

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

Note IOS XR release 7.3.2 supports SRv6 with Full-length SID and Micro-SID formats; however, only one format is supported in the network at a time.

To use SRv6 Micro-SID (uSID), globally enable SRv6 and configure the 48-bit locator. See Configuring SRv6, on page 20.

To use SRv6 Full-length SID, see the Configure Segment Routing over IPv6 (SRv6) with Full-Length SIDs.

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

- Segment Routing over IPv6 Overview, on page 2
- SRv6 Micro-Segment (uSID), on page 6
- Usage Guidelines and Limitations, on page 18
- Configuring SRv6, on page 20
- Configuring SRv6 under IS-IS, on page 24
- Configuring SRv6 Flexible Algorithm under IS-IS, on page 26
- Configuring SRv6 Locator Prefix Summarization, on page 28
- Configuring TI-LFA with SRv6 IS-IS, on page 29
- Configuring SRv6 IS-IS Microloop Avoidance, on page 31
- Configuring Static SIDs, on page 32
- Configuring Explicit SRv6 uSID Allocation Start Range, on page 37
- Configuring SRv6 BGP-Based Services, on page 38
- Static SRv6 pseudo-wire, on page 110
- SRv6/MPLS L3 Service Interworking Gateway, on page 114
- L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway, on page 120
- L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway, on page 123
- Layer 3 service gateway for interconnecting SRv6 domains, on page 126
- SRv6/MPLS Dual-Connected PE, on page 135
- SRv6 Provider Edge (PE) Lite Support, on page 137
- SRv6 SID Information in BGP-LS Reporting, on page 143
- DHCPv4 Relay Agent and Proxy Support over SRv6, on page 143

• DHCPv6 Relay Agent Support over SRv6, on page 144
• Full-Replace Migration to SRv6 Micro-SID, on page 144
• SRv6 Traffic Accounting, on page 148
• Path Maximum Transmission Unit (MTU) Discovery for SRv6 Encapsulated Packets, on page 156
• VRF-to-VRF route leaking in SRv6 core, on page 158
• BGP SID locator tracking for SRv6 services, on page 163
• SRv6-TE explicit BSIDs and remote steering, on page 169
• SRv6 Flow Label entropy via source address, on page 175

Segment Routing over IPv6 Overview

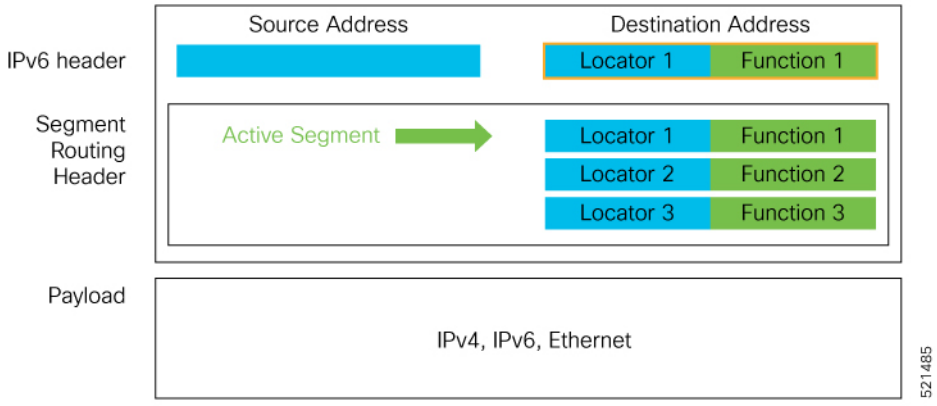
Segment Routing (SR) can be applied on both MPLS and IPv6 data planes. Segment Routing over IPv6 (SRv6) extends Segment Routing support with IPv6 data plane.

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

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

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

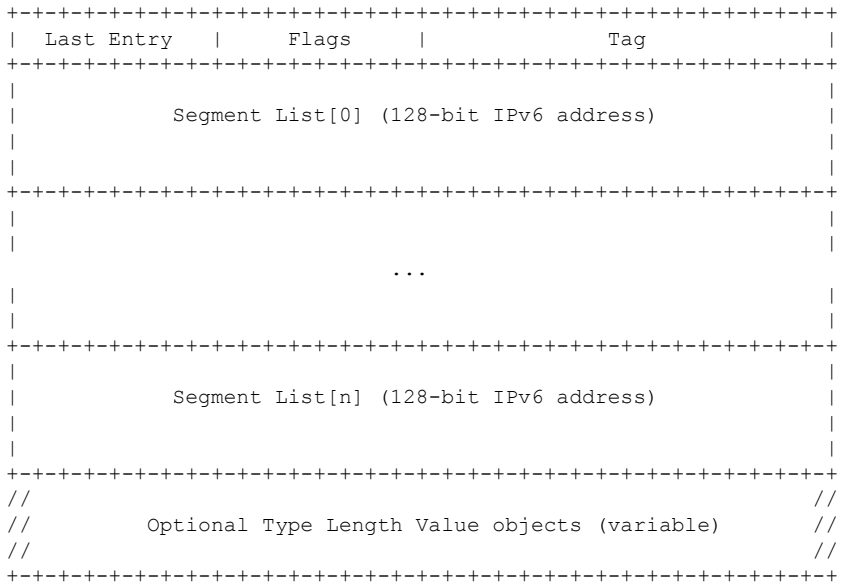
Figure 1: Network Program in the Packet Header



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

The SRH is defined as follows:

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



The following list explains the fields in SRH:

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

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

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

The locator part can be further divided into two parts:

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

- Node Id: This field is the node designator in an SRv6 network and is the least significant bit (LSB) portion of a locator subnet.

SRv6 Node Roles

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

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

SRv6 Head-End Behaviors

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

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

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

This section describes a set of SR Policy headend behaviors. The following list summarizes them:

- H.Encaps—SR Headend Behavior with Encapsulation in an SRv6 Policy
- H.Encaps.Red—H.Encaps with Reduced Encapsulation
- H.Insert—SR Headend with insertion of an SRv6 Policy
- H.Insert.Red—H.Insert with reduced insertion

SRv6 Endpoint Behaviors

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

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

- End—Endpoint function. The SRv6 instantiation of a Prefix SID [RFC8402].
- End.X—Endpoint with Layer-3 cross-connect. The SRv6 instantiation of an Adj SID [RFC8402].
- End.DX6—Endpoint with decapsulation and IPv6 cross-connect (IPv6-L3VPN - equivalent to per-CE VPN label).
- End.DX4—Endpoint with decapsulation and IPv4 cross-connect (IPv4-L3VPN - equivalent to per-CE VPN label).
- End.DT6—Endpoint with decapsulation and IPv6 table lookup (IPv6-L3VPN - equivalent to per-VRF VPN label).
- End.DT4—Endpoint with decapsulation and IPv4 table lookup (IPv4-L3VPN - equivalent to per-VRF VPN label).

- End.DT46—Endpoint with decapsulation and specific IP table lookup (IP-L3VPN - equivalent to per-VRF VPN label).
- End.DX2—Endpoint with decapsulation and L2 cross-connect (L2VPN use-case).
- End.B6.Encaps—Endpoint bound to an SRv6 policy with encapsulation. SRv6 instantiation of a Binding SID.
- End.B6.Encaps.RED—End.B6.Encaps with reduced SRH. SRv6 instantiation of a Binding SID.

SRv6 Endpoint Behavior Variants

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

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

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

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

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

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

  If Segments Left is 0, then:

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

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

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

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

    1. Remove the outer IPv6 Header with all its extension headers
    2. Submit the packet to the egress IPv6 FIB lookup and transmission to the new destination
    3. Else, if the Upper-layer Header type is 4 (IPv4)
    4. Remove the outer IPv6 Header with all its extension headers

5. Submit the packet to the egress IPv4 FIB lookup and transmission to the new destination
6. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

- **End.X** behavior: If the Upper-layer Header type is 41 (IPv6) or 4 (IPv4), then:
  1. Remove the outer IPv6 Header with all its extension headers
  2. Forward the exposed IP packet to the L3 adjacency J
  3. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

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

SRv6 Micro-Segment (uSID)

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
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, 6 uSIDs per uSID carrier results in 18 source-routing waypoints in only 40 bytes of overhead (in SRH).

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

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

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

The SRv6 uSID provides the following benefits:

- Leverages the SRv6 Network Programming with no change. SRv6 uSID is a new pseudo code in the existing SRv6 network programming framework.
- Leverages the SRv6 data plane (SRH) with no change. Any SID in the destination address or SRH can be an SRv6 uSID carrier.

- Leverages the SRv6 control plane with no change.
- Ultra-Scale—Scalable number of globally unique nodes in the domain, for example:
  - 16-bit uSID ID size: 65k uSIDs per domain block
  - 32-bit uSID ID size: 4.3M uSIDs per domain block
- Lowest MTU overhead
  - 6 uSIDs per uSID carrier
  - For example, 18 source-routing waypoints in only 40 bytes of overhead
- Hardware-friendliness:
  - Leverages mature hardware capabilities (inline IP Destination Address edit, IP Destination Address longest match).
  - Avoids any extra lookup in indexed mapping tables.
  - A micro-program with 6 or fewer uSIDs requires only legacy IP-in-IP encapsulation behavior.
- Scalable Control Plane:
  - Summarization at area/domain boundary provides massive scaling advantage.
  - No routing extension is required, a simple prefix advertisement suffices.
- Seamless Deployment:
  - A uSID may be used as a SID (the carrier holds a single uSID).
  - The inner structure of an SR Policy can stay opaque to the source. A carrier with uSIDs is just seen as a SID by the policy headend Security.
  - Leverages SRv6's native SR domain security.

SRv6 uSID Terminology

The SRv6 Network Programming is extended with the following terms:

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

  A uSID has an associated behavior that is the SRv6 function (for example, a node SID or Adjacency SID) associated with the given ID. The node at which an uSID is instantiated is called the “Parent” node.
- uSID Carrier—A 128-bit IPv6 address (carried in either in the packet destination address or in the SRH) in the following format:

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

  where:
  - uSID Block—An IPv6 prefix that defines a block of SRv6 uSIDs.
  - Active uSID—The first uSID that follows the uSID block.
  - Next uSID—The next uSID after the Active uSID.

- Last uSID—The last uSID in the carrier before the End-of-Carrier uSID.
- End-of-Carrier —A globally reserved uSID that marks the end of a uSID carrier. The End-of-Carrier ID is **0000**. All empty uSID carrier positions must be filled with the End-of-Carrier ID; therefore, a uSID carrier can have more than one End-of-Carrier.

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

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

SRv6 uSID Carrier Format

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

- **uSID Block**

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

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

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

- **uSID ID**

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

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

Note F3216 is the default format, and the only format that is supported in IOS XR 7.3.1 release.

SRv6 uSID Allocation Within a uSID Block

Table 2: Feature History Table

Feature Name	Release	Description
Wide LIB uSID Allocation for End.DT46 SRv6 SIDs	Release 7.8.1	This feature introduces support for Wide Local ID block (W-LIB). W-LIB provides an extended set of IDs available for local uSID allocation that can be used when a PE with large-scale pseudowire termination requires more local uSIDs than provided from the LIB. W-LIB uSID allocation is supported for End.DT46 SRv6 SIDs.

Key Concepts and Terminologies

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

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

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

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

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

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

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

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

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

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

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

Example: uSID Allocation

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

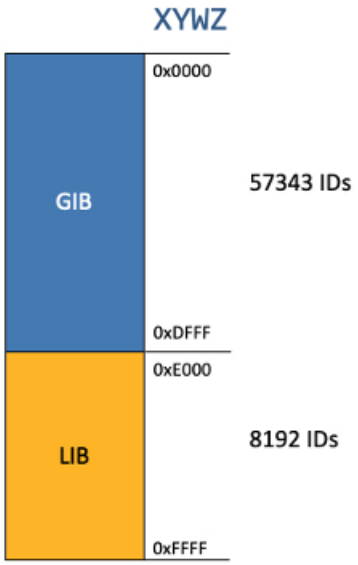
Consider the following example:

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

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

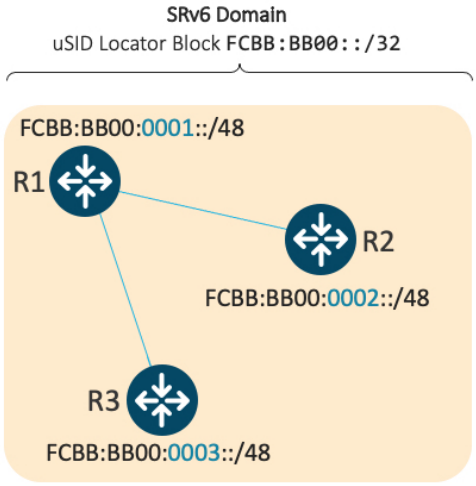
A uSID is allocated from the GIB or LIB of block FCBB:BB00::/32 depending on the value of the "**X**" nibble:

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



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

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



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

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

The underlay uSIDs present on R1 are:

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

GIB and LIB – IOS-XR Implementation

The following functionality is supported:

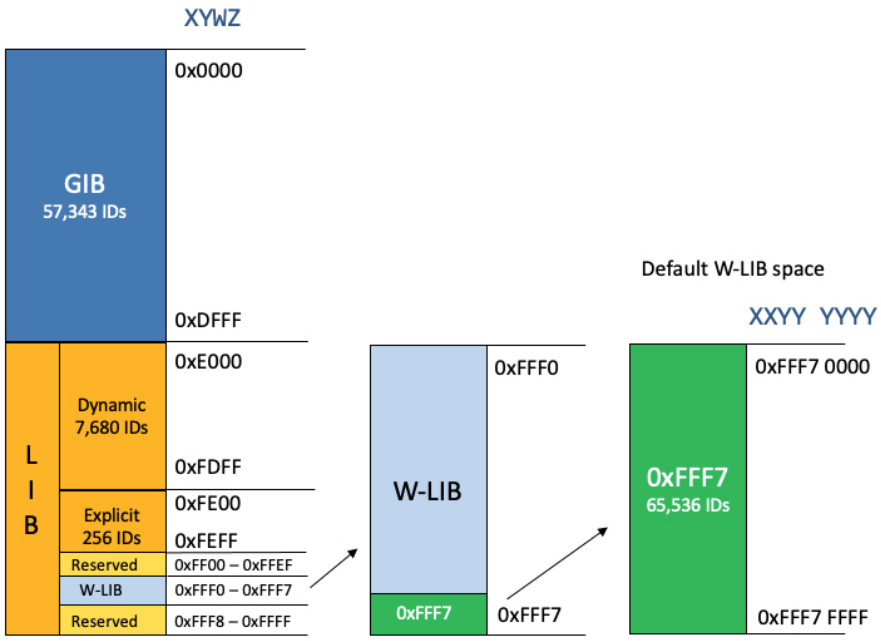
- GIB for user-assigned IDs of global segments (uNs)
- LIB for dynamically assigned IDs of local segments
  - uA end-point behavior
  - Service de-multiplexing end-point behaviors: End.DT, End.DX, End.DX2
- Explicit LIB for user-assigned IDs of local segments
  - Configurable explicit LIB range
  - Service de-multiplexing end-point behaviors: End.DT46
- Wide LIB (W-LIB)
  - Explicit W-LIB for user-assigned IDs of local segments
  - Configurable explicit W-LIB range
  - Service de-multiplexing end-point behaviors: End.DT46
- Remote W-LIB

- An SRv6 headend node is capable of receiving and installing remote SIDs with Wide (32-bit) functions
- Remote W-LIB is supported for Layer 3 (VPN/BGP global) and Layer 2 EVPN services (ELINE/ELAN)

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

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

Figure 2: GIB/LIB/W-LIB



SRv6 Endpoint Behaviors Associated with uSID

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

- uN—A short notation for the NEXT-CSID (Compressed SID) End behavior with a pseudocode of shift-and-lookup, and PSP/USD flavors

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

SRv6 uSID in Action - Example

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

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

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

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

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

- 32-bit SRv6 block = fcbb:bb01
- 16-bit SRv6 ID

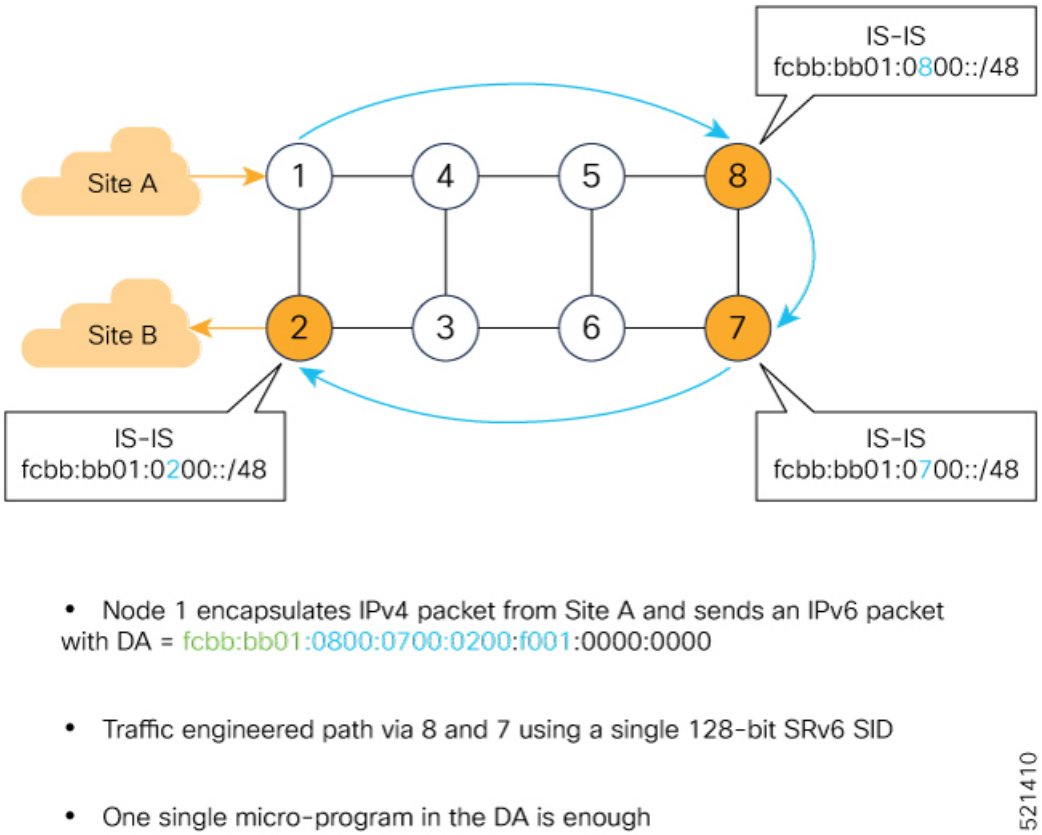
  For example:

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

The following IGP routes are advertised:

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

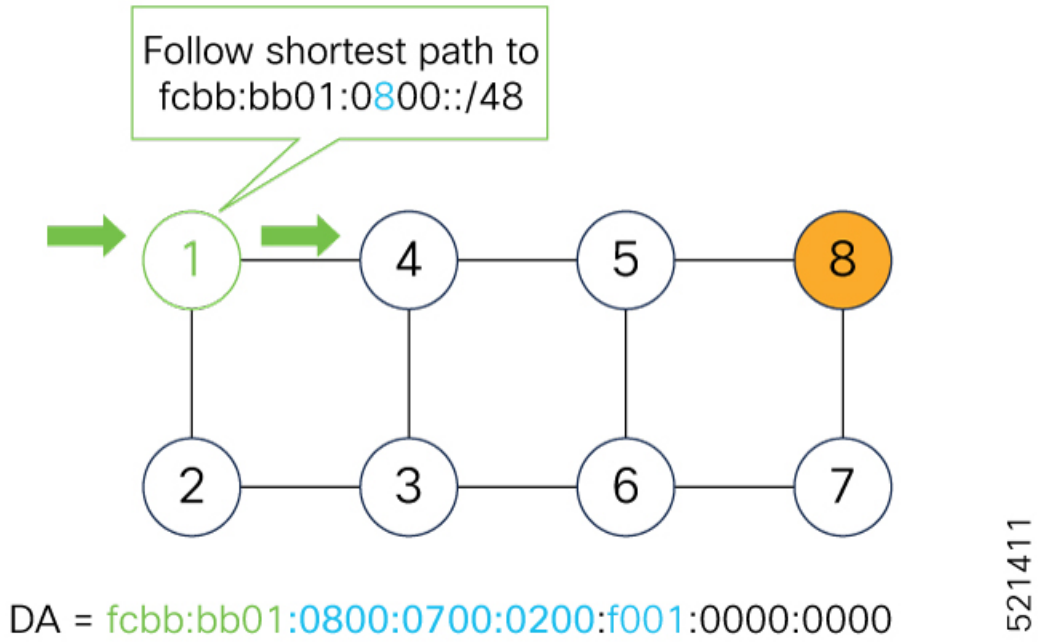
Figure 3: Integrated VPN and Traffic Engineering SRv6 uSID Use-case



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

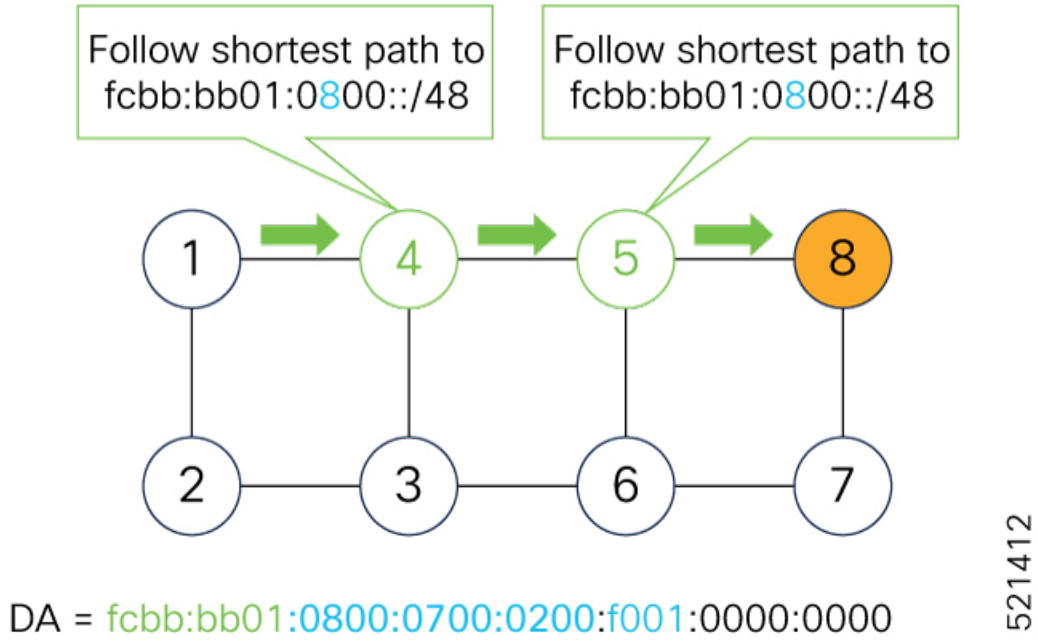
uSIDs (uNs) 0800, 0700, 0200 are used to realize the traffic engineering path to Node 2 with way points at Nodes 8 and 7. uSID f001 is the BGP-signalled instruction (uDT4) advertized by Node 2 for the VPNv4 service

