
Configure Flow Labels in SRv6 Header for PM Liveness	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.
Configure Flow Labels in SRv6 Header for PM Liveness	Release 24.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

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-um-performance-measurement-cfg.yang` • `Cisco-IOS-XR-perf-meas-oper.yang` See (GitHub, Yang Data Models Navigator)

Configure flow labels in the SRv6 header

Use this task to configure flow labels in the SRv6 header. This enables you to monitor the activeness of multiple paths for a given segment list.

Procedure

Step 1 Configure flow labels in the SRv6 header in the global configuration mode.

Example:

```
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
```

Step 2 Use the show running configuration to verify the flow label configuration.

Example:

```
performance-measurement
 liveness-profile name name1
  probe
   flow-label from 0 to 1000000 increment 10
  !
 !
!
```

Step 3 Verify the SRv6 header flow label configuration by displaying the detailed liveness information for an SR policy.

Example:

```
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 measurement is a network performance monitoring method that

- measures the latency or delay experienced by data packets when they traverse a network
- uses the IP/UDP packet format defined in simple TWAMP using RFC8972 for probes, and
- employs time stamps applied at the echo destination (reflector) to enable greater accuracy for two-way or round-trip measurement capabilities.

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
Delay Measurement	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) Delay measurement in SR networks involves monitoring the end-to-end delay experienced by traffic sent over an SR policy. Link delay metrics such as average, minimum, and maximum delay, and delay variance are used to determine network latency. You can ensure compliance with Service Level Agreements (SLAs) by monitoring the end-to-end delay experienced by traffic. This feature is now supported on: • 8011-4G24Y4H-I

Two-Way Active Measurement Protocol

The Two-Way Active Measurement Protocol (TWAMP) is a network measurement protocol used for measuring two-way or round-trip IP performance metrics, such as latency (delay) and packet loss, between any two devices in a network. It adds two-way or round-trip measurement capabilities. 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 then receives the reflected test packets and collects two-way metrics. This architecture allows for the collection of two-way metrics.

Benefits of delay measurement

Delay measurement offers several key benefits for network management:

- 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.

Measurement modes

SRv6 performance measurement supports three distinct modes for measuring delay: One-way, Two-way, and Loopback.

Each mode offers different capabilities and hardware requirements to assess network latency.

Table 44: Measurement Mode Requirements

Measurement Mode	One-way	Two-way	Loopback
Description	One-way measurement mode is a delay measurement mode that offers the most precise form of one-way delay measurement.	Two-way measurement mode is a delay measurement mode that focuses on measuring round-trip network performance. It provides two-way delay measurements.	Loopback measurement mode is a delay measurement mode that utilizes a loopback mechanism for delay measurement. It provides both two-way and one-way delay measurements.
Formula used for calculation	Delay measurement in one-way mode is calculated as (T2 – T1).	Delay measurement in two-way mode is calculated as ((T4 – T1) – (T3 – T2))/2.	Delay measurements in Loopback mode are calculated as follows: • Round-Trip Delay = (T4 – T1) • One-Way Delay = Round-Trip Delay/2
Sender: **PTP-Capable HW and HW Timestamping**	Required	Required	Required
Reflector: **PTP-Capable HW and HW Timestamping**	Required	Required	Not Required
PTP Clock Synchronization between Sender and Reflector	Required	Not Required	Not Required

How one-way delay measurement works

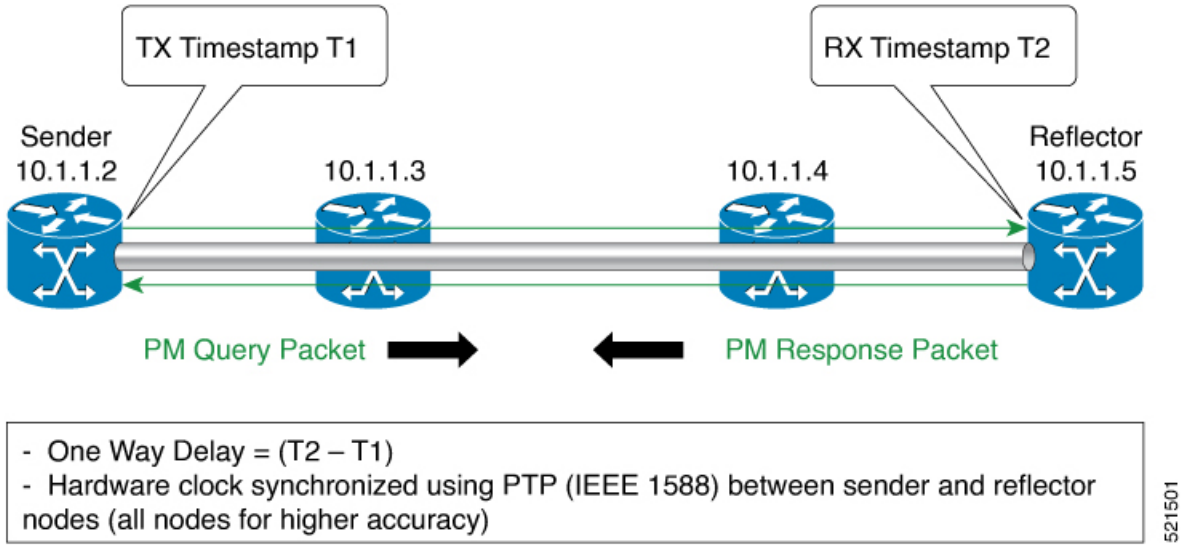
Delay measurement in one-way mode is calculated as (T2 – T1)

Summary

The one-way delay measurement process involves a local-end router and a remote-end router exchanging PM query and response packets with hardware timestamps to precisely calculate the one-way delay.