Figure 4: Node 1: End.B6.Encaps Behavior



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

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

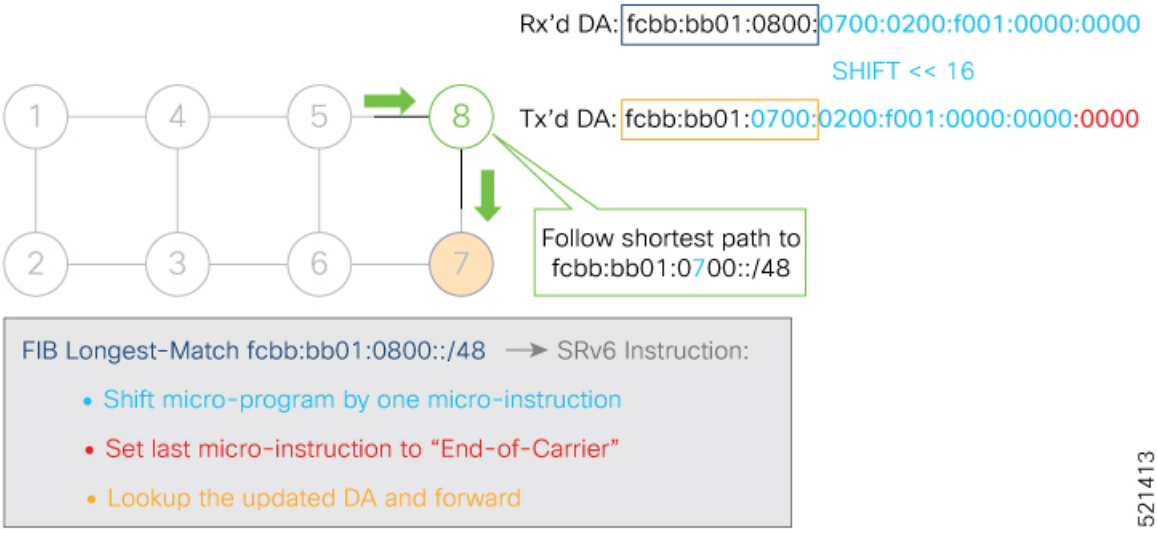


When Node 8 receives the packet, it performs SRv6 uN behavior (shift-and-lookup with PSP/USD). It removes its outer DA (0800) and advances the micro program to the next micro instruction by doing the following:

1. Pops its own uSID (0800)

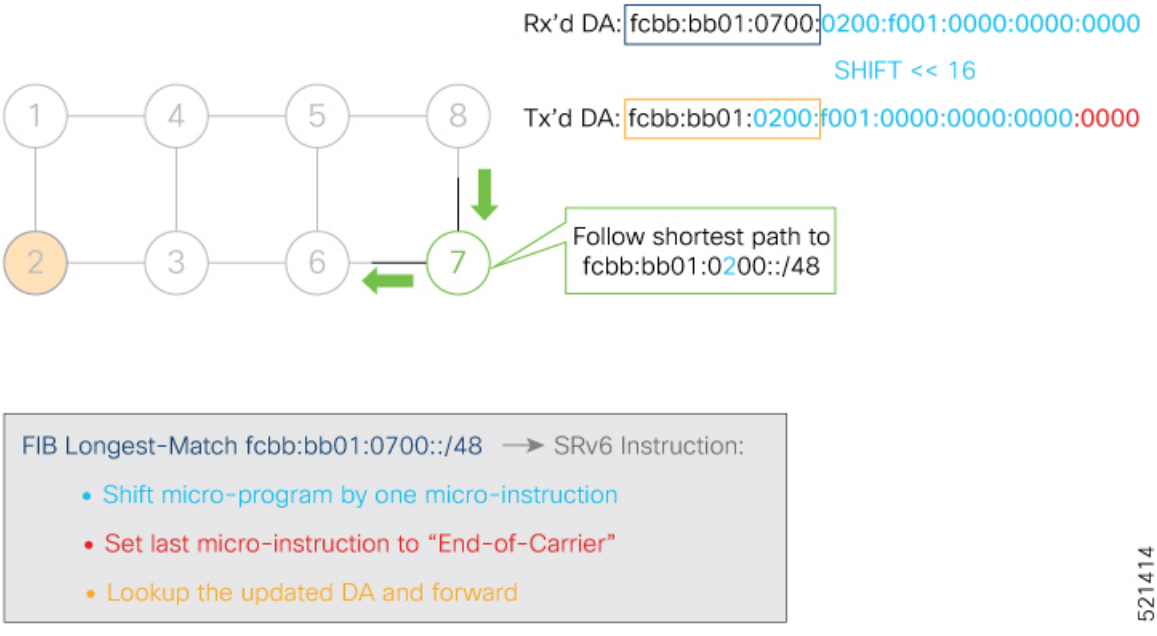
2. **Shifts** the remaining DA by 16-bits to the left
3. Fills the remaining bits with 0000 (End-of-Carrier)
4. Performs a **lookup** for the shortest path to the next DA (fcbb:bb01:**0700::/48**)
5. Forwards it using the new DA fcbb:bb01:**0700**:0200:f001:0000:0000:0000

Figure 6: Node 8: SRv6 uN Behavior (Shift and Forward)



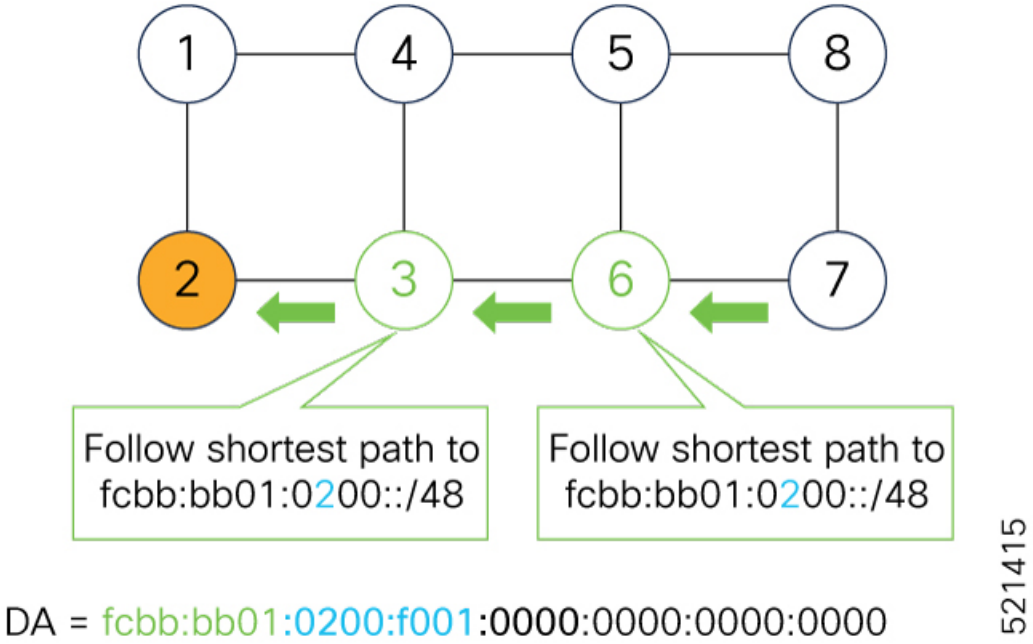
When Node 7 receives the packet, it performs the same SRv6 uN behavior (shift-and-lookup with PSP/USD), forwarding it using the new DA fcbb:bb01:**0200**:f001:0000:0000:0000:0000

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



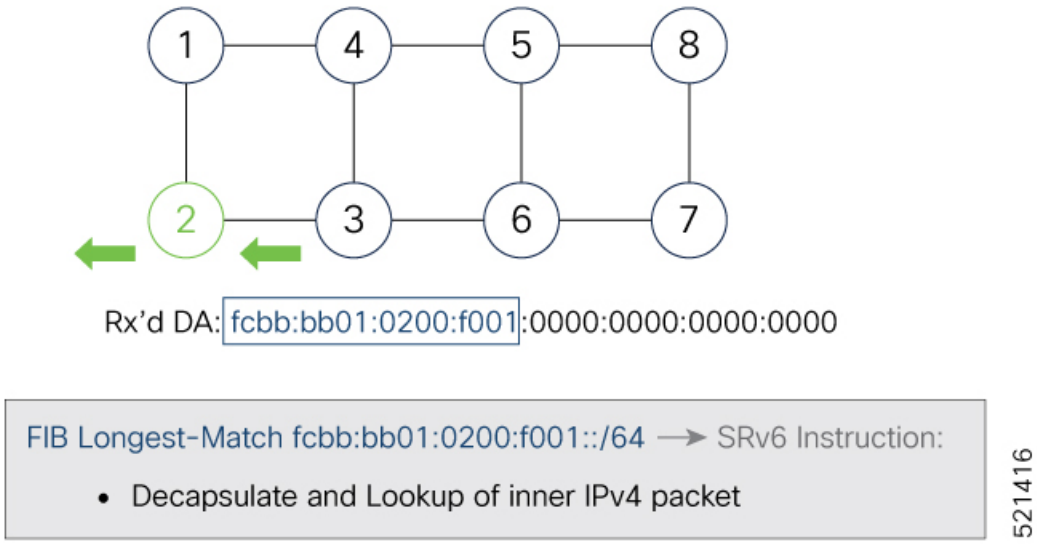
Nodes 6 and 3 simply forward the packet along the shortest path to Node 2, providing seamless deployment through classic IPv6 nodes.

Figure 8: Node 6 and Node 3: Classic IPv6 Nodes



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

Figure 9: Node 2: SRv6 uDT4 Behavior



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

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

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

Usage Guidelines and Limitations

General Guidelines and Limitations

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

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

- Cisco IOS XR supports up to 8 uSID locator prefixes.

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

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

  Up to 64 uSID blocks can be used across all uSID locators in the network.

- Cisco IOS XR Release 7.3.1 and later supports the following SRv6 uSID behaviors and variants:
  - uN with PSP/USD
  - uA with PSP/USD
  - uDT4
  - uDT6
  - uDT46

- SRv6 Underlay support includes:
  - IGP redistribution/leaking between levels
  - Prefix Summarization on ABR routers
  - IS-IS TI-LFA
  - Microloop Avoidance
  - Flex-algo

- SRv6 over GRE interface is not supported.
- SRv6 over BVI is supported only on physical and bundle interfaces, from Cisco IOS XR Release 7.5.1 onwards.

uSID Allocation Recommendation

We recommend that the uSID block allocation is made from the IPv6 Unique Local Address (ULA) range.

Note Allocation from the public Global Unicast Addresses (GUA) range is also supported.

- Use ULA /24 base from FC00::/8 space
  - FC*BB*:*BB*/24, with *B* indicating a nibble value picked by operator
- 256 uSID blocks possible from this allocation
  - In this release, 64 uSID blocks are supported
  - FC*BB*:*BBVV*/32, with *VV* two variable nibbles. The supported values for *VV* in Cisco IOS XR Release 7.3.1 are 0x00 to 0x3F.

For example:

- ULA /24 base = FC00:01/24
- uSID block space = 64 uSID blocks (from FC00:01**00**/32 to FC00:01**3F**/32)

Platform-Specific Guidelines and Limitations

This release supports SRv6 on the Cisco ASR 9000 series 3rd, 4th, and 5th generation line cards and the Cisco IOS XRv 9000.

This release supports SRv6 with ACL Chaining on Cisco ASR 9000 series 4th and 5th generation line cards.

Note SRv6 with ACL Chaining is supported for permit and deny ACL traffic options only.

On fourth and fifth generations of the Cisco ASR 9000 Series High-Density Ethernet line cards, enabling SRv6 and BGP Flowspec or local policy-based routing (PBR) is not supported.

This release supports the following SID Encap budget:

- P role:
  - Underlay H-Insert: 12 sids (2 carriers with 6 sids per carrier)
- PE role:
  - Underlay H-Insert: 12 sids (2 carriers with 6 sids per carrier)
  - Overlay H-Encaps: 3 sids (1 carrier with 3 sids per carrier)

Configuring SRv6

Enabling SRv6 involves the following high-level configuration steps:

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

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

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

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

(Optional) Configure Algorithm Associated with Locator

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

  For additional information about SRv6 Flexible Algorithm, see Configuring SRv6 Flexible Algorithm under IS-IS, on page 26.

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

(Optional) Configure Anycast Locator

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Anycast Locator	Release 7.3.1	An SRv6 Anycast locator is a type of locator that identifies a set of nodes (END SIDs). SRv6 Anycast Locators and their associated END SIDs may be provisioned at multiple places in a topology. One use case is to advertise Anycast END SIDs at exit points from an SRv6 network.

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

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

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

The following behaviors apply to Anycast Locator:

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

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

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

Example 1:

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

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

Example 2:

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

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

Example 3:

The following example shows how to configure Anycast locator.

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

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

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

(Optional) Customize SRv6 Encapsulation Parameters

This section describes the configurable SRv6 encapsulation parameters. These optional parameters include:

- **segment-routing srv6 encapsulation source-address** *ipv6-addr*—Source Address of outer encapsulating IPv6 header. The default source address for encapsulation is one of the loopback addresses.
- **segment-routing srv6 encapsulation hop-limit** {*count* | **propagate**}—The hop limit of outer-encapsulating IPv6 header. The range for *count* is from 1 to 255; the default value for hop-limit is 255. Use **propagate** to set the hop-limit value by propagation (from incoming packet/frame).
- **segment-routing srv6 encapsulation evpn next-header** *protocol-number*—The protocol number to use in the Next-header field of the IPv6 or SRH header. The range for *protocol-number* is from 59 to 252.
- **segment-routing srv6 encapsulation traffic-class** {*value* | **propagate**}—The traffic-class field settings on the IPv6 header. Specify the *value* (as 2 hexadecimal nibbles) for traffic class; valid values are from 0x0 to 0xff. Use **propagate** to set the traffic-class value by propagation (from incoming packet/frame).

(Optional) Customize SRv6 Logging for Locator Status Changes

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

(Optional) Customize SRv6 SID Parameters

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

Example 4:

The following example shows how to configure optional SRv6 parameters:

```
RP/0/RSP0/CPU0:Node1(config)# segment-routing srv6 encapsulation
RP/0/RSP0/CPU0:Node1(config-srv6-encap)# source-address 1::1
RP/0/RSP0/CPU0:Node1(config-srv6-encap)# hop-limit 60
RP/0/RSP0/CPU0:Node1(config-srv6-encap)# traffic-class propagate
RP/0/RSP0/CPU0:Node1(config-srv6-encap)# evpn next-header 65
RP/0/RSP0/CPU0:Node1(config-srv6-encap)# exit
RP/0/RSP0/CPU0:Node1(config-srv6)# logging locator status
RP/0/RSP0/CPU0:Node1(config-srv6)# sid holdtime 10

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

Verifying SRv6 Manager

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

```
Router# show segment-routing srv6 manager
Parameters:
```

```
  SRv6 Enabled: Yes
  SRv6 Operational Mode:
    Micro-segment:
      SID Base Block: 2001::/24
  Encapsulation:
    Source Address:
      Configured: ::
      Default: ::
    Hop-Limit: Default
    Traffic-class: Default
Summary:
  Number of Locators: 3 (3 operational)
  Number of SIDs: 3 (0 stale)
  Max SIDs: 64000
  OOR:
    Thresholds: Green 3200, Warning 1920
    Status: Resource Available
        History: (0 cleared, 0 warnings, 0 full)
    Block 2001::/32:
        Number of SIDs free: 7680
        Max SIDs: 7680
        Thresholds: Green 384, Warning 231
        Status: Resource Available
            History: (0 cleared, 0 warnings, 0 full)
Platform Capabilities:
  SRv6: Yes
  TILFA: Yes
  Microloop-Avoidance: Yes
  Endpoint behaviors:
    End (PSP)
    End.X (PSP)
    End.DX6
    End.DX4
    End.DT6
    End.DT4
    End.OP
    uN (PSP/USD)
    uA (PSP/USD)
    uDT6
    uDT4
    uDX2
    uB6 (Insert.Red)
  Headend behaviors:
    T
    H.Insert.Red
    H.Encaps.Red
  Security rules:
    SEC-1
    SEC-2
    SEC-3
  Counters:
    CNT-1
    CNT-3
  Signaled parameters:
    Max-SL          : 3
    Max-End-Pop-SRH : 3
    Max-H-Insert    : 3 sids
    Max-H-Encap     : 3 sids
    Max-End-D       : 4
  Configurable parameters (under srv6):
    Encapsulation:
      Source Address: Yes
      Hop-Limit     : value=Yes, propagate=No
      Traffic-class : value=Yes, propagate=Yes
```

```
  Max SIDs: 64000
  SID Holdtime: 3 mins
```

Verifying SRv6 Locator

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

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

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

Verifying SRv6 SIDs

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

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

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

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

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

Configuring SRv6 under IS-IS

Intermediate System-to-Intermediate System (IS-IS) protocol already supports segment routing with MPLS dataplane (SR-MPLS). This feature enables extensions in IS-IS to support Segment Routing with IPv6 data

plane (SRv6). The extensions include advertising the SRv6 capabilities of nodes and node and adjacency segments as SRv6 SIDs.

SRv6 IS-IS performs the following functionalities:

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

Usage Guidelines and Restrictions

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

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

Configuring SRv6 under IS-IS

To configure SRv6 IS-IS, use the **router isis** command. Enable SRv6 under the IS-IS IPv6 address-family and assign SRv6 locator(s) to it. Use the **level** {**1** | **2**} keywords to advertise the locator only in the specified IS-IS level.

Note If no level is specified, local locators will be advertised into all configured ISIS levels. Ensure that locators are included in the redistribution or propagation policy to prevent potential loops when redistributing between multiple instances or propagating between Level 2 and Level 1.

The following example shows how to configure SRv6 under IS-IS.

```
Router(config)# router isis core
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc1 level 1
Router(config-isis-srv6-loc)# exit
```

For more information about configuring IS-IS, refer to the "Implementing IS-IS" chapter in the *Routing Configuration Guide for Cisco ASR 9000*.

Configuring SRv6 Locator Metric and Tag

You can configure an SRv6 locator metric. Valid values are 0 (default) through 16777215.

You can configure an SRv6 locator tag. Valid values are 1 through 4294967295.

By default, the metric and tag are advertised for both levels. You can configure the specific level for these metrics.

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 myLoc
Router(config-isis-srv6-loc)# metric 450 level 1
Router(config-isis-srv6-loc)# tag 350 level 1
```

Running Configuration

```
router isis CORE
 address-family ipv6 unicast
  segment-routing srv6
   locator myLoc
    metric 450 level 1
    tag 350 level 1
   !
  !
 !
!
```

Configuring SRv6 Flexible Algorithm under IS-IS

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

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

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

Usage Guidelines and Restrictions

Observe the following usage guidelines and restrictions:

- You can configure up to 8 locators to support SRv6 Flexible Algorithm.
- The Flexible Algorithm locator prefix follows the same usage guidelines and restrictions of algo-0 locator prefixes. See Usage Guidelines and Limitations, on page 18.
- The Locator Algorithm value range is 128 to 255.

Configuring SRv6 Flexible Algorithm under IS-IS

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

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

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

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

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

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

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

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

```
Router(config)# router isis core
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# metric-type delay
Router(config-isis-flex-algo)# exit
Router(config-isis)# interface GigabitEthernet0/0/0/0
Router(config-isis-if)# address-family ipv6 unicast
```

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

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

Verification

```
Router# show segment-routing srv6 locator

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


Router# show isis flex-algo 128
```

```
IS-IS core Flex-Algo Database

Flex-Algo 128:

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


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


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

Configuring SRv6 Locator Prefix Summarization

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 IS-IS Prefix Summarization	Release 7.3.1	SRv6 leverages longest-prefix-match IP forwarding. This feature enables summarizing of locators at ABRs and ASBRs, which achieves massive-scale reachability.

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

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

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

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

Configuring TI-LFA with SRv6 IS-IS

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

TI-LFA provides link protection in topologies where other fast reroute techniques cannot provide protection. The goal of TI-LFA is to reduce the packet loss that results while routers converge after a topology change due to a link failure. TI-LFA leverages the post-convergence path which is planned to carry the traffic and ensures link and node protection within 50 milliseconds. TI-LFA with IS-IS SR-MPLS is already supported.

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

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

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- TI-LFA provides link protection by default. Additional tiebreaker configuration is required to enable node or SRLG protection.
- Usage guidelines for node and SRLG protection:
  - TI-LFA node protection functionality provides protection from node failures. The neighbor node is excluded during the post convergence backup path calculation.
  - Shared Risk Link Groups (SRLG) refer to situations in which links in a network share a common fiber (or a common physical attribute). These links have a shared risk: when one link fails, other links in the group might also fail. TI-LFA SRLG protection attempts to find the post-convergence backup path that excludes the SRLG of the protected link. All local links that share any SRLG with the protecting link are excluded.
  - When you enable link protection, you can also enable node protection, SRLG protection, or both, and specify a tiebreaker priority in case there are multiple LFAs.
  - Valid priority values are from 1 to 255. The lower the priority value, the higher the priority of the rule. Link protection always has a lower priority than node or SRLG protection.

Configuring SRv6 IS-IS TI-LFA

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

```
Router(config)# router isis core
Router(config-isis)# interface bundle-ether 1201
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# fast-reroute per-prefix
Router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Router(config-isis-if-af)# exit
Router(config-isis-if)# exit
Router(config-isis)# interface bundle-ether 1301
Router(config-isis-if)# address-family ipv6 unicast
Router(config-isis-if-af)# fast-reroute per-prefix
Router(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Router(config-isis-if-af)# fast-reroute per-prefix tiebreaker node-protecting index 100
Router(config-isis-if-af)# fast-reroute per-prefix tiebreaker srlg-disjoint index 200
```

```
Router(config-isis-if-af)# exit
```

Verification

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

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

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

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

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

Routing entry for cafe:0:2::2/128
  Known via "isis 1", distance 115, metric 20, type level-2
  Installed Feb 23 22:57:38.900 for 00:11:09
  Routing Descriptor Blocks
    fe80::1600:ff:feec:fe00, from cafe:0:2::2, via HundredGigE0/0/0/1, Backup (TI-LFA)
      Repair Node(s): cafe:0:4::4
      Route metric is 40
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:65              Path ref count:1
      NHID:0x20002(Ref:19)
      SRv6 Headend: H.Insert.Red [f3216], 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.Insert.Red SID-list {cafe:0:4::}
   via fe80::e00:ff:fe3a:c700/128, HundredGigE0/0/0/0, 6 dependencies, weight 0, class 0,
protected [flags 0x400]
    path-idx 1 bkup-idx 0 NHID 0x20001 [0x8f8420b0 0x0]
    next hop fe80::e00:ff:fe3a:c700/128

    Load distribution: 0 (refcount 2)

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

Configuring SRv6 IS-IS Microloop Avoidance

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 IS-IS Microloop Avoidance with Flexible Algorithm	Release 7.3.1	The SRv6 Microloop Avoidance feature detects and addresses microloops that (might) occur after a topology change.

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

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

The SRv6 Microloop Avoidance feature detects if microloops are possible following a topology change. If a node computes that a microloop could occur on the new topology, the node creates a loop-free SR-TE policy path to the destination using a list of segments. After the RIB update delay timer expires, the SR-TE policy is replaced with regular forwarding paths.

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

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

Configuring SRv6 IS-IS Microloop Avoidance

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

Note Complete the Configuring SRv6, on page 20 before performing these steps.

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

Configuring SRv6 IS-IS Microloop Avoidance with Flexible Algorithm

Microloop Avoidance 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 microloop avoidance path.

By default, Microloop Avoidance for SRv6 Flexible Algorithm uses the Microloop Avoidance configuration of Algo 0.

Use the **microloop avoidance disable** command to disable the microloop calculation on a per-algorithm basis:

```
Router(config)# router isis test-tilfa
Router(config-isis)# flex-algo 128
Router(config-isis-flex-algo)# microloop avoidance disable
```

Configuring Static SIDs

Manually allocated (static) SIDs are persistent over reloads and restarts, making them a reliable option to be used in explicit paths.

Configuring Static Adjacency SIDs

An adjacency SID (Adj-SID) is associated with an adjacency to a neighboring node. The adjacency SID steers the traffic to a specific adjacency. Adjacency SIDs have local significance and are only valid on the node that allocates them.

Adjacency SIDs that are dynamically allocated do not require any special configuration, however there are some limitations:

- A dynamically allocated Adj-SID value is not known until it has been allocated, and a controller will not know the Adj-SID value until the information is flooded by the IGP.

- Dynamically allocated Adj-SIDs are not persistent and can be reallocated after a reload or a process restart.
- Each link is allocated a unique Adj-SID, so the same Adj-SID cannot be shared by multiple links.

Manually allocated (static) Adj-SIDs are persistent over reloads and restarts, making them a reliable option to be used in explicit paths. They can be provisioned for multiple adjacencies to the same neighbor or to different neighbors.

You can specify that the Adj-SID is protected. If the Adj-SID is protected on the primary interface and a backup path is available, a backup path is installed. By default, static Adj-SIDs are not protected.

Usage Guidelines and Limitations

- Static Adj-SIDs are allocated from the explicit Local ID block (LIB). The default range for explicit SID allocation is 0xFE00 to 0xFEFF (256 IDs).
  - You can increase the size of the explicit LIB range by modifying the start of the range of IDs available in the explicit LIB range. See Configuring Explicit SRv6 uSID Allocation Start Range, on page 37 for more information.
- Static Adj-SIDs are allocated independent of IS-IS. If a static Adj-SID is successfully allocated, then the SID Manager notifies IS-IS. IS-IS then claims the allocated static Adj-SID and installs the Adj-SID in RIB or FIB.
- If you remove the static Adj-SID configuration, the SID Manager does not clear the allocated static Adj-SID; IS-IS will continue to use the allocated static Adj-SID. This is expected behavior. If you configure a different static Adj-SID, SID manager resolves the conflict and installs the new Adj-SID and notifies IS-IS.
- Static Adj-SID allocation does not follow the interface manager (IM) interface state. The static Adj-SID will be allocated even if the interface is down or doesn't exist in the system. However, the static Adj-SID is not installed in forwarding, since IS-IS didn't claim this Adj-SID.
- Static Adj-SID follows the locator state. Static Adj-SID allocation will fail if the locator is down or invalid.
- Static Adj-SID can be allocated as a protected Adj-SID.
- Static Adj-SID can be allocated to a specific algorithm.

Configuration

Use the **segment-routing srv6 static endpoint** command to configure static adjacency SID and SID allocation context.

Use the **sid** <*srv6-sid*> **behavior end-ua-psp-usd** command in srv6-static-endpoint config mode to configure the static Adj SID and endpoint behavior.

Use the **allocation-context nexthop** <*nhop-int*> [**algo-id** <*algo*> | **protected**] command in srv6-static-sid config mode to configure the next-hop interface. You can also configure the following options:

- **algo-id** *algo* — Specify a Flex Algo value for the static Adj-SID. Valid values for *algo* are 128 to 255; the default is 0 (SPF).
- **protected** — Specify if the static Adj-SID is protected

Example

```
Router(config)# segment-routing
Router(config-sr)# srv6
Router(config-srv6)# static
Router(config-srv6-static)# endpoint
Router(config-srv6-static-endpoint)# sid fccc:0:200:fe01:: behavior end-ua-psp-usd
Router(config-srv6-static-sid)# allocation-context nexthop TenGigE0/1/0/30/1
Router(config-srv6-static-sid)# exit
Router(config-srv6-static-endpoint)# sid fccc:0:200:fe88:: behavior end-ua-psp-usd
Router(config-srv6-static-sid)# allocation-context nexthop TenGigE0/1/0/0/3 protected
Router(config-srv6-static-sid)# exit
Router(config-srv6-static-endpoint)# sid fccc:3a:200:fe01:: behavior end-ua-psp-usd
Router(config-srv6-static-sid)# allocation-context nexthop TenGigE0/1/0/0/3 algo-id 132
protected
Router(config-srv6-static-sid)#
```

Running Configuration

```
segment-routing
 srv6
  static
   endpoint
    sid fccc:0:200:fe01:: behavior end-ua-psp-usd
     allocation-context nexthop TenGigE0/1/0/30/1
    !
    sid fccc:0:200:fe88:: behavior end-ua-psp-usd
     allocation-context nexthop TenGigE0/1/0/0/3 protected
    !
    sid fccc:3a:200:fe01:: behavior end-ua-psp-usd
     allocation-context nexthop TenGigE0/1/0/0/3 algo-id 132 protected
    !
```

Configuring Explicit End.DT46 SRv6 SIDs

Table 6: Feature History Table

Feature Name	Release	Description
Support for End.DT46 SRv6 Endpoint Behavior	Release 7.8.1	This feature adds support for the “Endpoint with decapsulation and specific IP table lookup” SRv6 end-point behavior (End.DT46). The End.DT46 behavior is used for dual-stack L3VPNs. This behavior is equivalent to the single per-VRF VPN label (for IPv4 and IPv6) in MPLS.
Support for Explicit End.DT46 SRv6 SIDs	Release 7.8.1	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.

Explicit End.DT46 SRv6 SIDs are persistent over reloads and restarts.

Multiple explicit uDT46 IDs allocated from the LIB or W-LIB range can be created under the same SRv6 locator. Each ID is uniquely associated to a VRF.

Note See GIB and LIB – IOS-XR Implementation, on page 11 for information about the LIB and W-LIB

Example: Explicit uDT46 (LIB)

- Locator: fcbb:bb**00**:**11**::/48
- Explicit functions: fe0a (VRF-A), fe0b (VRF-B), fe0c (VRF-C)
  - fcbb:bb00:**11**:fe0a::/64 — Explicit 16-bit DT46 function from LIB for VRF-A
  - fcbb:bb00:**11**:fe0b::/64 — Explicit 16-bit DT46 function from LIB for VRF-B
  - fcbb:bb00:**11**:fe0c::/64 — Explicit 16-bit DT46 function from LIB for VRF-C

Example: Explicit uDT46 (W-LIB)

- Locator: fcbb:bb**00**:**11**::/48
- Explicit functions: fff7:d (VRF-D), fff7:e (VRF-E), fff7:f (VRF-F)
  - fcbb:bb00:**11**:fff7:d::/80 — Explicit 32-bit DT46 function from W-LIB for VRF-D
  - fcbb:bb00:**11**:fff7:e::/80 — Explicit 32-bit DT46 function from W-LIB for VRF-E
  - fcbb:bb00:**11**:fff7:f::/80 — Explicit 32-bit DT46 function from W-LIB for VRF-F

An explicit uDT46 ID allocated from the LIB or W-LIB range can be associated to the same VRF under multiple SRv6 locators.

This association is useful when a look-up under a given VPN table is desired for a node with multiple locators (for example, unicast and Anycast locators).

The locators can be from the same ID block or different ID blocks:

- When the locators are from the same block, the manual uDT46 IDs for a given VRF must have the same value across locators.
- When the locators are from different blocks, the manual uDT46 IDs for a given VRF could be either the same value or different values.

We recommend using the same function ID across locators since it allows for simpler identification to the associated VRF table.

Example 1: Explicit uDT46 SIDs (LIB) with a common function ID under 2 locators of the same block-id

- Locator 1: fcbb:bb**00**:**10**::/48 (unicast locator)
- Locator 2: fcbb:bb**00**:**100**::/48 (Anycast locator)
- Explicit function: fe0a
- VRF lookup: VRF-A

```
fcbb:bb00:10:fe0a::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-A
fcbb:bb00:100:fe0a::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-A
```

Example 2: Explicit uDT46 SIDs (LIB) with a common function ID under 2 locators of different block-id

- Locator 1: fcbb:bb**00**:**11**::/48 (unicast locator – algo0)
- Locator 2: fcbb:bb**01**:**11**::/48 (unicast locator – algo128)
- Explicit function: fe0b
- VRF lookup: VRF-B

```
fcbb:bb00:11:fe0b::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-B
fcbb:bb01:11:fe0b::/64 segment-routing srv6 endpoint behavior uDT46 vrf VRF-B
```

Example 3: Explicit uDT46 SIDs (W-LIB) with a common function ID under 2 locators of the same block-id

- Locator 1: fcbb:bb**00**:**10**::/48 (unicast locator)
- Locator 2: fcbb:bb**00**:**100**::/48 (Anycast locator)
- Explicit function: fff7:d
- VRF lookup: VRF-D

```
fcbb:bb00:11:fff7:d::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-D
fcbb:bb00:100:fff7:d::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-D
```

Example 4: Explicit uDT46 SIDs (W-LIB) with a common function ID under 2 locators of different block-id

- Locator 1: fcbb:bb**00**:**11**::/48 (unicast locator – algo0)
- Locator 2: fcbb:bb**01**:**11**::/48 (unicast locator – algo128)
- Explicit function: fff7:e
- VRF lookup: VRF-E

```
fcbb:bb00:11:fff7:e::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-E
fcbb:bb01:11:fff7:e::/80 segment-routing srv6 endpoint behavior uDT46 vrf VRF-E
```

Configuration

To configure explicit uDT46 IDs allocated from the LIB or W-LIB range, use the **router static address-family ipv6 unicast** *prefix*/ *mask* **segment-routing srv6 endpoint behavior uDT46 vrf** *vrf-name* command:

```
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
```

Running Config

```
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
```

Configuring Explicit SRv6 uSID Allocation Start Range

You can increase the range of IDs available in the explicit LIB and explicit W-LIB by modifying their starting values.

See SRv6 uSID Allocation Within a uSID Block, on page 9 for information about the default range of IDs in the explicit LIB and W-LIB.

To modify the start value for the explicit LIB, use the following command:

- **segment-routing srv6 formats format usid-f3216 usid local-id-block explicit start** *lib-start-value*, where *lib-start-value* is from 0xE064 to 0xFEFF



Note When you increase the size of the explicit LIB range, you effectively decrease the number of available IDs in the dynamic LIB range. For example, if you configure the explicit LIB starting value to 0xE064, the dynamic LIB range is 0xE000 to 0xE063 (100 IDs).

To modify the start value for the explicit W-LIB, use the following command:

- **segment-routing srv6 formats format usid-f3216 usid wide-local-id-block explicit start** *wlib -start-value*, where *wlib-start-value* is from 0xFFF0 to 0xFFF7

Example

Use the **show segment-routing srv6 manager** command to display the default start values of the explicit LIB and W-LIB:

```
RP/0/RP0/CPU0:ios# show segment-routing srv6 manager
Fri Sep  9 18:30:06.503 UTC
Parameters:
  SRv6 Enabled: No
  SRv6 Operational Mode: None
  Encapsulation:
    Source Address:
      Configured: ::
      Default: ::
    Hop-Limit: Default
    Traffic-class: Default
  SID Formats:
    f3216 <32B/16NFA> (2)
      uSID LIB Range:
        LIB Start   : 0xe000
        ELIB Start  : 0xfe00
      uSID WLIB Range:
        EWLIB Start : 0xfff7
```

```
. . .
```

The following example shows how to modify the start of the range for explicit LIB and W-LIB:

```
RP/0/RP0/CPU0:ios(config)# segment-routing
RP/0/RP0/CPU0:ios(config-sr)# srv6
RP/0/RP0/CPU0:ios(config-srv6)# formats
RP/0/RP0/CPU0:ios(config-srv6-fmts)# format usid-f3216
RP/0/RP0/CPU0:ios(config-srv6-fmt)# usid local-id-block explicit start 0xE064
RP/0/RP0/CPU0:ios(config-srv6-fmt)# usid wide-local-id-block explicit start 0xFFF0
```

Running Config

```
segment-routing
 srv6
  formats
   format usid-f3216
    usid local-id-block explicit start 0xe064
    usid wide-local-id-block explicit start 0xfff0
   !
  !
 !
!
```

Use the **show segment-routing srv6 manager** command to display the configured starting values of the explicit LIB and W-LIB:

```
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)

. . .
```

Configuring SRv6 BGP-Based 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 ASR 9000 Series Routers* BGP Configuration Guide.

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 and Restrictions

- The following SRv6 BGP-based services are supported:
  - IPv4 Layer-3 VPNs
  - IPv6 Layer-3 VPNs
  - IPv4 BGP global
  - IPv6 BGP global
- uDT4 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 the following inheritance rules:

1. Use the locator as defined under the service.

   If not defined under the specific service, then:

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

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

3. Use the locator as defined globally under BGP.

Enabling SRv6 Globally under BGP

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

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

Assigning SRv6 Locator Globally under BGP

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

This example shows how to assign a locator:

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

For more information on how to configure an SRv6 locator, see Configuring SRv6, on page 20.

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

SRv6 Services: IPv4 L3VPN

Table 7: Feature History Table

Feature Name	Release	Description
Per-Prefix SRv6 Locator Assignment	Release 7.5.1	This feature provides the ability to assign a specific SRv6 locator for a given prefix or a set of prefixes (IPv4/IPv6 GRT, IPv4/IPv6 VPN). The egress PE advertises the prefix with the specified locator. This allows for per-prefix steering into desired transport behaviors, such as Flex Algo.
Support for iBGP as PE-CE protocol	Release 7.5.1	This feature introduces support for iBGP as PE-CE protocol.
BGP Route Leaking	Release 7.5.1	This feature adds support for importing routes from default-VRF to non-default VRF and routes from non-default VRF to default VRF.

Table 8: Feature History Table

Feature Name	Release	Description
Dual-Stack L3VPN Services (IPv4, IPv6) (SRv6 Micro-SID)	Release 7.3.2	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.

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

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF, or per-prefix.
- Per-VRF allocation mode is supported (uDT4 behavior)
- Dual-Stack L3VPN Services (IPv4, IPv6) are supported
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP (iBGP, eBGP), OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.
- Per-CE allocation mode is not supported (uDX4 behavior)

Configuring SRv6 based IPv4 L3VPN

To enable SRv6-based L3VPN, you need to enable SRv6 under BGP, specify the locator, and configure the SID allocation mode. The assignment of the locator can be done in different places under the **router bgp** configuration. See SRv6 Locator Inheritance Rules, on page 39.

Use Case 1: Assigning SRv6 Locator Globally

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

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

```
Node1(config-bgp-vrf-af)# commit
```

Running Config

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

Use Case 2: Assigning SRv6 Locator for All VRFs

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

- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6 alloc mode** {**per-vrf**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT4 behavior) be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6 locator** *WORD*: Specify the locator

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

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

Running Config

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

Use Case 3: Assigning SRv6 Locator for a specific VRF

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

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6 alloc mode** { **per-vrf**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT4 behavior) be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6 locator** *WORD*: Specify the locator

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

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

Running Config

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

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.

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 mode is per-VRF.
- 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.

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

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

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (10.1.1.0/24) 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)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
Node1(config)#
```

To specify per-prefix allocation mode for a specific VRF under IPv4 address family, use the following command:

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv4 unicast segment-routing srv6 alloc mode route-policy** *policy_name*

This example shows how to configure per-prefix allocation mode for a specific VRF (vrf_cust1) under IPv4 address family

```
Node1(config)# router bgp 100
Node1(config-bgp)# vrf vrf_cust1
Node1(config-bgp-vrf)# address-family ipv4 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# alloc mode route-policy set_per_prefix_locator_rpl
```

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (10.1.1.0/24) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (2.2.2.0/24) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3.3.3.0/24) then
    set srv6-alloc-mode per-vrf
  else
    drop
  endif
end-policy
!
router bgp 100
 vrf vrf_cust1
  address-family ipv6 unicast
   segment-routing srv6
    alloc mode route-policy set_per_prefix_locator_rpl
   !
  !
 !
!
```

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

```
Node1# show bgp vpnv4 unicast local-sids
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 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: 10.1.1.1:0 (default for vrf vrf_cust1)
*> 10.1.1.0/24        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
*> 10.1.1.5/32        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 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 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: 10.1.1.1:0 (default for vrf vrf_cust1)
*> 10.1.1.0/24        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
*> 10.1.1.5/32        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: 10.1.1.1:0
VRF ID: 0x60000004
BGP router identifier 10.1.1.1, 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: 10.1.1.1:0 (default for vrf vrf_cust1)
*> 10.1.1.0/24        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
*> 10.1.1.5/32        NO SRv6 Sid                              -            -
*> 3.3.3.3/32         NO SRv6 Sid                              -            -
*>i8.8.8.8/32         NO SRv6 Sid                              -            -
```

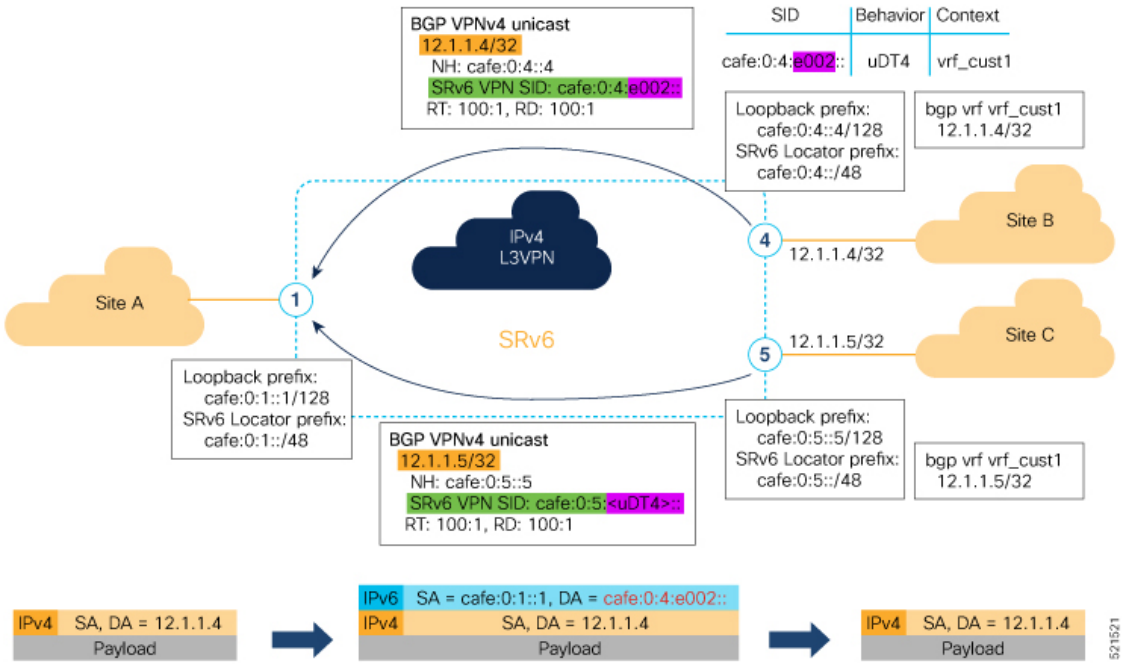
```
Node1# show bgp vrf vrf_cust1 received-sids
BGP VRF vrf_cust1, state: Active
BGP Route Distinguisher: 10.1.1.1:0
VRF ID: 0x60000004
BGP router identifier 10.1.1.1, local AS number 1
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000013   RD version: 37
BGP main routing table version 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: 10.1.1.1:0 (default for vrf vrf_cust1)
*> 10.1.1.0/24        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
*> 10.1.1.5/32        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::
```

Verification

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



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

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

```
Node1# show segment-routing srv6 sid

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

SID                         Behavior          Context                           Owner
           State  RW
--------------------------  ----------------  ------------------------------
------------------  -----  --
cafe:0:1::                  uN (PSP/USD)      'default':1                       sidmgr
           InUse  Y
```

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

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

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

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

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

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

```
Node1# show bgp vpnv4 unicast summary

BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 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 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 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 (10.1.1.4)
      Received Label 0xe00400
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 22
      Extended community: RT:1:1 RT:100:1
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:63, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv4 Unicast, Source VRF: vrf_cust1, Source Route Distinguisher: 100:1
```

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

```
Node1# show bgp vrf vrf_cust1 ipv4 unicast

BGP VRF vrf_cust1, state: Active
BGP Route Distinguisher: 100:1
VRF ID: 0x60000002
BGP router identifier 10.1.1.1, 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 (10.1.1.4)
      Received Label 0xe00400
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 22
      Extended community: RT:1:1 RT:100:1
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:63, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv4 Unicast, Source VRF: vrf_cust1, Source Route Distinguisher: 100:1
```

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

```
Node1# show route vrf vrf_cust1

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

Gateway of last resort is not set

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


Node1# show route vrf vrf_cust1 12.4.4.4/32

Routing entry for 12.4.4.4/32
  Known via "bgp 100", distance 200, metric 0, type internal
```

```
  Installed Feb 23 22:57:56.746 for 00:43:12
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
  No advertising protos.