Workflow

Figure 22: One-Way



These stages describe how one-way delay measurement works:

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.

How loopback delay measurement works

Loopback measurement mode provides both two-way and one-way delay 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:

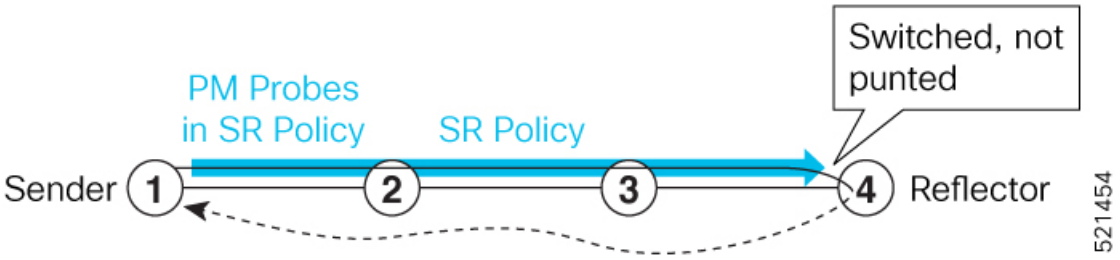
`Round-Trip Delay = (T4 - T1)` and `One-Way Delay = Round-Trip Delay/2.`

Summary

The loopback delay measurement process involves a local-end router sending probe packets that are looped back by an endpoint node without timestamping, allowing the local-end router to calculate round-trip and one-way delays.

Workflow

Figure 23: Loopback



These stages describe how loopback delay measurement works:

1. The local-end router sends PM probe packets periodically on the SR Policy.
2. The egress line card on the local-end router applies timestamps (T1) on these packets.
3. The probe packets are looped back on the endpoint node (not punted), with no timestamping performed on the endpoint node.
4. The local-end router timestamps the looped-back packet (T4) as soon as it is received.

How loopback delay measurement works

Loopback measurement mode provides both two-way and one-way delay 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:

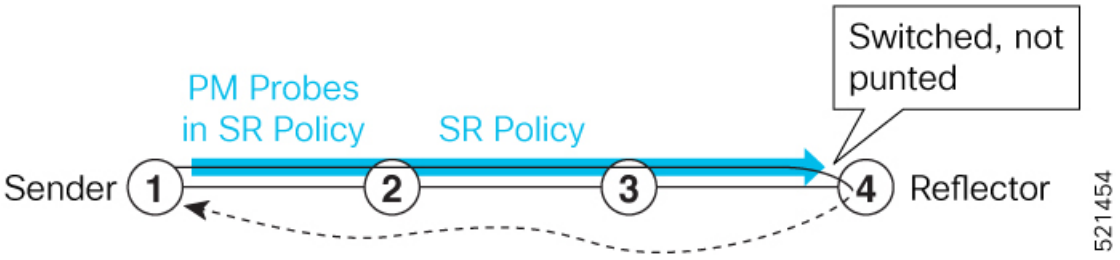
`Round-Trip Delay = (T4 - T1)` and `One-Way Delay = Round-Trip Delay/2.`

Summary

The loopback delay measurement process involves a local-end router sending probe packets that are looped back by an endpoint node without timestamping, allowing the local-end router to calculate round-trip and one-way delays.

Workflow

Figure 24: Loopback



These stages describe how loopback delay measurement works:

1. The local-end router sends PM probe packets periodically on the SR Policy.
2. The egress line card on the local-end router applies timestamps (T1) on these packets.

3. The probe packets are looped back on the endpoint node (not punted), with no timestamping performed on the endpoint node.
4. The local-end router timestamps the looped-back packet (T4) as soon as it is received.

Delay measurement for IP Endpoint

Delay for an IP endpoint is the amount of time that

- it takes for a data packet to travel from a source device to a specific IP endpoint within a network
- is measured by sending a probe packet from a source device to the target IP endpoint and recording the time from departure to arrival, and
- can be measured as one-way, two-way, roundtrip, or in loop-back mode.

Table 45: Feature History Table

Feature Name	Release Information	Feature Description
Delay Measurement for IP Endpoint over SRv6 Network	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) * This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

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)

Supported features for delay measurement for IP Endpoint

- 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)

IP endpoint probe statistics collection

- 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)

- These 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`

Usage guidelines for delay measurement for IP Endpoint

PTP-capable hardware

SR PM is supported only on hardware that supports Precision Time Protocol (PTP). This requirement applies to both one-way and two-way delay measurement.

Custom segment lists for delay measurement probes

You can specify a custom labeled path through one or more user-configured segment-lists. A user-configured segment-list defines the forwarding path from the sender to the reflector when the probe operates in delay-measurement mode. Examples of such custom segment lists include:

- A segment-list that includes a Flex-Algo prefix SID of the endpoint.
- A segment-list that includes a SID-list with labels to reach the endpoint or the sender (forward direction).
- A segment-list that includes a BSID associated with an SR policy to reach the endpoint.

Unsupported features

These features are not supported for delay measurement for IP Endpoint:

- Endpoint segment list configuration under a nondefault VRF.
- Liveness sessions without a segment list for an endpoint in a non-default VRF.
- SR Performance Measurement endpoint sessions over a BVI interface.

Configure IP Endpoint delay measurement over SRv6 network

Procedure

Step 1 Configure the IP Endpoint delay measurement.

Example:

```
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 delay 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
```

Step 2 Verify the show running configuration.

Example:

```
performance-measurement
 endpoint ipv6 FCBB:0:1::
  delay-measurement
   delay-profile name test
  !
 !
 liveness-profile name test
  probe
   flow-label explicit 100 200 300
  !
 !
!
```

Step 3 Verify the delay information for the endpoint.

Example:

```
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
```

Path tracing in SRv6 Network

An SRv6 path tracing is a network diagnostic feature that

- records the actual packet path as a sequence of interface IDs and timestamps
- measures end-to-end delay and per-hop delay between network nodes, and
- identifies the load on each egress interface along the packet delivery path.

Table 46: Feature History Table

Feature Name	Release	Description
Path Tracing Midpoint Node	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is now supported on: • 8011-4G24Y4H-I
Path Tracing Source and Sink Nodes	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.
Path Tracing Midpoint Node	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Path Tracing Source and Sink Nodes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

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.

Benefits of path tracing

- 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

Usage Guidelines for SRv6 Path Tracing

Be aware of these limitations and supported configurations when implementing path tracing to ensure expected functionality.

- Support for PT Midpoint, PT Source, and PT Sink functionalities starts from Cisco IOS XR Release 24.1.1.
- PT Source and Sink nodes are not supported. 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.
- TI-FLA on SRv6 midpoint node with PT is not supported. MCD is updated first, then the TILFA encap with new IPv6 header is added on top of it.
- 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.

How SRv6 Path Tracing works

Summary

Actors involved in the path tracing process are:

- PT Source Node: Generates and injects probe packets towards a destination node.
- PT Midpoint Node: Transit nodes that perform IPv6 routing and may record or export Path Tracing information. This category includes:
  - PT-Aware Midpoint: Records PT information (MCD) in the HbH-PT header.
  - PT-Legacy Midpoint: Exports PT information directly to the collector.
  - PT-Unaware Midpoint: Performs routing without recording or exporting PT information.
- PT Sink Node: Receives PT probes from the Source node, records its own PT information, and forwards them to a Regional Collector.
- Regional Collector (RC): Receives PT probes, parses them, reconstructs the packet delivery path, and stores the data.

The Path Tracing process begins with a PT Source Node injecting probe packets into the network. These probes travel through various PT Midpoint Nodes, which may add their tracing data to a Hop-by-Hop Path Tracing (HbH-PT) header or export it directly. The PT Sink Node receives these probes, adds its own information, and sends them to a Regional Collector (RC). The RC then reconstructs the complete packet delivery path and timing details from the collected data.

Workflow

These stages describe how path tracing works.

1. PT source node: Initiates a PT session by generating and injecting probe packets towards a destination.
2. PT midpoint node: A transit node that performs IPv6 routing and, depending on its capability, records or exports path tracing information.
   - PT-aware midpoint: Records its PT information (Midpoint Compressed Data - MCD) into the probe's Hop-by-Hop Path Tracing (HbH-PT) header.
   - PT-legacy midpoint: Exports its PT information directly to a collector.
   - PT-unaware midpoint: Forwards probes without recording or exporting PT information.
3. PT sink node: Receives PT probes, records its own Path Tracing information, and forwards the complete data to a Regional Collector.
4. Regional collector (RC): Collects, parses, and reconstructs the packet delivery path from received PT probes for storage and analysis.

Configure path tracing in SRv6 network

To enable and configure path tracing functionality across different network roles (source, midpoint, sink) for monitoring packet paths.

Path tracing provides a log of the packet path, including interface IDs, timestamps, and delay metrics. This task outlines the necessary configurations for each node type to participate in a Path Tracing session.

Procedure

Step 1 Configuration example of Source node:

a) Configure the performance measurement endpoint for path tracing and path assurance

Example:

```
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
```

b) Configure probe profile parameters.

Example:

```
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$
```

c) Configure Interface ID under path-tracing for the source node and for it to participate in the MCD updates inside the probe packets:

Example:

```
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
```

d) Verify the running configuration.

Example:

- 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
   !
  !
 !
!
```

Step 2 Configure the Path Tracing midpoint node.

a) Configure the Interface ID under Path-tracing for the Midpoint node and for it to participate in the MCD updates inside the probe packets:

Example:

```
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
```

Step 3 Configure the Path Tracing sink node.

a) Configure static routing for the SRv6 endpoint behavior.

Example:

```
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::
!
```

b) Configure Interface ID under Path-tracing for the Sink node and for it to participate in the MCD updates inside the probe packets.

Example:

```
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
```

c) Verify the running configuration.

Example:

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
   !
  !
 !
!
```

Step 4 Verify the path tracing configuration.

It is good to check the target interface configuration and performance-measurement configuration for that interface. Verify using the show commands listed to check if the PT configuration is applied to the interface properly.

Example:

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:

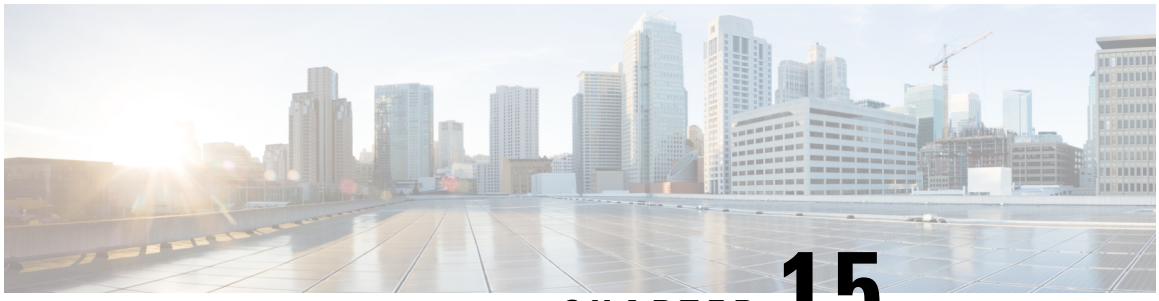
```
Router# show cef interface GigabitEthernet 0/2/0/0
GigabitEthernet0/2/0/0 is up if_handle 0x01000020 if_type IFT_GETHERNET(0xf)
  idb info 0x619f16f0 flags 0x30101 ext 0x627ef180 flags 0x30050
  Vrf Local Info (0x626510f0)
```