Node1# show route vrf vrf_cust1 12.4.4.4/32 detail

Routing entry for 12.4.4.4/32
  Known via "bgp 100", distance 200, metric 0, type internal
  Installed Feb 23 22:57:56.746 for 00:43:37
  Routing Descriptor Blocks
    cafe:0:4::4, from cafe:0:4::4
      Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Source RD attributes: 0x0000:100:1
      NHID:0x0(Ref:0)
      SRv6 Headend: H.Encaps.Red [f3216], SID-list {cafe:0:4:e004::}
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_REMOTE
  Download Priority 3, Download Version 3
  No advertising protos.
```

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

```
Node1# show cef vrf vrf_cust1

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


Node1# show cef vrf vrf_cust1 12.4.4.4/32

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

```
Node1# show cef vrf vrf_cust1 12.4.4.4/32 detail

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

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

   via cafe:0:4::/128, 3 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x78e2da14 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop cafe:0:4::/128 via cafe:0:4::/48
    SRv6 H.Encaps.Red SID-list {cafe:0:4:e004::}

    Load distribution: 0 1 (refcount 1)

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

SRv6 Services: IPv6 L3VPN

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: IPv6 L3VPN	Release 7.3.1	With this feature, the egress PE can signal an SRv6 Service SID with the BGP overlay service route. The ingress PE encapsulates the IPv4/IPv6 payload in an outer IPv6 header where the destination address is the SRv6 Service SID provided by the egress PE. BGP messages between PEs carry SRv6 Service SIDs as a means to interconnect PEs and form VPNs.

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

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF, or per-prefix.
- Per-VRF allocation mode is supported (uDT6 behavior)
- Dual-Stack L3VPN Services (IPv4, IPv6) are supported
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.

- BGP (iBGP, eBGP), OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.
- Per-CE allocation mode is not supported (uDX6 behavior)

Configuring SRv6-based IPv6 L3VPN

To enable SRv6-based L3VPN, you need to enable SRv6 under BGP, specify the locator, and configure the SID allocation mode. The assignment of the locator can be done in different places under the **router bgp** configuration. See SRv6 Locator Inheritance Rules, on page 39.

Use Case 1: Assigning SRv6 Locator Globally

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

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

Running Configuration

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

Use Case 2: Assigning SRv6 Locator for All VRFs

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

- **router bgp** *as-number* **address-family vpnv6 unicast vrf all segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family vpnv6 unicast vrf all segment-routing srv6 alloc mode** {**per-vrf**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT6 behavior) be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **address-family vpnv6 unicast vrf all segment-routing srv6 locator** *WORD*: Specify the locator

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

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

Running Configuration

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

Use Case 3: Assigning SRv6 Locator for a specific VRF

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

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6 alloc mode** { **per-vrf**}: Specify the SID behavior (allocation mode)
  - Use the **per-vrf** keyword to specify that the same service SID (uDT6 behavior) be used for all the routes advertised from a unique VRF.
- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6 locator** *WORD*: Specify the locator

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

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

Running Configuration

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

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.

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

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide Cisco ASR 9000 Series Routers*.

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

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (3001::1:1:1:1/128) then
Node1(config-rpl-if)# set srv6-alloc-mode per-vrf locator locator1
Node1(config-rpl-if)# elseif destination in (3001::2:2:2:2/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf locator locator2
Node1(config-rpl-elseif)# elseif destination in (3001::3:3:3:3/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf
Node1(config-rpl-elseif)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
```

To specify per-prefix allocation mode for a specific VRF under IPv6 Address Family, use the following command:

- **router bgp** *as-number* **vrf** *WORD* **address-family ipv6 unicast segment-routing srv6 alloc mode route-policy** *policy_name*

This example shows how to specify per-prefix allocation mode for a specific VRF (vrf_cust6) under the IPv6 address family:

```
Node1(config)# router bgp 100
Node1(config-bgp)# vrf vrf_cust6
Node1(config-bgp-vrf)# address-family ipv6 unicast
Node1(config-bgp-vrf-af)# segment-routing srv6
Node1(config-bgp-vrf-af-srv6)# alloc mode route-policy set_per_prefix_locator_rpl
```

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (3001::1:1:1:1/128) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (3001::2:2:2:2/128) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3001::3:3:3:3/128) then
    set srv6-alloc-mode per-vrf
  else
    drop
  endif
end-policy
!
router bgp 100
 vrf vrf_cust6
  address-family ipv6 unicast
   segment-routing srv6
```

```
    alloc mode route-policy set_per_prefix_locator_rpl
   !
  !
 !
!
```

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

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

BGP router identifier 10.1.1.1, 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::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 10.1.1.1, 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::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: 10.1.1.1:0
VRF ID: 0x60000004
BGP router identifier 10.1.1.1, 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

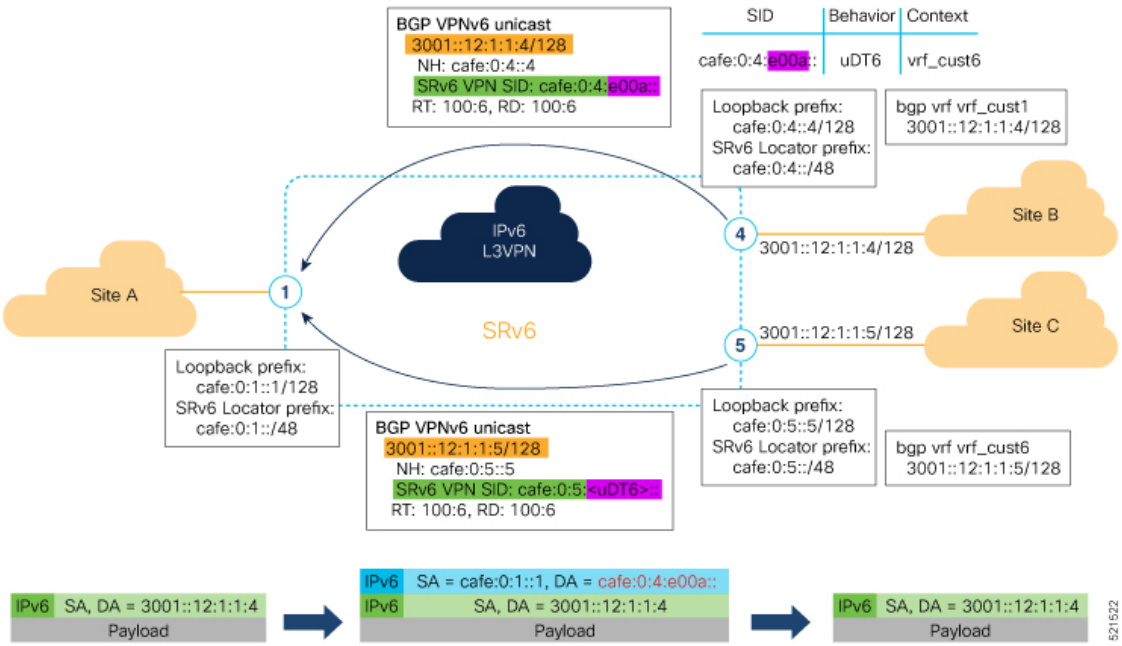
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Local Sid                                  Alloc mode   Locator
Route Distinguisher: 8:8
*>i3008::8:8:8:8/128  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::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 10.1.1.1, 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::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::
```

Verification

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



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

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

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

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

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

```
          InUse  Y
```

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

```
Node1# show bgp vpnv6 unicast summary
Fri Jan 29 19:33:01.177 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 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 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 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 (10.1.1.4)
```

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

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

```
Node1# show bgp vrf vrf_cust6 ipv6 unicast
Fri Jan 29 19:42:05.675 UTC
BGP VRF vrf_cust6, state: Active
BGP Route Distinguisher: 100:6
VRF ID: 0x60000007
BGP router identifier 10.1.1.1, 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 (10.1.1.4)
      Received Label 0xe00a00
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate, imported
      Received Path ID 0, Local Path ID 1, version 17
      Extended community: RT:100:6
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4::, Behavior:62, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv6 Unicast, Source VRF: vrf_cust6, Source Route Distinguisher: 100:6
```

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

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

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

Gateway of last resort is not set

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

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

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

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

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

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

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

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

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

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

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

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

    Load distribution: 0 1 (refcount 1)

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

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

SRv6 Services: IPv4 BGP Global

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

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv4 unicast address family
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- Dual-Stack L3 Services (IPv4 BGP global, IPv6 BGP global) are supported.

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

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

- **router bgp** *as-number* **address-family ipv4 unicast segment-routing srv6**: Enable SRv6
- **router bgp** *as-number* **address-family ipv4 unicast segment-routing srv6 alloc mode** {**per-vrf** | **route-policy** *policy_name*}: Specify the SID behavior (allocation mode).
  - **per-vrf**: Specifies that the same label is be used for all the routes advertised from a unique VRF.
  - **route-policy** *policy_name*: Uses a route policy to determine the SID allocation mode and locator (if provided) for given prefix.
- **router bgp** *as-number* **address-family ipv4 unicast segment-routing srv6 locator** *WORD*: Specify the locator
- **router bgp** *as-number* {**af-group** *WORD*| **neighbor-group** *WORD* | **neighbor** *ipv6-addr*} **address-family ipv4 unicast encapsulation-type srv6**: Specify the encapuslation type for SRv6.
  - Use **af-group** *WORD* to apply the SRv6 encapsulation type to the address family group for BGP neighbors.
  - Use **neighbor-group** *WORD*to apply the SRv6 encapsulation type to the neighbor group for BGP neighbors.
  - Use **neighbor** *ipv6-addr* to apply the SRv6 encapsulation type to the specific BGP neighbor.

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

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

```
Node1(config)# router bgp 1
Node1(config-bgp)# bgp router-id 10.1.0.1
Node1(config-bgp)# address-family ipv4 unicast
Node1(config-bgp-af)# segment-routing srv6
```

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

Running Configuration

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

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

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

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

- Supported SID allocation mode is per-VRF.
- For per-VRF allocation mode, you can also specify the SRv6 locator.
  - If an SRv6 locator is specified in the route policy, BGP will use that to allocate per-VRF SID. If the specified locator is invalid, the SID will not be allocated.

- If an SRv6 locator is not specified in the route policy, the default locator is used to allocate the SID. If the default locator is not configured in BGP, then the SID will not be allocated.

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

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

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (10.1.1.0/24) 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)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
Node1(config)#
```

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

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

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (10.1.1.0/24) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (2.2.2.0/24) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3.3.3.0/24) then
    set srv6-alloc-mode per-vrf
  else
    drop
  endif
end-policy
!
router bgp 100
 address-family ipv4 unicast
  segment-routing srv6
   alloc mode route-policy set_per_prefix_locator_rpl
  !
 !
!
```

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

```
Node1# show bgp ipv4 unicast local-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Local Sid                                Alloc mode   Locator
*> 10.1.1.0/24        fc00:8:1:41::                            per-vrf      locator2
*> 2.2.2.0/24         fc00:0:1:41::                            per-vrf      locator1
```

```
*> 3.3.3.0/24         fc00:9:1:42::                            per-vrf      locator4
*> 10.1.1.5/32        NO SRv6 Sid                              -            -
* i8.8.8.8/32         NO SRv6 Sid                              -            -
```

```
Node1# show bgp ipv4 unicast received-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop                          Received Sid
*> 10.1.1.0/24        66.2.2.2                          NO SRv6 Sid
*> 2.2.2.0/24         66.2.2.2                          NO SRv6 Sid
*> 3.3.3.0/24         66.2.2.2                          NO SRv6 Sid
*> 10.1.1.5/32        66.2.2.2                          NO SRv6 Sid
* i8.8.8.8/32         77.1.1.2                          fc00:0:2:41::
```

SRv6 Services: IPv6 BGP Global

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Services: BGP Global IPv6	Release 7.3.1	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 as a means to interconnect PEs.

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

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv6 unicast address family
- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.
- BGP, OSPF, Static are supported as PE-CE protocol.
- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.
- Dual-Stack L3 Services (IPv4 BGP global, IPv6 BGP global) are supported.

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

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

- **router bgp** *as-number* **address-family ipv6 unicast segment-routing srv6**: Enable SRv6

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

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

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

```
Node1(config)# router bgp 100
Node1(config-bgp)# bgp router-id 10.1.1.1
Node1(config-bgp)# segment-routing srv6
Node1(config-bgp-gbl-srv6)# locator Node1
Node1(config-bgp-gbl-srv6)# exit
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# locator Node1
Node1(config-bgp-af-srv6)# alloc mode per-vrf
Node1(config-bgp-af-srv6)# exit
Node1(config-bgp-af)# exit
Node1(config-bgp)# neighbor 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 10.1.1.1
 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
```

Use Case 2: BGP Global IPv6 over SRv6 with Per-Prefix SID Allocation

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

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

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

For more information on configuring routing policies, refer to the "Implementing Routing Policy" chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

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

```
Node1(config)# route-policy set_per_prefix_locator_rpl
Node1(config-rpl)# if destination in (3001::1:1:1:1/128) then
Node1(config-rpl-if)# set srv6-alloc-mode per-vrf locator locator1
Node1(config-rpl-if)# elseif destination in (3001::2:2:2:2/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf locator locator2
Node1(config-rpl-elseif)# elseif destination in (3001::3:3:3:3/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-vrf
Node1(config-rpl-elseif)# elseif destination in (3001::4:4:4:4/128) then
Node1(config-rpl-elseif)# set srv6-alloc-mode per-ce
Node1(config-rpl-elseif)# else
Node1(config-rpl-else)# drop
Node1(config-rpl-else)# endif
Node1(config-rpl)# end-policy
```

The following example shows how to configure BGP global IPv6 over SRv6 with a route policy to determine the SID allocation mode for given prefix.

```
Node1(config)# router bgp 100
Node1(config-bgp)# address-family ipv6 unicast
Node1(config-bgp-af)# segment-routing srv6
Node1(config-bgp-af-srv6)# alloc mode route-policy set_per_prefix_locator_rpl
```

Running Configuration

```
route-policy set_per_prefix_locator_rpl
  if destination in (3001::1:1:1:1/128) then
    set srv6-alloc-mode per-vrf locator locator1
  elseif destination in (3001::2:2:2:2/128) then
    set srv6-alloc-mode per-vrf locator locator2
  elseif destination in (3001::3:3:3:3/128) then
    set srv6-alloc-mode per-vrf
  else
    drop
  endif
end-policy
!
router bgp 100
 address-family ipv6 unicast
  segment-routing srv6
   alloc mode route-policy set_per_prefix_locator_rpl
  !
 !
```

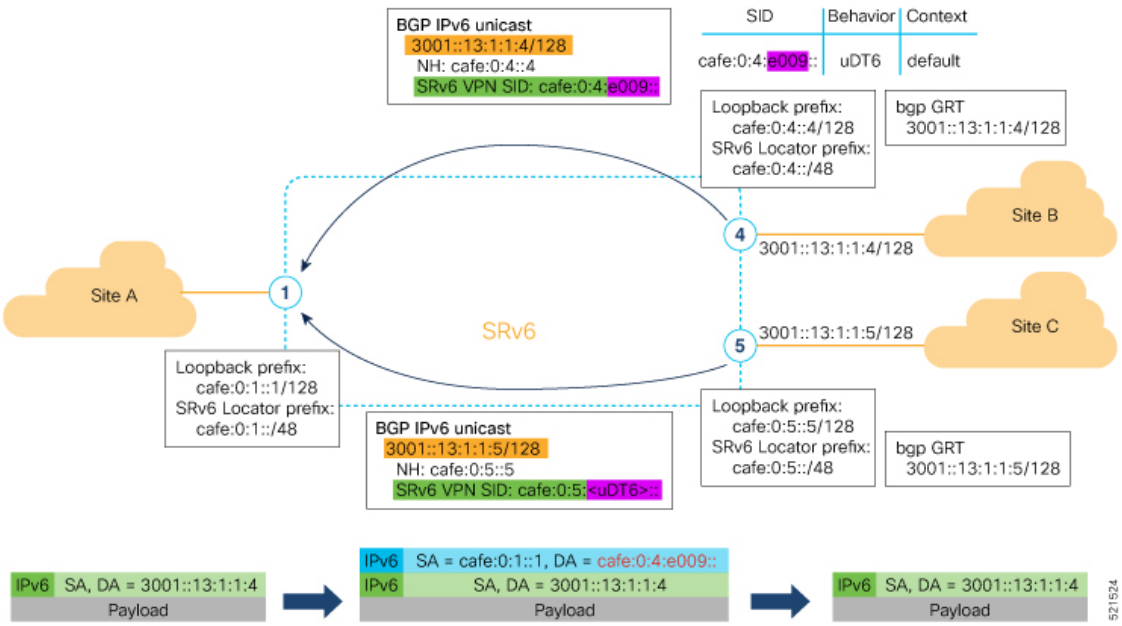
Verify that the local and received SIDs have been correctly allocated under BGP IPv6 address family:

```
Node1# show bgp ipv6 unicast local-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Local Sid                                Alloc mode   Locator
*> 3001::1:1:1:1/128  fc00:8:1:41::                            per-vrf      locator2
*> 3001::2:2:2:2/128  fc00:0:1:41::                            per-vrf      locator1
*> 3001::3:3:3:3/128  fc00:9:1:42::                            per-vrf      locator4
*> 3001::5:5:5:5/128  NO SRv6 Sid                               -            -
* i3008::8:8:8:8/128  NO SRv6 Sid                               -            -


Node1# show bgp ipv6 unicast received-sids
…
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop                            Received Sid
*> 3001::1:1:1:1/128  66.2.2.2                            NO SRv6 Sid
*> 3001::2:2:2:2/128  66.2.2.2                            NO SRv6 Sid
*> 3001::3:3:3:3/128  66.2.2.2                            NO SRv6 Sid
*> 3001::5:5:5:5/128  66.2.2.2                            NO SRv6 Sid
* i3008::8:8:8:8/128  77.1.1.2                            fc00:0:2:41::
```

Verification

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



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

```
Node1# show bgp ipv6 unicast summary
Fri Jan 29 19:48:23.255 UTC
BGP router identifier 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 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 10.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0800000   RD version: 4
BGP main routing table version 4
BGP NSR Initial initsync version 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 (10.1.1.4)
      Origin IGP, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 3
      PSID-Type:L3, SubTLV Count:1
       SubTLV:
        T:1(Sid information), Sid:cafe:0:4:e009::, Behavior:62, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
```

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

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

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

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

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

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

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

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

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

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

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

    Load distribution: 0 1 (refcount 1)

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

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

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
BGP Signaling for co-existence of IP routes	Release 24.3.1	SRv6 with BGP supports the coexistence of IP routes with or without SRv6 SID over an SRv6-enabled core network. This support enables integrating SRv6 capabilities into existing network infrastructures without replacing IP routing completely. This feature enables flexibility and scalability, transition to new technologies, and enhanced network efficiency, making it easier to migrate from MPLS to SRV6. The feature introduces these changes: CLI: • **encapsulation-type srv6 relax-sid**

Need for BGP Signaling Over SRv6 core

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

To differentiate between the SRv6 core and non-SRv6 core sides, an **encapsulation-type SRv6** was introduced under the IPv6 BGP peer for the IPv4 unicast address-family. When the "encapsulation-type srv6" is enabled, routes without an SRv6-SID are not sent to the neighbor sessions during update generation. For more information, see Configuring SRv6 BGP-Based Services, on page 38 and https://datatracker.ietf.org/doc/rfc9252/.

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

Co-existence of IP routes with or without SRv6 SID

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

Benefits

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

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

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

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

Configure BGP Signaling over SRv6 Core

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

Follow these steps to configure BGP signaling over SRv6 Core.

Procedure

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

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

Example:

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

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

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

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

Example:

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

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

Example:

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

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

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

Example:

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

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

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

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

Example:

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

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


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

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

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

Example:

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

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


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

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

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

Usage Guidelines and Restrictions

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

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

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

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

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

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

Configuration Example

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

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

Running Configuration

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

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

Verification

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

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

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

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


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

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



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

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

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

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

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

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

SRv6 Services: IPv4 L3VPN Active-Active Redundancy

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

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

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

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

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

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

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

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

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

Note EVPN does not support untagged interfaces.

SRv6 Services: EVPN VPWS — All-Active Multi-Homing

Table 12: Feature History Table

Feature Name	Release	Description
SRv6 Services: EVPN VPWS — All-Active Multi-Homing (SRv6 Micro SID)	Release 7.3.2	This feature provides an ELINE (P2P) service with all-active multihoming capability over an SRv6 network. All-Active Multi-Homing enables an operator to connect a customer edge (CE) device to two or more provider edge (PE) devices to provide load balancing and redundant connectivity. With All-Active Multi-Homing, all the PEs can forward traffic to and from the multi-homed device.

EVPN VPWS All-Active Multi-Homing over SRv6 provides an ELINE (P2P) service with all-active multihoming capability over an SRv6 network.

All-Active Multi-Homing enables an operator to connect a customer edge (CE) device to two or more provider edge (PE) devices to provide load balancing and redundant connectivity. With All-Active Multi-Homing, all the PEs can forward traffic to and from the multi-homed device.

Note For information about EVPN VPWS, refer to the "EVPN Virtual Private Wire Service (VPWS)" chapter in the *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*.

Configuring EVPN VPWS over SRv6

Note Complete the steps in Configuring SRv6, on page 20 before performing these steps.

An SRv6 Locator for an EVPN VPWS service can be configured at 3 different levels independently:

- global_locator is the default locator for all EVPN-VPWS services
- evi_locator is applied to all EVPN-VPWS services for the specific EVI
- evi_service_locator is applied to an individual EVI service

When locators are configured at different levels at the same time, the following priority is implemented:

1. evi_service_locator
2. evi_locator
3. global_locator

This example shows how to configure an EVPN VPWS over SRv6 using a global locator for EVPN:

```
evpn
 segment-routing srv6
  locator sample_global_loc

l2vpn
 xconnect group sample_xcg
  p2p sample-vpws-12001-2002
   interface Bundle-Ether12001.2002
   neighbor evpn evi 12001 service 2002 segment-routing srv6
```

This example shows how to configure EVPN VPWS over SRv6 using specific EVI locator:

```
evpn
 evi 11001 segment-routing srv6
  locator sample_evi_loc

l2vpn
 xconnect group sample_xcg
  p2p sample-vpws-11001-2002
   interface Bundle-Ether11001.2002
   neighbor evpn evi 11001 service 2002 segment-routing srv6
```

This example shows how to configure an EVPN VPWS over SRv6 using a locator for an individual EVI service:

```
l2vpn
 xconnect group sample_xcg
  p2p sample-vpws-11001-2001
   interface Bundle-Ether11001.2001
```

```
   neighbor evpn evi 11001 service 2001 segment-routing srv6
    locator sample_evi_service_loc
```

Verification

```
Router# show segment-routing srv6 locator

Name                  ID       Algo  Prefix                    Status   Flags
--------------------  -------  ----  ------------------------  -------  --------
sample_evi_loc        1        128   2001:0:8::/48             Up       U
sample_global_loc     2        0     2001:0:1::/48             Up       U


Router# show segment-routing srv6 sid

*** Locator: 'sample_evi_loc' ***

SID                         Behavior          Context                          Owner
          State  RW
--------------------------  ----------------  -------------------------------
------------------  -----  --
2001:0:8::                  uN (PSP/USD)      'default':8                      sidmgr
          InUse  Y
2001:0:8:e000::             uDX2              11001:2002                       l2vpn_srv6
          InUse  Y
2001:0:8:e002::             uA (PSP/USD)      [BE11, Link-Local]:128           isis-20
          InUse  Y
2001:0:8:e004::             uA (PSP/USD)      [BE60, Link-Local]:128           isis-20
          InUse  Y
2001:0:8:e006::             uA (PSP/USD)      [BE30, Link-Local]:128           isis-20
          InUse  Y

*** Locator: 'sample_global_loc' ***

2001:0:1::                  uN (PSP/USD)      'default':1                      sidmgr
          InUse  Y
2001:0:1:e001::             uDX2              12001:2002                       l2vpn_srv6
          InUse  Y
2001:0:1:e003::             uA (PSP/USD)      [BE11, Link-Local]:0             isis-20
          InUse  Y
2001:0:1:e005::             uA (PSP/USD)      [BE60, Link-Local]:0             isis-20
          InUse  Y
2001:0:1:e007::             uA (PSP/USD)      [BE30, Link-Local]:0             isis-20
          InUse  Y


Router# show evpn segment-routing srv6 detail

Configured default locator: sample_global_loc
EVIs with unknown locator config: 0
VPWS with unknown locator config: 0

Locator name         Prefix                    OOR      Service count    SID count
------------         ------                    ---      -------------    ---------
sample_evi_loc       2001:0:8::/48             False    1                1
   Configured on EVIs <evi>: 11001
sample_global_loc    2001:0:1::/48             False    1                1
   Default locator


Router# show l2vpn xconnect group sample_xcg detail
Thu Sep  2 14:39:22.575 UTC
```

```
Group sample_xcg, XC sample-vpws-11001-2002, state is up; Interworking none
  AC: Bundle-Ether11001.2002, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [2002, 2002]
    MTU 1504; XC ID 0xc0002ee8; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor ::ffff:10.0.0.1, PW ID: evi 11001, ac-id 2002, state is up ( established
)
    XC ID 0xa0001f47
    Encapsulation SRv6
    Encap type Ethernet
    Ignore MTU mismatch: Enabled
    Transmit MTU zero: Disabled
    Reachability: Up

      SRv6              Local                       Remote
      ----------------  --------------------------- --------------------------
      uDX2              2001:0:8:e000::             2001:0:3:e000::
      AC ID             2002                        2002
      MTU               1518                        1518
      Locator           sample_evi_loc              N/A
      Locator Resolved  Yes                         N/A
      SRv6 Headend      H.Encaps.L2.Red             N/A
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0

Group sample_xcg, XC sample-vpws-12001-2002, state is up; Interworking none
  AC: Bundle-Ether12001.2002, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [2002, 2002]
    MTU 1504; XC ID 0xc0002eea; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor ::ffff:10.0.0.2, PW ID: evi 12001, ac-id 2002, state is up ( established
)
    XC ID 0xa0001f49
    Encapsulation SRv6
    Encap type Ethernet
    Ignore MTU mismatch: Enabled
    Transmit MTU zero: Disabled
    Reachability: Up

      SRv6              Local                       Remote
      ----------------  --------------------------- --------------------------
      uDX2              2001:0:1:e001::             2001:0:2:e001::
      AC ID             2002                        2002
      MTU               1518                        1518
      Locator           sample_global_loc           N/A
      Locator Resolved  Yes                         N/A
      SRv6 Headend      H.Encaps.L2.Red             N/A
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

SRv6-Services: EVPN ELAN Layer 2 Gateway With Automated Steering To Flexible Algorithm Paths

Table 13: Feature History Table

Feature Name	Release Information	Feature Description
SRv6-Services: EVPN ELAN Layer 2 Gateway With Automated Steering To Flexible Algorithm Paths	Release 7.5.2	This feature builds upon EVPN BGP signaling to provide Emulated Local Area Network (ELAN) multipoint-to-multipoint Ethernet services over an SRv6-based network. You can enable automated steering of EVPN ELAN traffic into the path associated with a best-effort or Flex- Algorithm locator. This feature combines the benefits of EVPN ELAN service and SRv6 Micro-SIDs. For this feature, the **segment-routing srv6** option was added to the **evi** command: evi (bridge-domain)

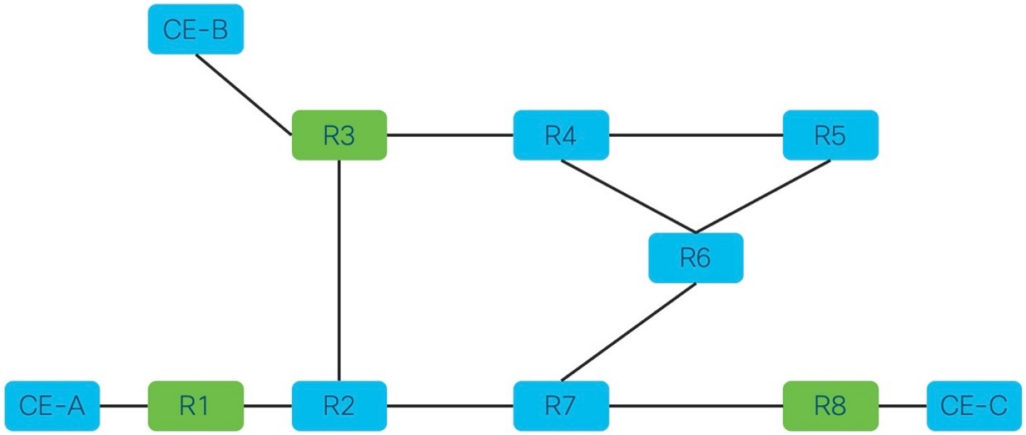
You can transport EVPN ELAN bridged unicast and broadcast, unknown unicast, and multicast (BUM) traffic over an SRv6 network in the Micro-SID format. Relevant SRv6 headend and endpoint definitions are noted below:

- **H.Encaps.L2.Red**: This headend router operation involves reduced encapsulation of Layer 2 or Ether frames using an SRv6 Policy.
- **uDT2U**: This endpoint router operation involves traffic decapsulation and unicast MAC L2 table lookup. This is used for the EVPN bridging unicast traffic use case.
- **uDT2M**: This endpoint router operation involves traffic decapsulation and L2 table flooding. This is used for the EVPN bridging BUM traffic with ESI filtering use case.



Note For more information on SRv6 headend and endpoint behaviors, refer to Segment Routing over IPv6 Overview.

The following topology is used to explain this feature.



Topology pointers:

- Customer edge (CE) devices send traffic between each other over the SRv6 network. The CE devices are CE-A, CE-B, and CE-C.
- The SRv6 network devices transport customer traffic, and they are R1, R2 .. till R8.
- The provider edge (PE) devices, R1, R3 and R8, are displayed in green. The SRv6 EVPN configurations must be enabled on the PE devices since they participate in the EVPN EVI.

This is a high-level overview of the traffic flow from CE-A to CE-C:

1. CE-A sets the source and destination addresses of the L2 frame and sends it to the connected PE device, R1.
2. R1 looks up the destination MAC address in the frame. Based on its forwarding table, R1 performs an H.Encaps.L2.Red operation and adds the destination DT2U SRv6 SID (say, fccc:ccc1:a1:e000::) to the packet.
3. From R1, traffic is sent over the SRv6 network to destination PE device R8.
4. When R8 receives the traffic, it performs the uDT2U function - It decapsulates the packet, performs a destination MAC address lookup in its forwarding table, and sends the frame through the local interface to CE-C.

Guidelines and Limitations

- For transporting BUM traffic, the BGP Route Reflector device should have an IOS XR release version 7.5.2 or later.
- For devices with Cisco ASR 9000 High Density 100GE Ethernet line cards, enable the **hw-module l3 feature ler disable** and **hw-module l3 feature lsr disable** commands.
- For faster convergence when transporting BUM traffic on Cisco ASR 9000 devices with multiple line cards, enable the **flood mode resilience-optimized** command in the L2VPN bridge-domain configuration mode.

Configure SRv6 EVPN Bridging

Enable the following configurations on the PE routers R1, R3 and R8 since they participate in the EVPN EVI.

Note Complete the steps in Segment Routing over IPv6 Overview before performing these steps.

Associate SRv6 with EVPN

```
Router# configure terminal
Router(config)# evpn
```

Enable SRv6 under the EVPN mode and associate a global locator (**sample**, in this case) with EVPN.

```
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator sample
Router(config-evpn-srv6)# exit
```

Associate an EVI-specific locator (**sample_evi_loc**) with EVI 1.

```
Router(config-evpn)# evi 1 segment-routing srv6
Router(config-evpn-instance)# locator sample_evi_loc
Router(config-evpn-instance)# commit
```

Associate SRv6 with L2VPN

Associate the sub-interface to the bridge domain:

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Hu0/0/0/0.1
Router(config-l2vpn-bg-bd-ac)# exit
```

Enable the **evi 1 segment-routing srv6** command under L2VPN bridge domain **bd1**.

```
Router(config-l2vpn-bg-bd)# evi 1 segment-routing srv6
Router(config-l2vpn-bg-bd-evi-srv6)# commit
```

When transporting BUM traffic on a device with multiple line cards, enable the **flood mode resilience-optimized** command for faster network convergence.

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg11001
Router(config-l2vpn-bg)# bridge-domain elan-11001-2001
Router(config-l2vpn-bg-bd)# flood mode resilience-optimized
Router(config-l2vpn-bg-bd)# commit
```

Verification

In this sample output, SRv6 EVPN ELAN traffic unicast and multicast SID information is displayed.

```
Router# show evpn evi vpn-id 1 detail

VPN-ID     Encap      Bridge Domain                 Type
---------- ---------- ----------------------------- -----------------
1          SRv6       bd1                           EVPN

..
   Stitching: Regular
   Unicast SID:  fccc:ccc1:a1:e000::
   Multicast SID: fccc:ccc1:a1:e001::
..
```

In this sample output, EVI 1 details, including the corresponding SID and EVPN MAC address details are displayed.

```
Router# show evpn evi vpn-id 1 mac

VPN-ID    Encap   MAC            IP        Nexthop      Label    SID
                  address        address
-----     ------ --------       --------   -------      -----         ---

1         SRv6    0010.3000.01d0  ::       Hu0/0/0/0.1    0     fccc:ccc1:a1:e000::
```

In this sample output, for the specified EVI and EVPN MAC address, SRv6 EVPN ELAN traffic details are displayed.

```
Router# show evpn evi vpn-id 1 mac 0010.3000.01d0 detail

VPN-ID Encap       MAC            IP       Nexthop      Label         SID
                   address        address
------ -----       ------         ------  -------       -----         ---
1     SRv6   ee03.0500.0130  ::     192.168.0.3  IMP-NULL fccc:ccc1:a3:e000::

   Ethernet Tag                           : 0
   Multi-paths Resolved                   : True
   Multi-paths Internal label             : None
   Local Static                           : No
   Remote Static                          : Yes
   Local Ethernet Segment                 : N/A
   Remote Ethernet Segment                : 0100.0205.acce.5500.0500
   Local Sequence Number                  : N/A
   Remote Sequence Number                 : 0
   Local Encapsulation                    : N/A
   Remote Encapsulation                   : SRv6
   Local E-Tree                           : Root
   Remote E-Tree                          : Root
   Remote matching E-Tree RT              : No
   Local AC-ID                            : 0x0
   Remote AC-ID                           : 0x13
```

In this sample output, for the specified EVI, multicast SID details are displayed.

```
Router# show evpn evi vpn-id 1 inclusive-multicast detail

VPN-ID        Encap           EtherTag             Originating IP
---           ----            -------              -------------
1            SRv6       0                   192.168.0.1
..
          TEPid: 0xffffffff
  PMSI Type: 6
  Nexthop: ::
  SR-TE Info: N/A
     SID: fccc:ccc1:a1:e001::
  Source: Local
  E-Tree: Root
..
```

In this sample output, for the specified MAC address, bridge domain information is displayed.

```
Router# show l2route evpn mac all | i ee03.0500.0130

Topo ID   Mac Address             Producer     Next Hop(s)
------    -----------             -------      -------------
1         ee03.0500.0130    L2VPN        ::ffff:10.0.0.10/IID/V6, N/A
```

In this sample output, SRv6 network locator and corresponding SID information are displayed.

uDT2U and **uDT2M** refer to SRv6 network endpoint operations. **uDT2U** indicates SRv6 traffic decapsulation, wherein EVPN bridged unicast traffic is forwarded out of the SR network. **uDT2M** indicates SRv6 traffic decapsulation, wherein EVPN bridged multicast traffic is forwarded out of the SR network.

```
Router# show segment-routing srv6 sid

*** Locator: 'sample_evi_loc' ***

SID                     Behavior      Context         Owner         State   RW
---                     --------      -------         -----         -----   --
fccc:ccc1:a1::          uN(PSP/USD)  'default':161   sidmgr         InUse   Y
fccc:ccc1:a1:e000::     uDT2U   7:0             l2vpn_srv6    InUse   Y
fccc:ccc1:a1:e001::     uDT2M   7:0             l2vpn_srv6    InUse   Y
```

In this sample output, CEF information is displayed, including SRv6 network endpoint details. **uDT2U** is an SRv6 network endpoint operation wherein SRv6 traffic is decapsulated and EVPN bridged unicast traffic is forwarded out of the SR network.

```
Router# show cef ipv6 fccc:ccc1:a1:e000:: detail

fccc:ccc1:a1:e000::, version 14, SRv6 Endpoint uDT2U, internal 0x1000001
0x0 (ptr 0x8ba26050) [1], 0x400 (0x8bbf7b58), 0x0 (0x92396138)

 Prefix Len 64, traffic index 0, precedence n/a, priority 0
  gateway array (0x8ba33e90) reference count 4, flags 0x0, source rib (7), 0 backups
                [5 type 3 flags 0x8401 (0x8baf8ca8) ext 0x0 (0x0)]

  LW-LDI[type=3, refc=1, ptr=0x8bbf7b58, sh-ldi=0x8baf8ca8]
  gateway array update type-time 1 Sep  8 11:46:51.242

 LDI Update time Sep  8 11:46:51.303
 LW-LDI-TS Sep  8 11:46:51.380
   via ::/128, 0 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0x8afdf120 0x0]
    next hop ::/128
    XConnect ID: 0x80000003
    Bridge ID: 0x1
    Shg ID: 0x1

    Load distribution: 0 (refcount 5)

    Hash  OK  Interface                 Address
    0     Y   recursive                 Lookup in table
```

In the following examples, SRv6 EVPN ELAN traffic-related IID information is displayed.

```
Router# show evpn internal-id vpn-id 3001 detail

VPN-ID  Encap  Ethernet Segment Id         EtherTag     Internal ID
------  ------ -------------------------- ----------   -----------
1       SRv6   0001.0001.0001.1501.0015    0         ::ffff:10.0.0.4

   Summary pathlist:

 0x05000002 (P) 192.168.0.3 fccc:ccc1:a3:e000::
 0x05000002 (P) 192.168.0.3 fccc:ccc1:a4:e000::

Router# show cef vrf **iid ipv6 ::ffff:10.0.0.4

::ffff:10.0.0.4/128, version 39, SRv6 Headend, IID (EVPN-MH), internal 0x1000001 0x0 (ptr
0x8ba21798) [3], 0x0 (0x0), 0x0 (0x923967b0)
 Updated Sep  8 18:01:06.495
 Prefix Len 128, traffic index 0, precedence n/a, priority 0
  gateway array (0x8ba36018) reference count 1, flags 0x2010, source rib (7), 0 backups
```

```
              [1 type 3 flags 0x48441 (0x8baf9a28) ext 0x0 (0x0)]
 LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
 gateway array update type-time 1 Sep  8 18:01:06.495
LDI Update time Sep  8 18:01:06.495

 Level 1 - Load distribution: 0
 [0] via fccc:ccc1:a3::/128, recursive

  via fccc:ccc1:a3:e000::/128, 10 dependencies, recursive [flags 0x0]
   path-idx 0 NHID 0x0 [0x8ba24e78 0x0]
   next hop VRF - 'default', table - 0xe0800000
   next hop fccc:ccc1:a3::/128 via fccc:ccc1:a3::/48
   SRv6 H.Encaps.L2.Red SID-list { fccc:ccc1:a3:e000::}

   Load distribution: 0 (refcount 1)

   Hash  OK  Interface        Address
   0     Y   Hu0/0/0/0        remote

  via fccc:ccc1:a4::/128, 10 dependencies, recursive [flags 0x100]
   path-idx 0 NHID 0x0 [0x8ba24e78 0x0]
   next hop VRF - 'default', table - 0xe0800000
   next hop fccc:ccc1:a4::/128 via fccc:ccc1:a4::/48
   SRv6 H.Encaps.L2.Red SID-list {fccc:ccc1:a4:e000::}
```

In this sample output, SRv6 EVPN ELAN traffic-related IID information is displayed.

```
Router# show rib ipv6 iid

IID         Prefix                Context             Owner           State      RW
-------     ------                -------             -------         -----      --
0xa000001 ::ffff:10.0.0.1
[EVPN-ELAN:evi=7:esi=8300.fccc.ccc1.00a4.0000:nh=fccc:ccc1:a4:::eth_tag=0:type=0:encap=255:opaque=0]

                                                        l2vpn_iid       InUse      Y

0xa000002 ::ffff:10.0.0.2
[EVPN-ELAN:evi=8:esi=8300.fccc.ccc1.00a4.0000:nh=fccc:ccc1:a4:::eth_tag=0:type=0:encap=255:opaque=0]

                                                        l2vpn_iid       InUse      Y

0xa000003 ::ffff:10.0.0.3
[EVPN-ELAN:evi=9:esi=8300.fccc.ccc1.00a4.0000:nh=fccc:ccc1:a4:::eth_tag=0:type=0:encap=255:opaque=0]

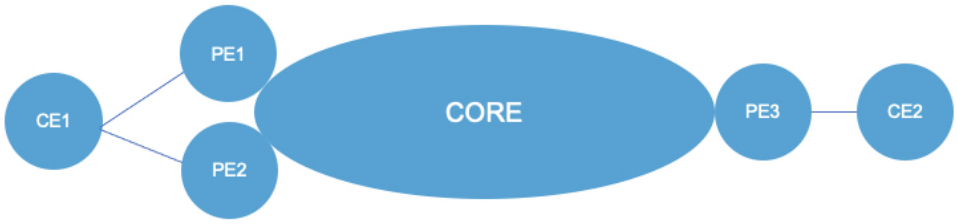
                                                        l2vpn_iid       InUse      Y
```

SRv6 ESI Filtering

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 ESI Filtering	Release 7.11.1	Split Horizon Group (SHG) labels and Ethernet Segment Identifier (ESI) filtering functionalities exist on MPLS underlay networks. This feature introduces ESI filtering functionality to SRv6 underlay networks, using the End.DT2M SRv6 endpoint behavior. This behavior uses the "Arg.FE2" argument for SRv6, which is similar to the SHG label for MPLS. This feature allows nodes to identify BUM traffic based on the advertised ESI and prevent a loop by avoiding re-broadcasting the same traffic back towards the access node. This functionality is enabled by default.

Consider the below network topology where CE1 is attached to PE1 and PE2.



When broadcast, unknown unicast, and multicast (BUM) traffic is received by PE1 from CE1, PE1 floods this traffic on the core network. This results in a copy of the BUM traffic being sent to PE2. PE2 in turn floods it to the core and access network, and the BUM traffic would be received at CE1. This is undesirable as we are flooding the same traffic to the source from which it originated, thus causing a loop.

SRv6 ESI Filtering

A unique 16-bit ID is generated by all nodes in an Ethernet Segment and is transmitted to all nodes in the same EVPN instance via ES/EAD Route Type 1 NLRI (see "EVPN Route Types" for description of Route Type 1: Ethernet Auto-Discovery (AD) Route). This unique label functions in a similar way as the Split Horizon group label (SHG label). Ethernet Segment Identifier (ESI) filtering is the mechanism that controls which nodes to re-broadcast BUM traffic.

Note For more information, see "Split Horizon Groups" in the *L2VPN and Ethernet Services Configuration Guide*.

When PE1 receives BUM traffic from an attachment circuit (AC), it floods the traffic to all nodes in its core network, where PE2 is also member node. The traffic flooded to PE2 is sent with its previously advertised ID. When PE2 decodes the packet and examines the label, it recognizes this to be the label that it had generated. In this mechanism, PE2 will not flood the BUM traffic back towards the access. PE2 has successfully prevented a loop by avoiding re-broadcasting the same traffic back towards the source (CE1).

This feature introduces ESI filtering to SRv6 underlay networks, using the End.DT2M SRv6 endpoint behavior. This behavior uses the "Arg.FE2" argument, as defined in IETF RFC 8986 SRv6 Network Programming and IETF draft SRv6 Argument Signaling for BGP Services. This argument provides a local mapping to ESI for split-horizon filtering of the received traffic to exclude a specific outgoing L2 interface (OIF), or a set of OIFs, from L2 table T flooding. The allocation of the argument values is local to the SR Segment Endpoint Node instantiating this behavior, and the signaling of the argument to other nodes for the EVPN functionality occurs via the control plane.

Usage Guidelines and Limitations

This feature is supported on third, fourth, and fifth generation of ASR 9000 Series High-Density Ethernet line cards. Refer to the *Cisco ASR 9000 Series Aggregation Services Router Ethernet Line Card Installation Guide* for details on ASR 9000 Series Line Cards.

Verification

The output of the **show evpn ethernet-segment interface** *interface-name* now displays both the SHG label information for MPLS EVPN instances (EVIs) and the Arg.FE2 information for SRv6 EVIs:

```
Router# show evpn ethernet-segment interface Bundle-Ether 1

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- -----------------
0011.1111.1111.1111.1114 BE1                               192.168.0.2
                                                           192.168.0.3
. . .

Peering timer : 3 sec [not running]
 Recovery timer : 30 sec [not running]
 Local SHG label : 34251
 Remote SHG labels : 2
  38216 : nexthop 192.168.0.1
  Arg.FE2 1:16 : nexthop 192.168.0.2
```

SRv6 Services: L3 EVPN

Table 15: Feature History Table

Feature Name	Release	Description
Support for SRv6 Services: L3 EVPN	Release 7.9.1	This feature adds support for carrying L3VPN routes in L2VPN EVPN RT5 address-family instead of VPNv4 unicast and/or VPNv6 unicast address-family across SRv6 core (EVPN over SRv6 underlay). This allows for efficient and scalable delivery of VPN services using SRv6 technology.

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

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

Usage Guidelines and Limitations

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

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

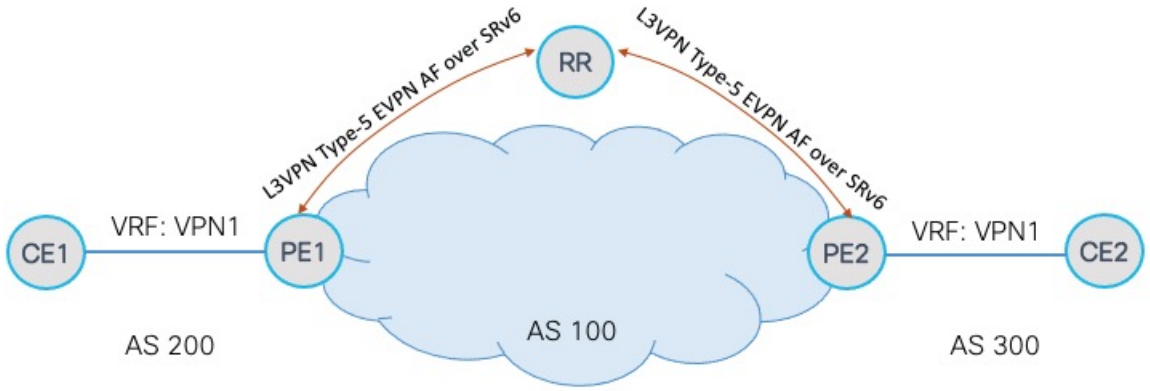
The following behaviors are supported:

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

Configuring SRv6-based L3 EVPN

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

Figure 10: Configuration Example: Dual Stack L3 EVPN over SRv6



Configure the VRF (Dual-Stack IPv4/IPv6)

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

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

```
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
```

Running Configuration

```
vrf VPN1
 address-family ipv4 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
  !
  export route-target
   1:1
  !
 !
!
router bgp 100
 address-family vpnv4 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy add-path
 !
 address-family vpnv6 unicast
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy add-path
 !
 address-family l2vpn evpn
  additional-paths receive
  additional-paths send
  additional-paths selection route-policy add-path
 !
```

```
 neighbor 1111::1
  remote-as 100
  address-family l2vpn evpn
   advertise vpnv4 unicast
   advertise vpnv6 unicast
  !
 !
 vrf VPN1
  rd 100:1
  address-family ipv4 unicast
   segment-routing srv6
    locator LOC1
    alloc mode per-vrf
   !
  !
  address-family ipv6 unicast
   segment-routing srv6
    locator LOC1
    alloc mode per-vrf
   !
  !
  neighbor 1.1.1.1
   remote-as 200
   address-family ipv4 unicast
   !
  !
  neighbor 3333::3
   remote-as 200
   address-family ipv6 unicast
   !
  !
 !
!
```

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

Table 16: Feature History Table

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

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

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

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

fields when the SRv6 Service SID is signaled in split parts to enable the receiver to put together the SID accurately.

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

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

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

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

Example

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

- BGP Path 1 from next-hop 7::1 and a 32-bit uDT4 function (0xfff0 4002) allocated from W-LIB
- BGP Path 2 from next-hop 9::1 and a 16-bit uDT4 function (0x4002) allocated from LIB
- BGP Path 3 from next-hop 8::1 and a 16-bit uDT4 function (0x4002) allocated from LIB

Note the following fields in the output:

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