```
  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)
```



CHAPTER 15

SRv6 Traffic Accounting

Effective management of Segment Routing over IPv6 (SRv6) networks necessitates granular visibility into traffic flow and resource consumption. This chapter addresses the critical requirement for traffic accounting within an SRv6 domain, leveraging the inherent programmability and explicit path control offered by the architecture. It details the methodologies, mechanisms, and tools employed to systematically monitor and quantify data traversing specific segments, policies, and Segment Identifiers (SIDs). The objective is to provide actionable data essential for optimizing network resource utilization, validating Service Level Agreements (SLAs), facilitating informed capacity planning, and enabling precise troubleshooting within complex SRv6 deployments.

SRv6 traffic accounting

SRv6 traffic accounting is a network management solution that:

- enables routers to record the number of packets and bytes transmitted on specific egress interfaces for IPv6 traffic using per-locator SRv6 counters
- monitors aggregated traffic flows as they enter, traverse, and leave the SRv6-enabled network, enabling traffic analysis from ingress to egress nodes, and
- provides essential data for capacity planning, traffic forecasting, SLA compliance, and anticipating network failures or expansion needs.

Per-locator SRv6 counters are measurement tools within SRv6 traffic accounting that track packets and bytes for each SRv6 locator, providing granular insight into traffic volumes for detailed monitoring and capacity planning.

Table 47: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Traffic Accounting	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SRv6 Traffic Accounting	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
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)

Importance of SRv6 traffic accounting

Modern networks handle increasing traffic volumes and require precise, real-time insights into how traffic moves through the SRv6 domain. Without accurate traffic accounting, operators risk congestion, inefficient resource allocation, difficulty meeting SLAs, and challenges in planning for growth or failures.

Benefits of SRv6 traffic accounting

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).

Usage guidelines for SRv6 traffic accounting

Supported traffic types

- IPv6 packets.
- SRv6 VPNv4
- SRv6 VPNv6
- SRv6 INETv4
- SRv6 INETv6

SRv6 packet accounting behavior

- 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.

Measurement and counting details

- 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.

Configuration requirements

- Supports a minimum telemetry pull interval of 30 seconds.

Configuration requirements

- 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.

How SRv6 traffic accounting works

Summary

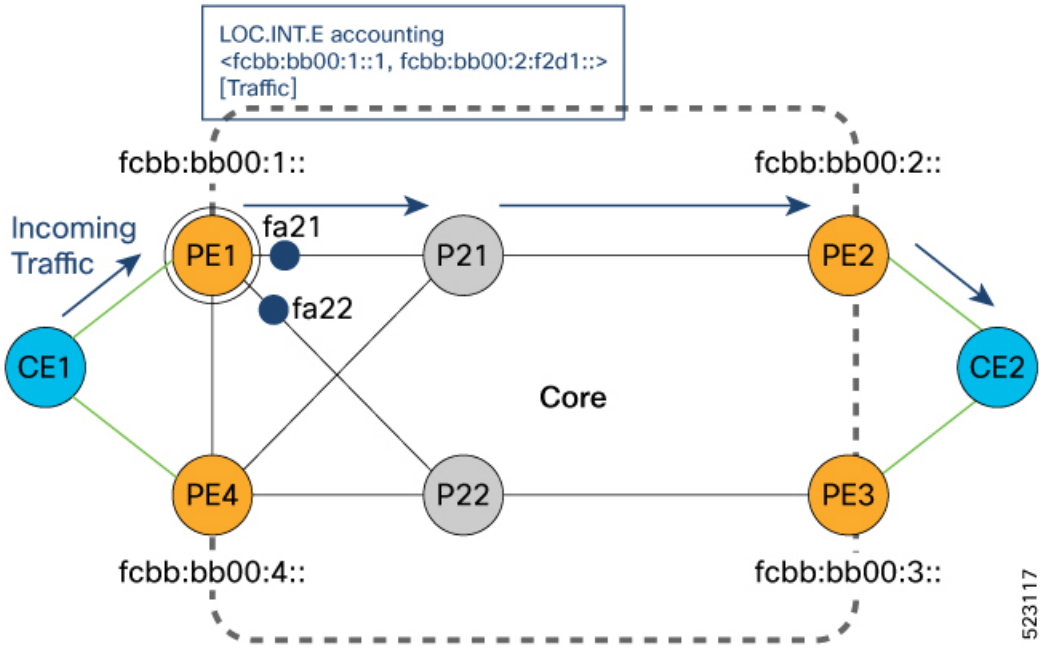
The key components that are involved in the SRv6 locator counters and traffic accounting process are:

- CE1 (Customer Edge 1): An endpoint that originates external traffic in the network topology.
- CE2 (Customer Edge 2): An endpoint that receives both external and internal traffic.
- PE1 (Provider Edge 1): A network device that learns reachability, imposes traffic with SRv6 locators, and performs external traffic accounting.
- PE4 (Provider Edge 4): A network device that originates internal traffic destined for other parts of the network, typically passing through provider edge devices.
- P21 and P22: Specific interfaces or paths on PE1 that are used for forwarding traffic, contributing to the ECMP (Equal-Cost Multi-Path) setup.
- SRv6 Locator Counters (LOC.INT.E): The mechanism used for accounting traffic on a per-prefix and per-egress interface basis within the SRv6 domain.
- Demand Matrix (DM): The Demand Matrix (DM) 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.

The SRv6 locator counters and traffic accounting process involves tracking traffic flows and quantifying data transmission. The key components work together to track external and internal traffic flows and calculate the net external traffic, providing a comprehensive view of network traffic distribution. 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.

Workflow

Figure 25: Sample Topology for SRv6 Traffic Accounting



These stages describe how SRv6 locator counters and traffic accounting work.

1. 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::.

2. 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]
   ```

3. 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.

4. 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. The traffic from PE4 to PE1 is considered as internal traffic.

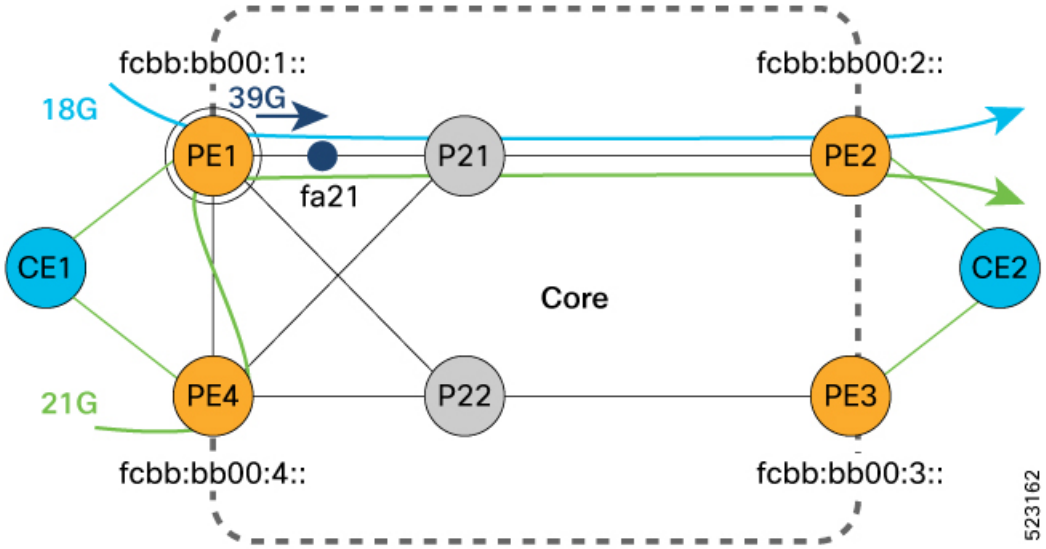
5. When traffic reaches PE1, it looks for the PE2 locator ID and keeps the traffic count at PE1 when the traffic exit the fa21 interface.

6. SRv6 traffic is calculated using the demand matrix.

   Let's understand the calculation with an example. In the topology, the amount of external traffic destined for PE2 is a combination of external and internal traffic.

Note The traffic transmitted from PE1 is marked in blue. The traffic transmitted from PE4 is marked as in green.

Figure 26: SampleTopology for Demand Matrix



The amount of external traffic that PE2 receives is equal to the total traffic sent out from PE1 minus the received internal traffic.

Let's consider that 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.

The formula for external traffic to PE2 is applied:

```
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)


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
```

Table 48: 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 PE2 recieves 18 gigabits per second external traffic from PE1
PE4	21- (18 + 0 + 0) = 3 gigabits per second	39 - (18 + 0 + 0) = 21 gigabits per second

Configure SRv6 traffic accounting

Before you begin

Ensure that you enable SRv6 and its services.

Follow these steps to configure SRv6 traffic accounting on the router.

Procedure

Step 1 Enable SRv6 traffic accounting

Example:

```
Router#configure
Router(config)#accounting
Router(config-acct)#prefixes ipv6 mode per-prefix per-nexthop srv6-locators
Router(config-acct)#commit
```

Step 2 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.

Example:

```
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.
```

Step 3 Configure the sensory path to retrieve the accounting data using telemetry:

Example:

```
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#
```

Step 4 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.