```
Router# show bgp vpnv4 un 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
```

SRv6 Services: L2 EVPN Services with Local SIDs from W-LIB

Table 17: Feature History Table

Feature Name	Release	Description
SRv6-Services: L2 EVPN 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 Local W-LIB is supported for Layer 2 EVPN services (ELAN/ELINE) services. Users with a large number of L2 EVPN services can now have a larger address space (local ID block) for local identifiers because this functionality enables an SRv6 headend node to allocate and advertise local SIDs with Wide (32-bit) functions (Local W-LIB). The larger address space for local identifiers ensures efficient address space utilization and better scalability for BGP deployments. This feature introduces the **usid allocation wide-local-id-block** command.

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

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

By default, in BGP, the allocation of uSIDs is limited to the LIB space. However, with this feature, BGP can instruct the SID-Manager to allocate uSIDs specifically from the W-LIB (Wide Local Identifier Block). This enforcement provides a larger address space for local identifiers, which allows for more efficient utilization of the address space and better scalability for BGP deployments.

BGP uses transposition to encode the service SID for VPN services into the label part of the NLRI, as specified in IETF RFC 9252. In the current LIB implementation, BGP transposes the 16-bit function into the label field of the NLRI. In the case of W-LIB, BGP transposes the last 16-bits of the W-LIB 32-bit function into the label part of the NLRI for EVPN routes.

For example, if a local identifier looks like BBBB:BBBB:Loc:Service, resulting in an IPv6 address like FC00:0100:8200:F111. With the W-LIB extension, the local identifier includes two additional segments for the MSB (Most Significant Bits) and LSB (Least Significant Bits), resulting in an IPv6 address like FC00:0100:8200:F111:1111.

For more information on transposition, see the *SRv6 Services: L2 and L3 Services with Remote SIDs from W-LIB* section.

Usage Guidelines and Limitations

- The **usid allocation wide-local-id-block** command cannot be used together with the **alloc mode route-policy** command in BGP. The `wide-local-id-block` SID allocation does not support dynamic allocation from locators based on route-policy selection.

- This feature is supported on thrid, fourth, and fifth generation ofASR 9000 Series High-Density Ethernet line cards. Refer to the *Cisco ASR 9000 Series Aggregation Services Router Ethernet Line Card Installation Guide* for details on ASR 9000 Series Line Cards.

Configuration

The local WLIB configuration can be applied at three levels:

- Globally under EVPN
- Global locator
- Per-EVI locator

Use the **usid allocation wide-local-id-block** command to enable the allocation and advertisement of an SRv6 Service SID with wide function (W-LIB) for L2 EVPN services.

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

- W-LIB uSID Allocation Applied Globally under EVPN:

```
evpn
 segment-routing srv6
  locator loc3
  !
  usid allocation wide-local-id-block
 !
!
```

The following ouput shows the W-LIB uSID allocation under global EVPN:

```
RP/0/0/CPU0:PE1# show evpn segment-routing srv6 detail

Configured default locator: loc3
Configured default SID Function Length: 32 bits
EVIs with unknown locator config: 0
VPWS with unknown locator config: 0
Global SID Function Length: 32 bits

Locator name    Prefix                      OOR      Flags                   Service
count    SID count
------------    ------                    ---      -----                 -------------
    ---------
loc3            fccc:cc01:1::/48            False    micro-segment           6
        6
   Default locator
   Configured on EVIs <evi>: 4
```

- W-LIB uSID Allocation Applied under the Global Locator for EVPN:

```
evpn
 segment-routing srv6
  locator loc3
   usid allocation wide-local-id-block
  !
 !
!
```

The following ouput shows the W-LIB uSID allocation under global locator:

```
RP/0/0/CPU0:PE1# show evpn segment-routing srv6 detail

Configured default locator: loc3
Configured default SID Function Length: 32 bits
EVIs with unknown locator config: 0
VPWS with unknown locator config: 0
Global SID Function Length: 16 bits

Locator name    Prefix                      OOR      Flags                   Service
count    SID count
------------    ------                    ---      -----                 -------------
    ---------
loc3            fccc:cc01:1::/48            False    micro-segment           6
        6
   Default locator
   Configured on EVIs <evi>: 4
```

- W-LIB uSID Allocation Applied under Specific EVI Locator:

```
evpn
 evi 4 segment-routing srv6
  locator loc3
   usid allocation wide-local-id-block
  !
 !
!
```

The following ouput shows the W-LIB uSID allocation for a apecific EVI locator:

```
RP/0/0/CPU0:PE1# show evpn evi vpn-id 4 detail

VPN-ID     Encap      Bridge Domain                 Type
---------- ---------- ----------------------------- --------------------
4          SRv6       VPWS:4                        VPWS (vlan-unaware)
   Stitching: Regular
   Unicast SID:  <no SIDs>
   Multicast SID:  <no SIDs>
   E-Tree: Root
   Forward-class: 0
   Advertise MACs: No
   Advertise BVI MACs: No
   Aliasing: Enabled
   UUF: Enabled
   Re-origination: Enabled
   Multicast:
     Source connected   : No
     IGMP-Snooping Proxy: No
     MLD-Snooping Proxy : No
   BGP Implicit Import: Enabled
   VRF Name:
   SRv6 Locator Name: loc3
   SRv6 SID Function Length: 32 bits
   Preferred Nexthop Mode: Off
   Ignore MTU Mismatch: Enabled
   Transmit MTU Zero: Enabled
   BVI Coupled Mode: No
   BVI Subnet Withheld: ipv4 No, ipv6 No

   Statistics:
     Packets            Sent                 Received
       Total          : 0                    0
       Unicast        : 0                    0
       BUM            : 0                    0
```

```
  Bytes              Sent                 Received
    Total          : 0                    0
    Unicast        : 0                    0
    BUM            : 0                    0
RD Config: none
RD Auto  : (auto) 192.168.0.1:4
RT Auto  : 100:4
Route Targets in Use           Type
------------------------------ --------------------
100:4                          Import
100:4                          Export
```

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

Table 18: Feature History Table

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

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

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

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

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

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

Note There is no transposition for BGPv4/BGPv6 global routing table.

Usage Guidelines and Limitations

- This feature is supported on third, fourth, and fifth generation of ASR 9000 Series High-Density Ehternet line cards. Refer to the *Cisco ASR 9000 Series Aggregation Services Router Ethernet Line Card Installation Guide* for details on ASR 9000 Series Line Cards.
- The **usid allocation wide-local-id-block** command cannot be used together with the **alloc mode route-policy** command in BGP. The `wide-local-id-block` SID allocation does not support dynamic allocation from locators based on route-policy selection.

Configuration

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

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

- W-LIB uSID Allocation Applied Globally under BGP:

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

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

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

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

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

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

```
router bgp 1
  vrf foo
    address-family ipv4 unicast
```

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

Verification

The following ouput shows the W-LIB uSID allocation:

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

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

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

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

```
Table version synced to RIB: 44
Table version acked by RIB: 44
IGP notification: IGPs notified
RIB has converged: version 0
RIB table prefix-limit reached ?  [No], version 0
Permanent Network Unconfigured
Segment Routing SRv6 Alloc Mode: 0
Segment Routing SRv6 uSID WLIB allocation: Enforced


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

VRF: foo
-------

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

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

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

VRF foo Address family: IPv4 Unicast
Dampening is enabled
Client reflection is not enabled in global config
Dynamic MED is Disabled
Dynamic MED interval : 10 minutes
Dynamic MED Timer : Not Running
Dynamic MED Periodic Timer : Not Running
Scan interval: 60
Total prefixes scanned: 85
Prefixes scanned per segment: 100000
Number of scan segments: 1
```

```
Nexthop resolution minimum prefix-length: 0 (not configured)
IPv6 Nexthop resolution minimum prefix-length: 0 (not configured)
Main Table Version: 152
Table version synced to RIB: 152
Table version acked by RIB: 152
IGP notification: IGPs notified
RIB has converged: version 1
RIB table prefix-limit reached ?  [No], version 0
Permanent Network Unconfigured
Segment Routing SRv6 uSID WLIB allocation: Enforced
```

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

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

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

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

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

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

Static SRv6 pseudo-wire

Static SRv6 pseudo-wire (PW) gives the flexibility to

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

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
Static SRv6 pseudowire	Release 25.1.1	This feature introduces support for Static Segment Routing over IPv6 (SRv6) Pseudo-wires in the IOSXR platform. This enhancement aims to extend the existing Virtual Private Wire Service (VPWS) capabilities by incorporating SRv6, providing improved flexibility and scalability for service providers.

Key aspects of SRv6 static pseudo-wire

This feature enhances the existing Virtual Private Wire Service (VPWS) by incorporating Segment Routing over IPv6 (SRv6) capabilities, which previously supported only MPLS.

This enhancement enables the configuration and management of static SRv6 pseudo-wires, 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 pseudo-wires that rely on signaling protocols like LDP or BGP, static pseudo-wires are manually configured. This allows for more control and customization in specific network scenarios.
- **Segment Routing over IPv6 (SRv6)**: SRv6 is a network architecture that uses IPv6 addresses to encode the path that packets should follow through the network. Each segment in the path is represented by an SRv6 Segment Identifier (SID).
- **Pseudo-wire (PW)**: A pseudo-wire 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.

Benefits of Static SRv6 pseudo-wire

The benefits of having an option to configure pseudo-wire 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 pseudo-wires, improving network efficiency and performance.
- **Scalability**: SRv6 allows for a scalable solution that can handle a large number of pseudo-wires 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.

Follow these steps to configure static SRv6 pseudo-wire

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:

```
Router#sh run int be101.40010001

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::
```

SRv6/MPLS L3 Service Interworking Gateway

Table 20: Feature History Table

Feature Name	Release	Description
Identical Route Distinguisher (RD) for Interworking Gateways between MPLS and SRv6 Domains	Release 24.1.1	You can now configure the same Route Distinguisher (RD) for interworking gateways catering to both MPLS and SRv6 domains that help conserve hardware resources, reduce the BGP table scale and minimize the processing load on routers. At the same time, it ensures seamless connectivity across SRv6 and MPLS L3 EVPN domains, thus promoting interoperability and efficiency in modern network environments. Previously, a unique RD was required to extend L3 services between MPLS and SRv6 domains resulting in higher router load and resource consumption, which could have affected performance.
SRv6/MPLS L3 Service Interworking Gateway (SRv6 Micro-SID)	Release 7.3.2	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 a way to migrate from MPLS L3VPN to SRv6 L3VPN.

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

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

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

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

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

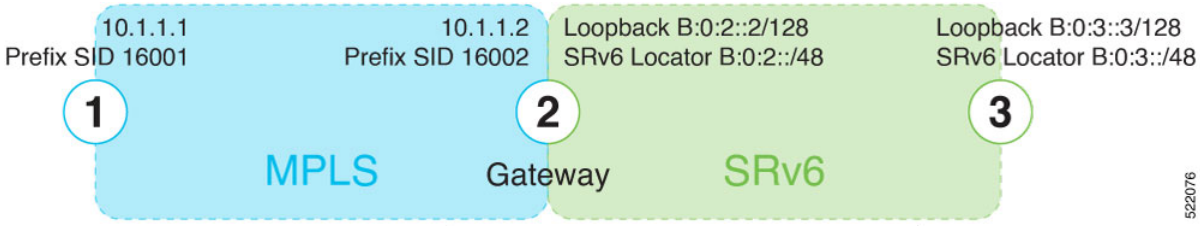
The gateway performs the following actions:

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

SRv6/MPLS L3 Service Interworking Gateway Scenarios

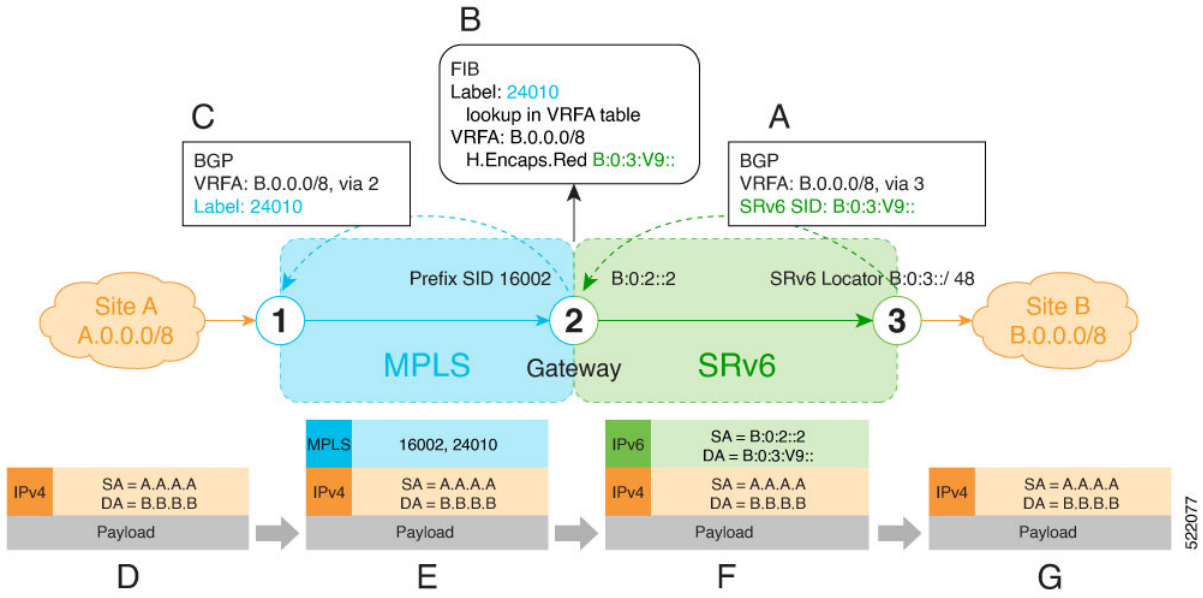
The following scenario is used to describe the gateway functionality:

- Node 1 is an L3VPN PE in the MPLS domain with an SR prefix SID label of 16001 for its Loopback interface 10.1.1.1/32.
- Node 2 is the SRv6/MPLS L3 Service Interworking Gateway. In the MPLS domain, it has an SR prefix SID label of 16002 for its Loopback interface 10.1.1.2/32. In the SRv6 domain, it has an SRv6 locator of B:0:2::/48 and Loopback interface B:0:2::2/128.
- Node 3 is an L3VPN PE in the SRv6 domain with SRv6 locator of B:0:3::/48 and Loopback interface B:0:3::3/128.



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

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



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

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

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

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

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

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

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

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

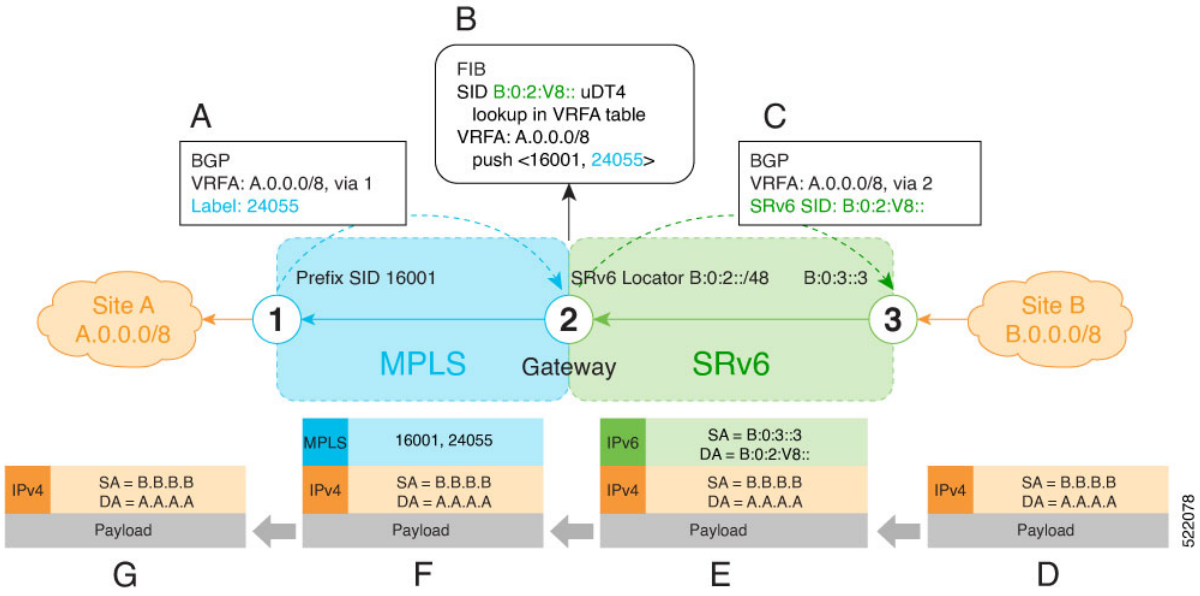
F. Node 2 performs the following actions:

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

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

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

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



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

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

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

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

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

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

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

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

F. Node 2 performs the following actions:

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

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

Example

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

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

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

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

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

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

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

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 neighbor 10.1.1.1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   route-reflector-client
   advertise vpnv4 unicast re-originated
```

```
 !
 neighbor B:0:3::1
  address-family vpnv4 unicast
   import stitching-rt re-originate
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated stitching-rt
 !
 vrf ACME
  address-family ipv4 unicast
   enable label-mode
   segment-routing srv6
```

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

```
/* Configuration on Node 1*/
vrf ACMErd 5000:2
 address-family ipv4 unicast
  import route-target
   1111:1
  !
  export route-target
   1111:1
  !
 !
!

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

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

L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway

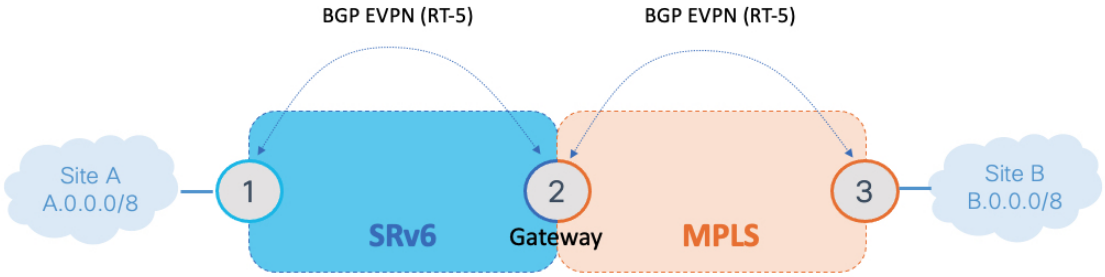
Table 21: Feature History Table

Feature Name	Release	Description
Support for L3 EVPN/SRv6 and L3 EVPN/MPLS Interworking Gateway	Release 7.9.1	This feature allows for SRv6 L3 EVPN domains to interwork with existing MPLS L3 EVPN domains, enabling you to extend L3 EVPN services between SRv6 and MPLS domains by providing service continuity on the control plane and data plane. This feature allows for SRv6 L3 EVPN domains to interwork with existing MPLS L3 EVPN domains. The feature also allows a way to migrate from MPLS L3 EVPN to SRv6 L3 EVPN.

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

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

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



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

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

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

The gateway performs the following actions:

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

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

- MPLS-to-SRv6 Control Plane Direction

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

- SRv6-to-MPLS Control Plane Direction

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

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

Usage Guidelines and Limitations

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

Configuration Example

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

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

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

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

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

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

```
   2222:1 stitching
  !
  export route-target
   1111:1
   2222:1 stitching
  !
 !
!
```

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

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
 !
 neighbor 2222::2
  remote-as 100
  description SRv6 side peering
  address-family l2vpn evpn
   import reoriginate stitching-rt (Imports NLRIs that match normal route target
             identifier and exports re-originated NLRIs assigned with the stitching
             route target identifier)
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated (Specifies advertisement of re-originated
             VPNv4 unicast routes)
   advertise vpnv6 unicast re-originated (Specifies advertisement of re-originated
             VPNv6 unicast routes)
  !
 !
 neighbor 3.3.3.3
  remote-as 100
  description MPLS side peering stitching side
  address-family l2vpn evpn
   import stitching-rt reoriginate (Imports NLRIs that match stitching route target
             identifier and exports re-originated NLRIs assigned with the normal route
             target identifier)
   advertise vpnv4 unicast re-originated stitching-rt (Advertise local VPNv4 unicast
             routes assigned with stitching route target identifier)
   advertise vpnv6 unicast re-originated stitching-rt (Advertise local VPNv6 unicast
             routes assigned with stitching route target identifier)
  !
 !
 vrf VPN1
  rd 100:2
  address-family ipv4 unicast
   mpls alloc enable
  !
  address-family ipv6 unicast
   mpls alloc enable
  !
 !
!
```

L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway

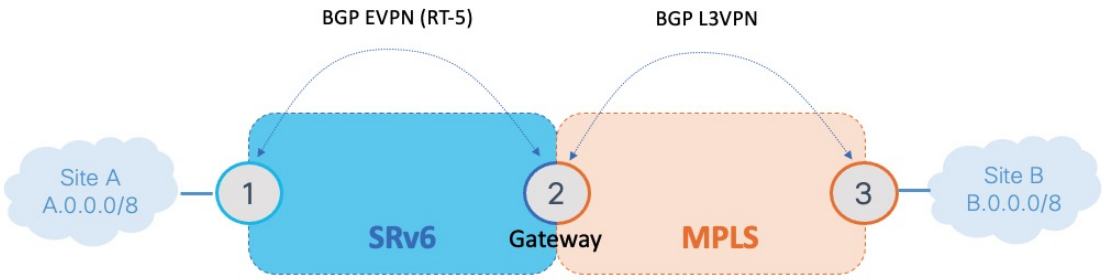
Table 22: Feature History Table

Feature Name	Release	Description
Support for L3 EVPN/SRv6 and L3VPN/MPLS Interworking Gateway	Release 7.9.1	This feature allows for SRv6 L3 EVPN domains to interwork with existing MPLS L3VPN domains, enabling you to extend L3 services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane. The feature also allows a way to migrate from MPLS L3VPN to SRv6 L3 EVPN.

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

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

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



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

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

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

The gateway performs the following actions:

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

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

- MPLS to SRv6 Control Plane Direction

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

- SRv6 to MPLS Control Plane Direction

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

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

Usage Guidelines and Limitations

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

Configuration Example

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

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

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

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

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

```
  !
 !
!
```

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

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
 !
 neighbor 2222::2
  remote-as 100
  description SRv6 side peering
  address-family l2vpn evpn
   import reoriginate stitching-rt (Imports NLRIs that match normal route target
             identifier and exports re-originated NLRIs assigned with the stitching
             route target identifier)
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated (Specifies advertisement of re-originated
             VPNv4 unicast routes)
   advertise vpnv6 unicast re-originated (Specifies advertisement of re-originated
             VPNv6 unicast routes)
  !
 !
 neighbor 3.3.3.3
  remote-as 100
  description MPLS side peering stitching side
  address-family 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 layer 3 services between two distinct SRv6 domains
- enables seamless service continuity on both control and data planes for SRv6-based networks, and
- supports route re-origination and summarization between two domains for SRv6 VPNv4 and VPNv6 address families.

Table 23: Feature History Table

Feature Name	Release	Description
Layer 3 service gateway for interconnecting SRv6 domains	Release 25.3.1	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:

- Optimizing Encap ID consumption: 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.

- Enabling 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 highlights and 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.