Example:

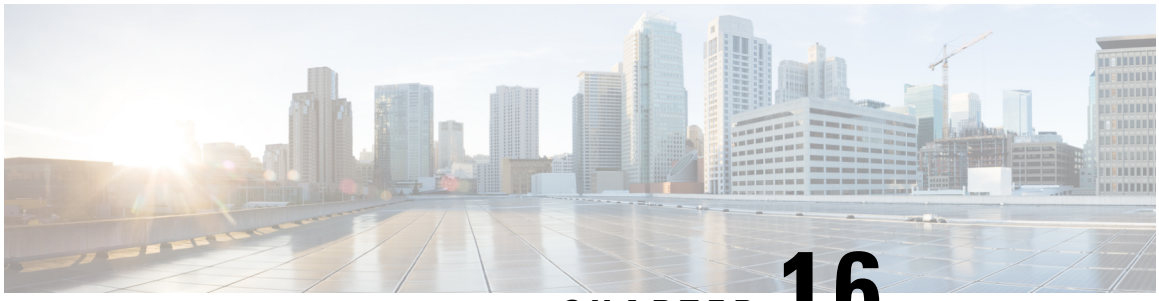
```
{
"Cisco-IOS-XR-fib-common-oper:cef-accounting": {
  "vrfs": {
   "vrf": [
    {
     "vrf-name": "default",
     "afis": {
      "afi": [
       {
        "afi-type": "ipv6",
        "pfx": {
         "srv6locs": {
          "srv6loc": [
           {
            "ipv6-address": " fccc:cc00:1::",
            "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"
            }
           ]
          }
         }
        }
       ]
      }
     }
    ]
   }
}
}
```

Step 5 Run **showcef ipv6 accounting** command to display the packets per bytes:

Example:

```
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
```



CHAPTER 16

SRv6 uSID Migration

This chapter introduces the Full-Replace Migration to SRv6 Micro-SID (f3216), a feature enabling network operators to seamlessly transition existing SRv6 deployments from the legacy "format1" full-length SIDs to the more efficient f3216 uSIDs.

Full-replace migration to SRv6 uSID

The full-replace migration is a network migration process that

- enables incremental migration from format1 to f3216 using the Ship in the Night method
- migrates both the underlay and overlay from format1 to uSID f3216, and
- minimizes service disruption and ensures seamless migration by replacing the initial SRv6 locator with another SRv6 locator.

The Ship in the Night method is a migration approach that enables you to introduce segment routing into your environment incrementally while allowing existing transport protocols to continue operating independently until you are ready to phase them out, thereby minimizing service disruption and supporting a seamless migration process.

Table 49: Feature History Table

Feature Name	Release Information	Feature Description
Full-Replace Migration to SRv6 Micro-SID	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature 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.

The services migration is performed using a replacement procedure, where the initial SRv6 locator is replaced by another SRv6 locator. The format1 to f3216 migration minimizes traffic loss, requires minimal configurations, and no IETF signaling extensions.

Supported configurations and modes

These are supported modes for migration from format1 to u-SIDs:

- Base: SRv6 classic mode with only format1 supported.
- Dual: SRv6 classic with format1 and SRv6 uSID with f3216 will both coexist.
- f3216: uSID format. f3216 represents the format 3216, which is 32-bit block and 16-bit IDs.
- The router supports IS-IS underlay configurations, including TILFA, uLoop, and FlexAlgo, for migration from format1 to uSIDs.

Migration states

The migration starts with SRv6 base format1 and ends with SRv6 uSID f3216. The migration states are:

- Initial state: This is the initial migration state of a deployment. This state comprises SRv6 base with format1.
- In-migration state: The migration procedure is initiated, and in progress. This state comprises SRv6 in dual mode (base with format1, and uSID with f3216). While this state may be maintained for a period of time, it is recommended to progress to the next migration state as soon as possible.
- 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. In the Full-Replace migration end state, both underlay and overlay are migrated to uSID f3216.

Restrictions for full-replace migration to SRv6 micro-SID

Line card reload

You need to reload the line cards as the hardware profiles go through multiple transitions during the Full-Replace migration to SRv6 uSID.

Delayed Delete Command

You can overcome the traffic drop duration during the swap from format1 by f3216 on a PE node, depending on the BGP/EVPN convergence, by using the **delayed_delete** command. When the **delayed_delete** command

is configured against the format1 SID locator, the RIB notifies EVPN about this change. The EVPN in turn stores the delayed flag in its RIB locator database.

How uSID f3216 migration works

Summary

The uSID f3216 migration involves Network nodes, IS-IS protocol, P nodes, PE nodes, BGP, EVPN, and SRv6. The process begins with network nodes being upgraded to support uSID f3216 and allow coexistence. Subsequently, IS-IS on P and PE nodes is configured to incrementally shift the underlay to f3216 while temporarily coexisting with format1. Finally, BGP and EVPN configurations on PE nodes are updated to migrate overlay services, with SRv6 managing the allocation and deallocation of SIDs. This systematic approach ensures a seamless transition to a fully uSID f3216-based network architecture.

Workflow

These stages describe how the uSID f3216 migration process works.

1. Prepare for migration: Upgrade the network nodes to an image that is uSID f3216 capable, and allows the coexistence of format1 and f3216
2. Migrate the underlay to uSID: IS-IS as an underlay protocol was already configured on P and PE nodes using format1 locators. In this migration step, add f3216 locators in the IS-IS configuration, in addition to the format1 locators. Both format1 and f3216 endpoint SIDs are concurrently allocated, installed, and announced during this stage. f3216 is the preferred option over format1 for underlay paths.

   After the f3216 locators have been configured on all P and PE nodes, the format1 locators are removed from the P nodes. The overlay services still use the format1 locators after completing this migration step, therefore, the format1 locator configuration must be maintained on the PE nodes.

   At the end of this step, the migration status of the P nodes is:

   - Locator reachability: f3216 only
   - Underlay endpoints/headends: f3216 only

   At the end of this step, the migration status of the PE Nodes is:

   - Locator reachability: format1 and f3216
   - Underlay endpoints/headends: f3216 only
   - Overlay endpoints/headends: format1

3. Migrate the overlay to uSID: Enable overlay f3216 locators under BGP and EVPN on all PE nodes. The BGP and EVPN configurations replace the format1 locator with the f3216 locator. During this stage, the f3216 uSIDs are allocated, installed, and announced, while the format1 SIDs are deallocated, uninstalled, and withdrawn.

   For a transient period, BGP and EVPN might have some paths with a format1 SID and some with an f3216 SID.

   The format1 locators are removed from the underlay after all the overlay traffic has converged to use f3216 SIDs. By unconfiguring the format1 locators from BGP and EVPN, they are deleted from SRv6.

At the end of this step, the migration status of the P/PE Nodes is:

- Locator reachability: f3216 only
- Underlay endpoints/headends: f3216 only
- Overlay endpoints/headends: f3216 only

4. Final 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. In the Full-Replace migration end state, both underlay and overlay are migrated to uSID f3216.

Migrate an SRv6 network from format1 to uSID

Migrate the network from SRv6 base format1 to uSID f3216. This migration enables faster convergence and seamless traffic transition.

Upgrade your routers to support SRv6 uSID `f3216`, which allows coexistence and phased migration from format1. This procedure guides network administrators through each transition stage: initial, in-migration, and end state.

Before you begin

- Upgrade all network nodes (P and PE) to a software image that supports SRv6 uSID `f3216` and allows the coexistence of `format1` and `f3216` locators.
- Ensure IS-IS, BGP, and EVPN protocols are configured and operational with SRv6 `format1` locators.

Procedure

Step 1 Configure SRv6 locators in the initial state using format1 only.

This step sets up the base SRv6 configuration with format1 locators, if they are not already present.

Example:

```
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
```

Step 2 Configure IS-IS with the SRv6 format1 locator in the initial state.

This step integrates IS-IS with the format1 SRv6 locator.

Example:

```
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
```

Step 3 Configure BGP and EVPN with SRv6 locator in the initial state.

This step integrates BGP and EVPN with the format1 SRv6 locator.

Example:

```
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
```

Step 4 Initiate in-migration to dual mode by configuring the uSID locator.

This step configures the uSID locator, enabling dual mode operation. The `delayed-delete` command ensures the format1 locator remains active during the transition.

Example:

```
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
```

Step 5 Update IS-IS and BGP/EVPN to use both locators during in-migration.

This step modifies the IS-IS and BGP and EVPN configuration to recognize and use both locators.

Example:

```
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

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
```

Step 6 Finalize end-state configuration by retaining only uSID locators.

Example:

```
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
```

```
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


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
```

show

commands

Step 7 Verify SRv6 migration status to confirm that the SRv6 deployment is operating correctly with uSID locators.

Example:

```
RP/0/RSP0/CPU0:Router# show route ipv6 fc00:cc30:600:e004:: detail

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.
```