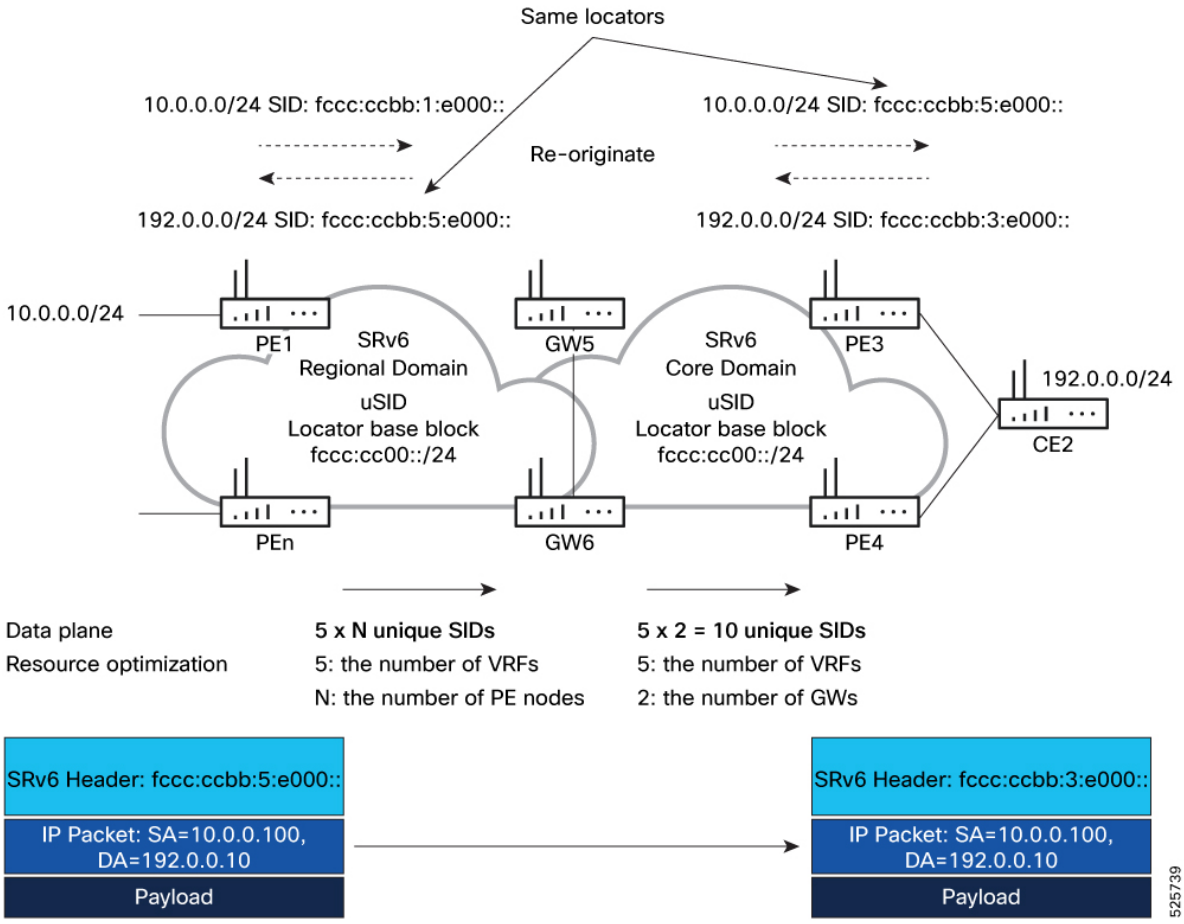
- Regional domains: Specific SRv6 domains representing partitioned parts of a larger L3VPN network, connected to a core domain through the gateway.
- Core domain: The central SRv6 domain in a partitioned L3VPN network, connected to regional domains through the gateway.
- Base format SRv6 locators: A 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.

Workflow

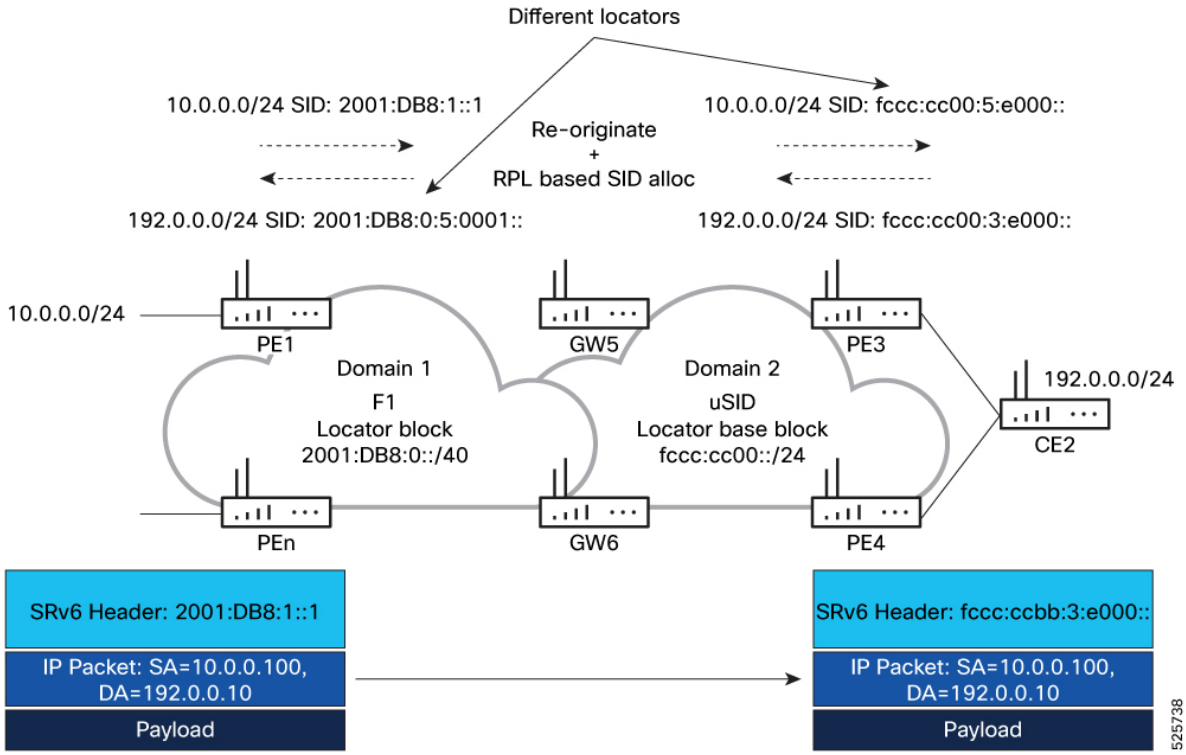
These stages describe SRv6 layer 3 service gateway interconnect domains:

Control plane:

1. SID re-origination and summarization (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. Format translation (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.



Data plane

1. Traffic arrival: Traffic originating from one SRv6 domain arrives at the SRv6 L3 service gateway.
2. Pop or decapsulation: The gateway decapsulates the incoming SRv6 packet by removing its existing SRv6 header.
3. 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.
4. 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.
5. Traffic forwarding: The newly encapsulated traffic is then pushed out towards the target SRv6 domain.

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

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:

```
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
```

Configure SRv6 layer 3 service gateway with different SID formats

Before you begin

- 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
```

SRv6/MPLS Dual-Connected PE

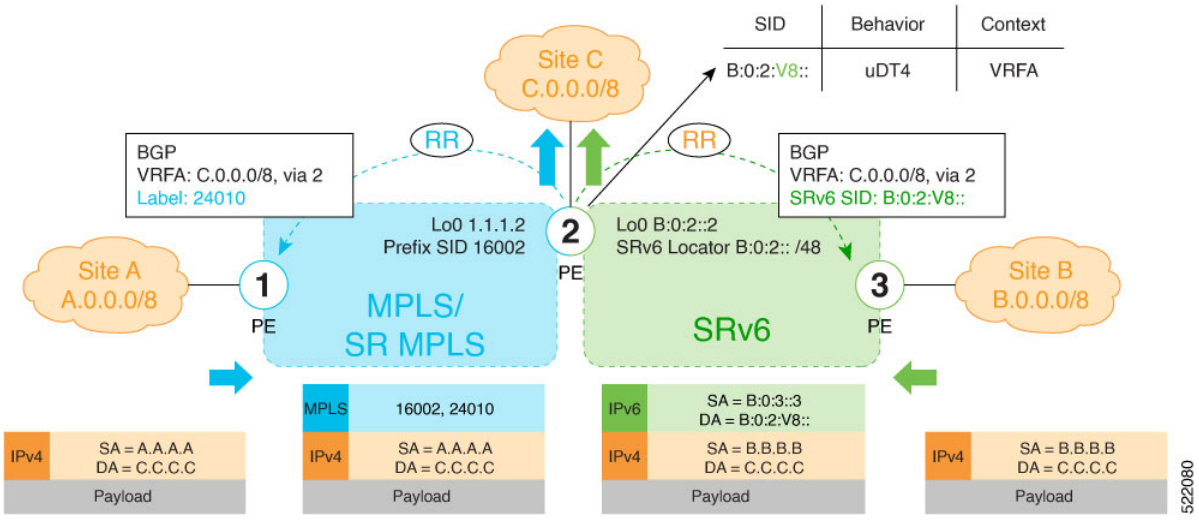
Table 24: Feature History Table

Feature Name	Release	Description
SRv6/MPLS Dual-Connected PE (SRv6 Micro SID)	Release 7.3.2	This feature allows a PE router to support IPv4 L3VPN services for a given VRF with both MPLS and SRv6. This is MPLS and SRv6 L3VPNv4 co-existence scenario and is sometimes referred to as dual-connected PE.

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

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

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



Configure BGP to Support Dual-Mode

Enable MPLS Label Allocation

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

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

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

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

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

Running Config

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

SRv6 Provider Edge (PE) Lite Support

Table 25: Feature History Table

Feature Name	Release	Description
SRv6 Provider Edge (PE) Lite	Release 7.8.1	This feature provides VPN de-multiplexing-only behaviors (End.DT4/DT6/DT46) at an SRv6 PE node. This allows for a lightweight-PE implementation (no VPN encapsulation) that steers SRv6-encapsulated traffic across an SR-MPLS backbone after performing a VPN lookup.

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

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

Note See Configuring Explicit End.DT46 SRv6 SIDs, on page 34 for information about explicit End.DT46 SRv6 SIDs.

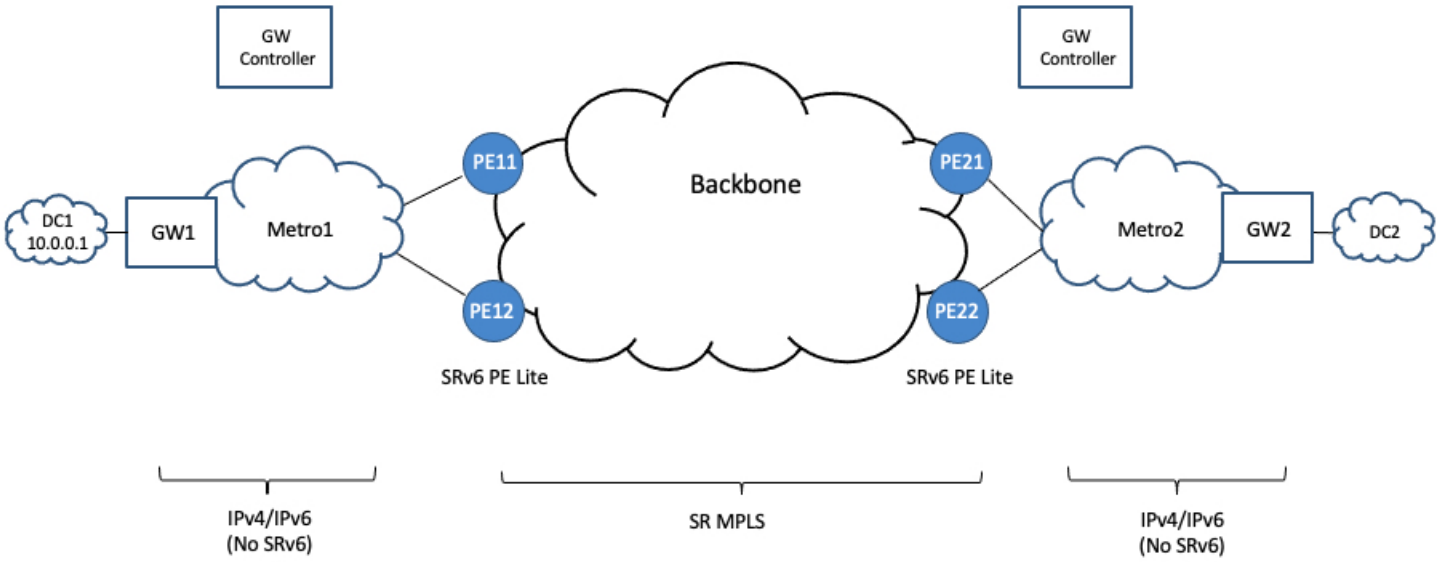
When traffic arrives at the ingress PE, it performs End.DT46 SRv6 end-point behavior.

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

Use Case

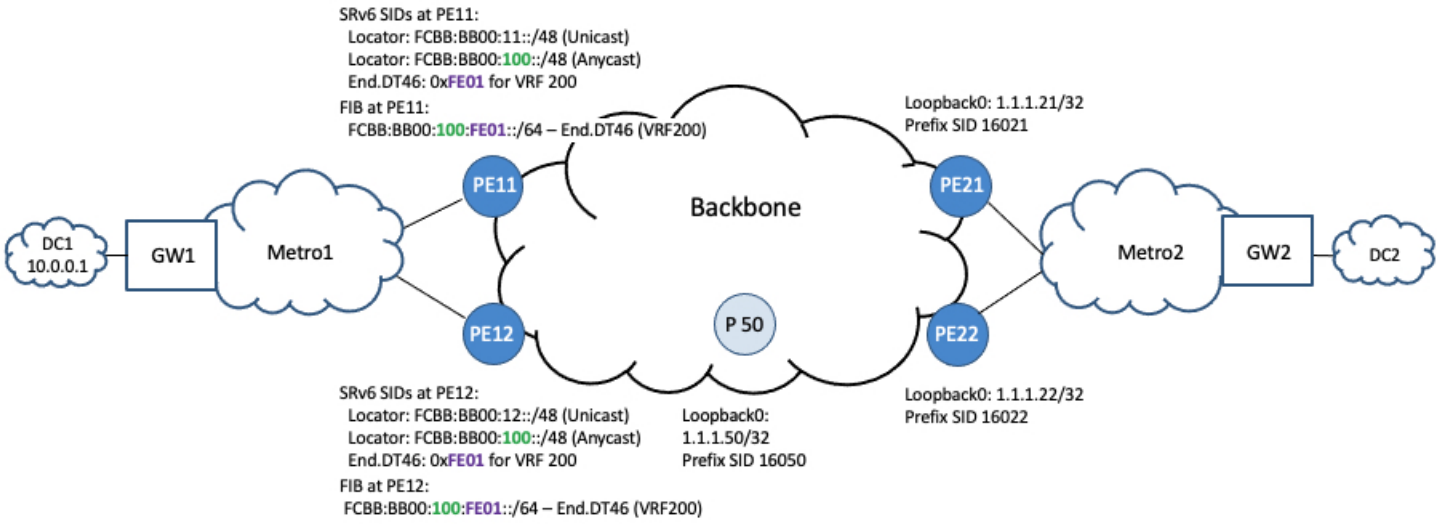
The figure below shows a use case where inter-data-center traffic is encapsulated in SRv6 (IP-in-IPv6) and is carried over IP-only metro domains and then over an SR-MPLS backbone.

Figure 11: Example Topology



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

Figure 12: SRv6 Locator/Functions and SR-MPLS Prefix SIDs (Traffic Direction – DC1 to DC2)

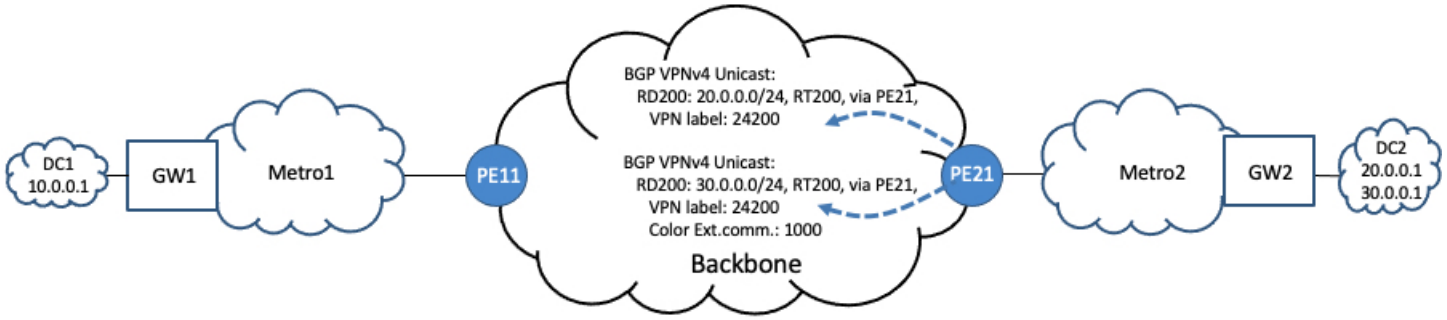


The SRv6 PE lite nodes are configured with SRv6 locators and explicit (manually assigned) service de-multiplexing end-point behaviors to perform decapsulation and VPN table lookup.

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

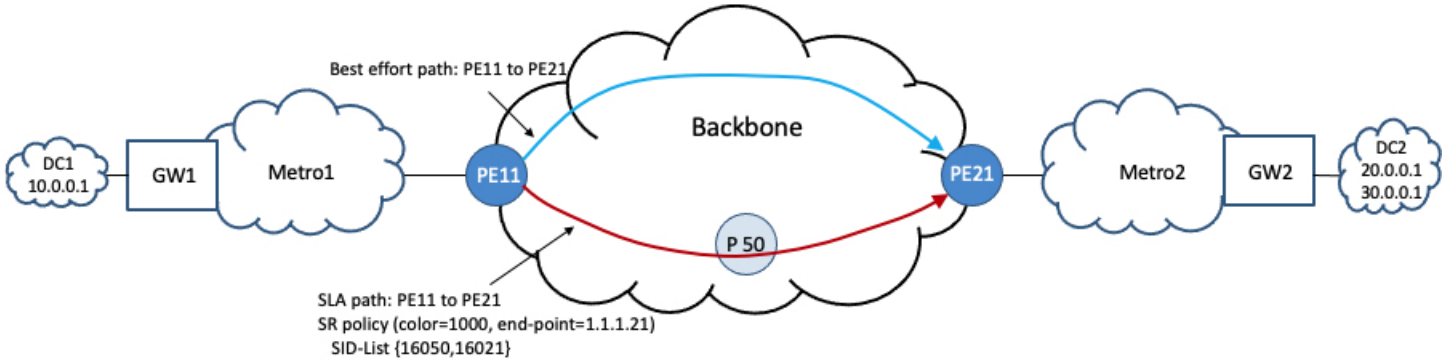
For example, SRv6 PE lite nodes (PE11 and PE12) are configured with the Anycast SRv6 locator (FCBB:BB00:100::/48) and a common End.DT46 function (0xFE01) associated with MPLS VPN VRF 200. The SRv6 PE lite nodes, which are part of the SR MPLS backbone, are configured with corresponding prefix SIDs.

Figure 13: BGP VPN Overlay Route Advertisement (Traffic Direction – DC1 to DC2)



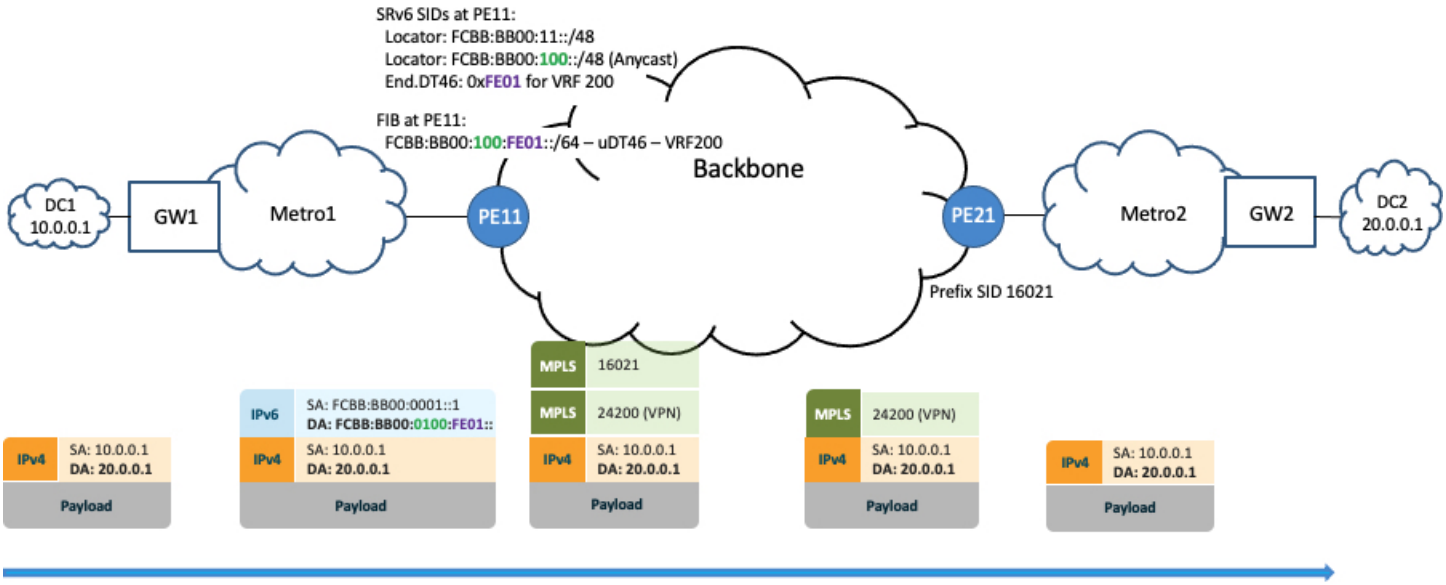
Prefixes from the data center are advertised in the backbone via multiprotocol BGP as part of a VPN. These prefixes can include a color extended community in order to indicate the desired transport SLA. For example, PE21 advertises BGP VPN overlay routes for DC2, 20.0.0.0/24 and 30.0.0.0/24. Prefix 20.0.0.0/24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

Figure 14: Backbone Transport Paths (Traffic Direction – DC1 to DC2)



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

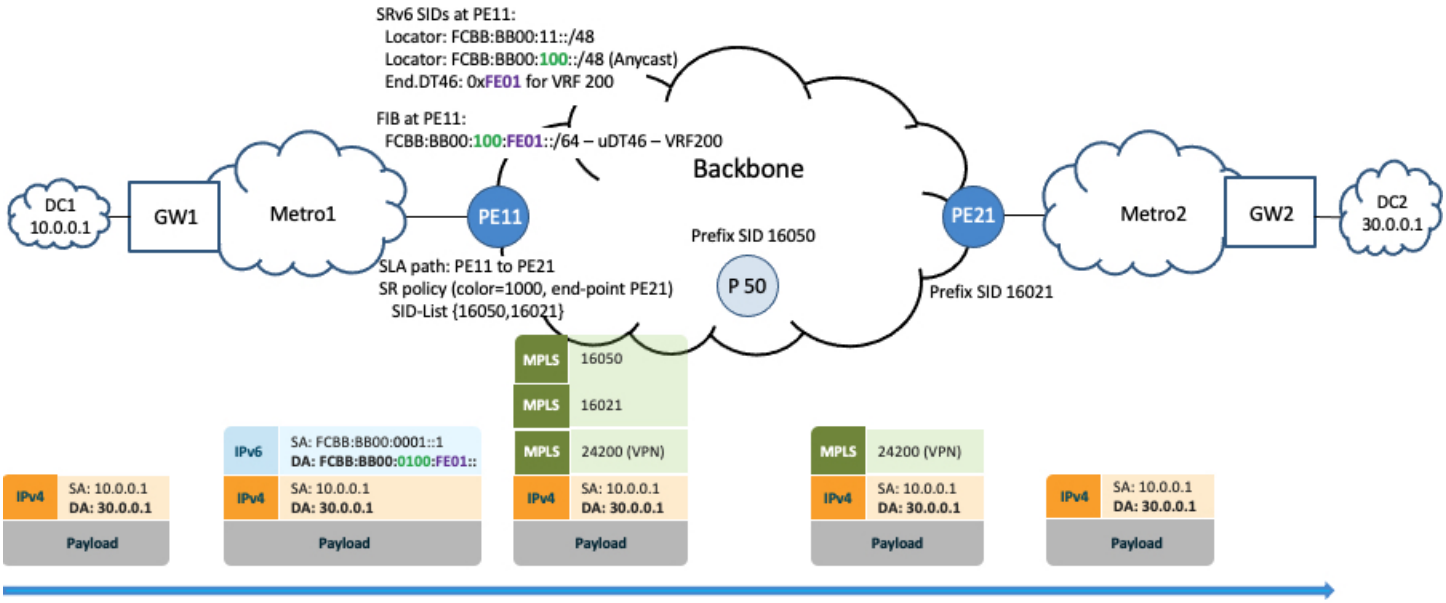
Figure 15: Best Effort Traffic (Traffic Direction – DC1 to DC2)



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

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

Figure 16: SLA Traffic (Traffic Direction – DC1 to DC2)



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

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

Configuration for SRv6 PE Lite Node 11

Configure SRv6:

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

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

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

Configure interface Loopback0:

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

Configure the SR policy:

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

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

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

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

Configure BGP:

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

SRv6 SID Information in BGP-LS Reporting

BGP Link-State (BGP-LS) is used to report the topology of the domain using nodes, links, and prefixes. This feature adds the capability to report SRv6 Segment Identifier (SID) Network Layer Reachability Information (NLRI).

The following NLRI has been added to the BGP-LS protocol to support SRv6:

- Node NLRI: SRv6 Capabilities, SRv6 MSD types
- Link NLRI: End.X, LAN End.X, and SRv6 MSD types
- Prefix NLRI: SRv6 Locator
- SRv6 SID NLRI (for SIDs associated with the node): Endpoint Function, BGP-EPE Peer Node/Set

This example shows how to distribute IS-IS SRv6 link-state data using BGP-LS:

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



Note It is still possible to ping or trace a SID:

- **ping** B:k:F::
- **traceroute** B:k:F::

It is possible to use a list of packed carriers to ping or trace a SID, to ping or trace route, use **<destination SID> via srv6-carriers <list of packed carriers>**

DHCPv4 Relay Agent and Proxy Support over SRv6

This feature introduces support for DHCPv4 Relay Agent and Proxy over SRv6.

An IOS XR router can act as a DHCPv4 relay agent/proxy with a DHCPv4 server connected over an SRv6 network.

The following functionality is supported:

- DHCPv4 relay agent/proxy over SRv6 with DHCPv4 server (helper-address) located in default VRF (global)
- DHCPv4 relay agent/proxy over SRv6 with DHCPv4 server (helper-address) located in non-default VRF
- DHCPv4 relay agent/proxy on interfaces associated with a default VRF (global)
- DHCPv4 relay agent/proxy on interfaces associated with a non-default VRF
- DHCPv4 relay agent/proxy on Ethernet physical interfaces
- DHCPv4 relay agent/proxy on Ethernet bundle interfaces

For information on configuring DHCPv4 relay agent and proxy, refer to the “Implementing the Dynamic Host Configuration Protocol” chapter in the *IP Addresses and Services Configuration Guide* .

DHCPv6 Relay Agent Support over SRv6

This feature introduces support for DHCPv6 Relay Agent over SRv6.

An IOS XR router can act as a DHCPv6 relay agent with a DHCPv6 server connected over an SRv6 network.

The following functionality is supported:

- DHCPv6 relay agent over SRv6 with DHCPv6 server (helper-address) located in default VRF (global)
- DHCPv6 relay agent over SRv6 with DHCPv6 server (helper-address) located in non-default VRF
- DHCPv6 relay agent on interfaces associated with a default VRF (global)
- DHCPv6 relay agent on interfaces associated with a non-default VRF
- DHCPv6 relay agent on Ethernet physical interfaces
- DHCPv6 relay agent on Ethernet bundle interfaces

For information on configuring DHCPv6 relay agent, refer to the “Implementing the Dynamic Host Configuration Protocol” chapter in the *IP Addresses and Services Configuration Guide*.

Full-Replace Migration to SRv6 Micro-SID

During the Full-Replace migration, both underlay and services are migrated from format1 to f3216. The underlay migration is done using the *Ship in the night* strategy, where updates into your environment are incremental, thereby phasing out your existing transport protocols when ready. This method minimizes the service disruption, and is recommended for seamless migration. The services migration is done using *swap* procedures, where the incoming transport label is swapped with an outgoing transport label.

The format1 to f3216 migration is seamless, requires minimal configurations, and no IETF signaling extensions. The migration enables preference of Micro-SID f3216 over format1, and minimizes traffic drop with faster convergence.

EVPN supports migration of the following services from format1 to f3216:

- IS-IS underlay (TILFA, uLoop, FlexAlgo)
- L3 overlay (VPNv4/VPNv6 and IPv4/IPv6)
- L2 overlay (EVPN VPWS - All-Active Multi-Homing)
- SRv6-MPLS IW Gateway, dual-connected PE

The following modes are supported in the context of migration:

- **Base**: SRv6 classic with format1 only.
- **Dual**: SRv6 classic with format1 and SRv6 Micro-SID with f3216 will both coexist.
- **f3216**: Micro-segment format. f3216 represents the format 3216, which is 32-bit block and 16-bit IDs.

The migration process involves the following steps:

1. **Prepare for migration**: Upgrade the network nodes to an image that is Micro-SID f3216 capable, and allows the coexistence of format1 and f3216.

2. **Migrate the underlay to f3216**: Enable IS-IS as an underlay protocol on PE nodes. The IS-IS configuration adds f3216 locators to format1 locators. Both format1 and f3216 endpoint SIDs are allocated, installed, and announced during this stage. f3216 is the preferred option over format1 for underlay paths.

   The IS-IS SR headends provide faster convergence to f3216. Faster convergence to f3216 is done on the per-prefix per-path level, does not need any new CLI, and avoids packet drops. The format1 locators are removed after underlay traffic convergence to f3216 on all nodes. The format1 locators are unconfigured from IS-IS, and deleted from SRv6.

   At the end of this step, the migration status of the following P Nodes are:

   - Locator reachability: f3216 only
   - Underlay endpoint/headends: f3216 only

   At the end of this step, the migration status of the following PE Nodes are:

   - Locator reachability: format1 and f3216
   - Underlay endpoint/headends: f3216 only
   - Overlay endpoint/headends: format1

3. **Migrate the overlay to f3216**: Enables overlay f3216 under BGP and EVPN on all PE nodes. The BGP and EVPN configuration replaces format1 by f3216 locators. During this stage, the f3216 Micro-SIDs are allocated, installed, and announced, while the format1 SIDs are deallocated, uninstalled, and withdrawn.

   The format1 locators are removed after overlay traffic convergence to f3216 on all nodes. The format1 locators are unconfigured from BGP and EVPN, and deleted from SRv6. For a transient period, BGP and EVPN might have some paths with format1 and some with f3216.

   At the end of this step, the migration status of the following is:

   - For P/PE Nodes:
     - Locator reachability: f3216 only
     - Underlay endpoint/headends: f3216 only
     - Overlay endpoint/headends: f3216 only

The migration starts with SRv6 base format1, and ends with SRv6 Micro-SID f3216. The migration states are:

1. **Initial state**: This is the early migration state of a deployment, for the supported features. This state comprises SRv6 base with format1.

   This example shows the initial state of migration with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc0
Router(config-srv6-locators)# prefix f1bb:bbbb:bb00:0001::/64
```

This example shows the initial state of migration with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc0
```

This example shows the initial state of migration with SRv6 and BGP/EVPN:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myLoc0
```

2. **In-migration state**: The migration procedures are initiated, and are in progress. This state comprises SRv6 in dual mode (base with format1, and Micro-SID with f3216).

   This example shows the in-migration state with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc0
Router(config-srv6-locators)# prefix f1bb:bbbb:bb00:0001::/64
Router(config-srv6-locators)# delayed-delete
Router(config-srv6-locators)# locator myuLoc0
Router(config-srv6-locators)# micro-segment behavior unode psp-usd
Router(config-srv6-locators)# prefix fcbb:bb00:0001::/48
```

   This example shows the in-migration state with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc0
Router(config-isis-srv6)# locator myuLoc0
```

   This example shows the in-migration state with SRv6 and BGP/EVPN:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myuLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myuLoc0
```

3. **End state**: This is the state of deployment at the end of the migration. At the end state, you can update the network and add new features. The Full-Replace migration end state can be of two modes:

   - **Full-Replace**: Both underlay and overlay are migrated to Micro-SID f3216. Full-Replace is the Cisco recommended migration type.

- **uF1**: Underlay migrated to Micro-SID f3216, overlay remains format1. The uF1 migration is a transient state of the Full-Replace migration type.

This example shows the end state with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myuLoc0
Router(config-srv6-locators)# micro-segment behavior unode psp-usd
Router(config-srv6-locators)# prefix fcbb:bb00:0001::/48
```

This example shows the end state with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myuLoc0
```

This example shows the end state with SRv6 and BGP/EVPN:

```
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myuLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myuLoc0
```

Run the following command to check the result of migration, as shown in the example:

```
RP/0/RSP0/CPU0:Router# sh route ipv6 fc00:cc30:600:e004:: detail
Wed Nov 10 18:57:56.645 UTC

Routing entry for fc00:cc30:600::/48
    Known via "isis 2", distance 115, metric 141, SRv6-locator, type level-2
    Installed Nov 2 18:56:55.718 for 00:01:01
    Routing Descriptor Blocks
      fe80::232:17ff:fec3:58c0, from 7511::1, via TenGigE0/0/0/16.1, Protected
        Route metric is 141
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:1  Path ref count:0
        NHID:0x20006(Ref:193)
        Backup path id:65
      fe80::226:80ff:fe36:7c01, from 7511::1, via TenGigE1/0/9/1.1, Backup (TI-LFA)
        Repair Node(s): 3888::1
        Route metric is 251
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:65  Path ref count:1
        NHID:0x20007(Ref:163)
        SRv6 Headend:H.Insert.Red [f3216], SID-list {fc00:cc30:700::}
 Route version is 0x0 (8)
 No local label
```

```
IP Precedence: Not Set
QoS Group ID: Not Set
Flow-tag: Not Set
Fwd-Class: Not Set
Route Priority:RIB_PRIORITY_NON_RECURSIVE_LOW (8) SVD Type RIB_SVD_TYPE_LOCAL
Download Priority 2, Download Version 261731
No advertising protos.
```

Full-Replace Migration to SRv6 Micro-SID Restrictions

You need to reload the the line cards as the hardware profiles go through multiple transitions during the Full-Replace migration to SRv6 Micro-SID.

You can overcome the traffic drop duration at time of swap of format1 by f3216 on a PE node depending on the BGP/EVPN convergence using the **delayed_delete** command. When the **delayed_delete** command is configured against the format1 SID locator, RIB notifies EVPN about this change. The EVPN in turn stores the delayed flag in its RIB locator database.

SRv6 Traffic Accounting

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 Traffic Accounting	Release 7.10.1	You can now enable the router to record the number of packets and bytes transmitted on a specific egress interface for IPv6 traffic using the SRv6 locator counter. You can use this data to create deterministic data tools to anticipate and plan for future capacity planning solutions. This feature introduces or modifies the following changes: **CLI**: • **accounting prefixes ipv6 mode per-prefix per-nexthop srv6-locators** **YANG Data Models:** • `Cisco-IOS-XR-accounting-cfg` • `Cisco-IOS-XR-fib-common-oper.yang` (see GitHub, YANG Data Models Navigator)

SRv6 traffic accounting is an integral part of today's network for planning and forecasting traffic. Traffic accounting is the volume of aggregated traffic flows that enter, traverse, and leave the network in a given time. Traffic accounting is a solution to monitor the traffic that helps to measure traffic flows and record how much customer traffic is passing through the SR network.

To design a network topology and meet the defined Service-Level Agreement (SLA), capacity planning becomes essential for forecasting traffic load and failures. A complete view of the traffic in your network enables you to anticipate common failures, and provision for network expansion.

You can now monitor traffic on an ingress node of a domain that is SRv6 encapsulated towards an egress node of the domain. The traffic is recorded at the source using the per-locator, per-egress-interface (LOC.INT.E) counter, which is the locator per interface at egress to account the traffic. For a given locator (L) and interface (I), the router counts the number of packets and bytes for the traffic transmitted on the interface (I) with a destination address (DA) matching the locator L.

When this feature is enabled on routers, all traffic passing through the routers are accounted. These counters are periodically streamed through telemetry and you can retrieve the counters at any point.

To enable traffic accounting on PE and P routers, use the **accounting prefixes ipv6 mode per-prefix** command. You can retrieve the number of packets transmitted and received on the specific interface of a PE or P routers by using the following telemetry:

```
Cisco-IOS-XR-fib-common-oper:oef-accounting/vrfs/vrf[vrf-name='default']/afis/afi[afi-type=ipv6]/pfx/srv6locs/srv6loc
```

Benefits

Monitoring the traffic provides numerous benefits, and here are a few:

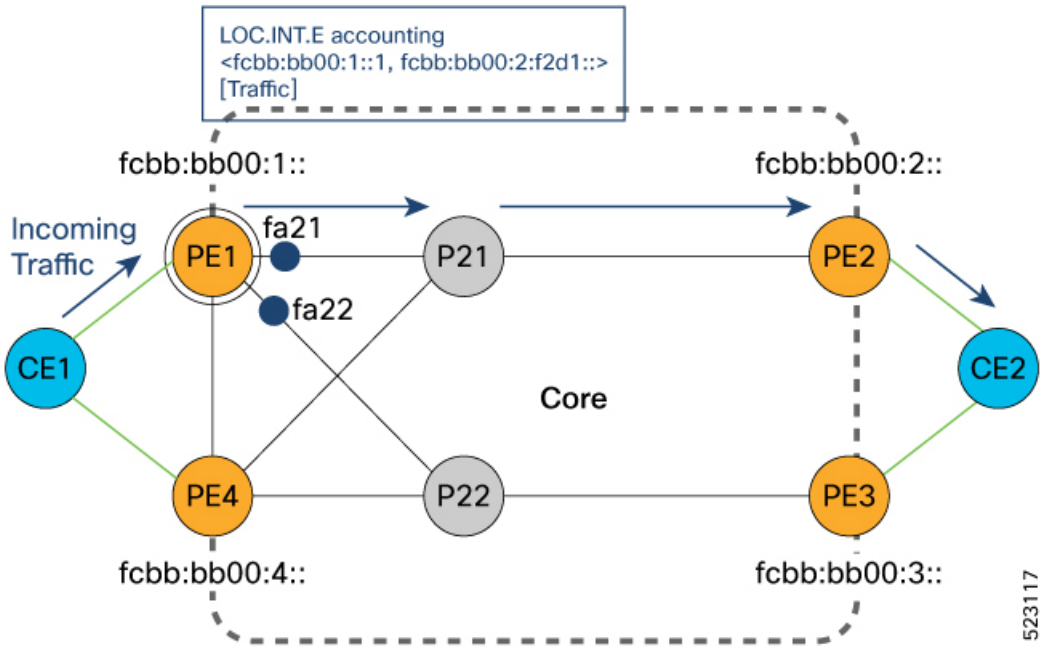
- To optimize network utilization and achieve a balance between underutilized and overutilized paths.
- To plan and optimize network capacity and avoid congestion.
- To plan the service provisioning and choose the right path and create an optimized backup path (for using SRLG's affinity, and so on).

Understanding SRv6 Locator Counters

Let's understand this feature with the following topology:

Consider the topology where traffic is passing from CE1 to CE2 through PE1. The traffic sent and received from CE1 is considered as the external traffic. The traffic from PE4 destined to PE2 is considered as the internal traffic.

Figure 17: Sample Topology for SRv6 Traffic Accounting



PE1 learns CE2 reachability through PE2. Consider PE1 has ECMP paths via P21 and P22 to reach PE2.

- When traffic reaches PE1, PE1 imposes traffic with the PE2 locator fcbb:bb00:2::.
- SRv6 traffic accounting LOC.INT.E is per prefix per egress interface accounting.

  When traffic exits the PE1 interface (fa21) through P21, PE1 keeps the count of this traffic that is sent. Also, when traffic exits the PE1 interface (fa22) through P22, PE1 keeps the count of this traffic that is sent. The traffic is accounted irrespective of the path PE1 takes to send traffic.

  Here is the SRv6 label of the outgoing traffic for PE2:

  ```
  <fcbb:bb00:1::1, fcbb:bb00:2:f2d1::> [CUSTTraffic]
  ```

- When the next set of packets are received and passed through PE1, the counters are incremented on fa21or fa22 interface based on the path the traffic sent through PE2.

The traffic from PE4 to PE1 is considered as internal traffic.

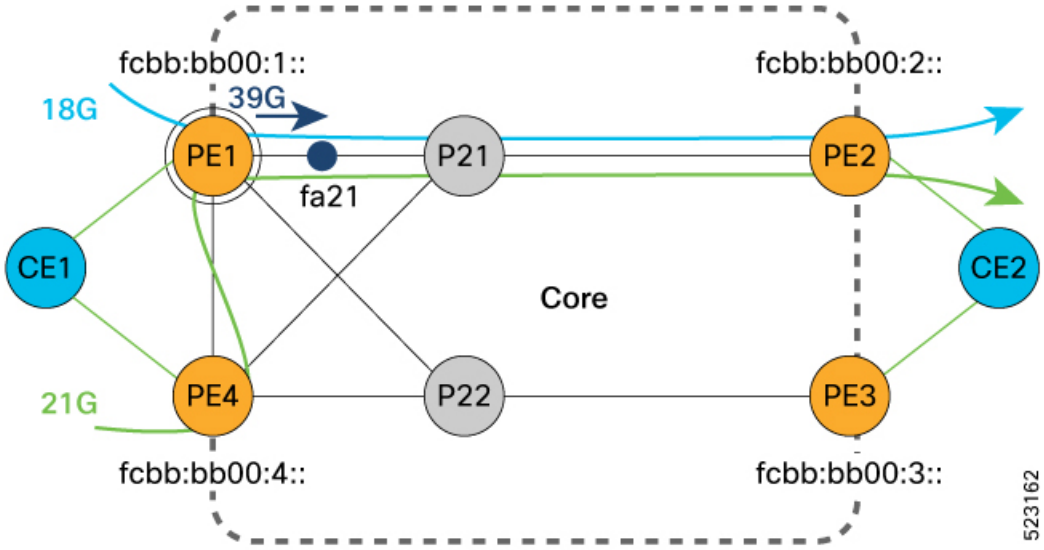
- When traffic is sent from PE4 to PE2 through PE1, PE4 imposes the traffic with the PE2 locator ID fcbb:bb00:2::. The traffic count is recorded at PE4 for this locator ID.
- When traffic reaches PE1, it looks for the PE2 locator ID and keeps the traffic count at PE1 when the traffic exit the fa21 interface.

Let's see how the SRv6 traffic is calculated using the demand matrix.

The Demand Matrix (DM) also known as a traffic matrix is a representation of the amount of data transmitted between every pair of routers. Each cell in the DM represents a traffic volume from one router to another. DM gives a complete view of the traffic in your network.

In the topology, the amount of external traffic destined for PE2 is a combination of external and internal traffic.

Figure 18: SampleTopology for Demand Matrix



- The traffic transmitted from PE1 is marked in **blue**.
- The traffic transmitted from PE4 is marked as in **green**.

The external traffic that PE2 receives is equal to the total traffic sent out from PE1 minus the received internal traffic.

```
External traffic to PE2
= (Total traffic sent out from PE1) - (Internal traffic received by PE1)
= (sum of all Loc.int.E counters on PE1) - ( sum of the Loc.int.E counters of all neighbors
 of PE1)
```

Let's try to calculate with this example.

- PE1 transmits a total of 39 gigabits per second towards PE2.
- PE1 receives 21 gigabits per second of internal traffic from PE4.
- PE1 receives 0 gigabits per second from P21 and P22.

You can calculate the external traffic to PE2 as follows:

```
External traffic to PE2
= (sum of all Loc.int.E counters on PE1) - ( sum of the Loc.int.E counters of all neighbors
 of PE1)
= 39 gigabits per second - (21 + 0 + 0) gigabits per second
= 18 gigabits per second external traffic
```

So, PE2 recieves 18 gigabits per second external traffic from PE1.

The calculation for external traffic for routers follows a similar approach. Let's see few examples in the following demand matrix.

Table 27: Demand Matrix showing traffic transmitted from PE1 and PE4 to PE2

From/To	PE1	PE2
PE1	NA	39 - (21 + 0 +0) = 18 gigabits per second
PE4	21- (18 + 0 + 0) = 3 gigabits per second	39 - (18 + 0 + 0) = 21 gigabits per second

Usage Guidelines and Limitations

Supported Traffic Types

- IPv6 packets.
- SRv6 packets with the local SID as the top SID.
  - If the top SID is a local uN, traffic is counted against the remote locator prefix of the next SID.
  - Traffic is not counted if the top SID is a local uA.
- SRv6 VPNv4
- SRv6 VPNv6
- SRv6 INETv4
- SRv6 INETv6
- SRv6 EVPN VPWS
- SRv6 EVPN ELAN (unicast and BUM traffic)

Limitations

- Supports a minimum telemetry pull interval of 30 seconds.
- Local generated control plane packets are considered for SRv6 traffic accounting.
- Ethernet header is not considered for bytes accounting.
- Packets are not counted if the local uA is the top SID.
- 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.
- The accounting counters for L2VPN ELAN BUM and unknown unicast traffic exhibit a notable disparity. For instance, when two million packets are sent, the counters record 2.5 million packets.

  To rectify this discrepancy, you must subtract the count of the `L2 egress LAG not local` drop counter from the value attributed to the bundle interface.

  This adjustment allows for an accurate tally of packets exiting the bundle interface. Notably, no such irregularities are observed in L3VPN, L2VPN VPWs, or L2VPN known unicast traffic.

  This discrepancy persists even in scenarios where bundle members span different NPs on a single line card router.

Configure SRv6 Traffic Accounting

Before you begin ensure that you enable SRv6 and its services.

Configuration Example

To enable SRv6 traffic accounting:

```
Router#configure
Router(config)#accounting
Router(config-acct)#prefixes ipv6 mode per-prefix per-nexthop srv6-locators
Router(config-acct)#commit
```

Running Configuration

```
Router#show run
accounting
 prefixes
  ipv6
   mode per-prefix per-nexthop srv6-locators
  !
 !
!
```

Verification

Verify the Stats ID allocated for remote locator. The following example shows the SRv6 locator ID and the stats ID allocated for the prefixes with the locator ID.

```
Router#show route ipv6  fccc:cc00:1:: detail

Routing entry for fccc:cc00:1::/48
  Known via "isis 100", distance 115, metric 101, SRv6-locator, type level-1  <=========
locator flag
  Installed Jun  1 11:59:10.941 for 00:00:04
  Routing Descriptor Blocks
    fe80::1, from 1::1, via Bundle-Ether1201, Protected, ECMP-Backup (Local-LFA)
      Route metric is 101
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:2       Path ref count:1
      NHID: 0x2001b (Ref: 79)
      Stats-NHID: 0x2001c (Ref: 6)
      Backup path id:1
    fe80::1, from 1::1, via TenGigE0/1/0/5/2, Protected, ECMP-Backup (Local-LFA)
      Route metric is 101
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      Path id:1       Path ref count:1
      NHID: 0x2001a (Ref: 79)
     Stats-NHID: 0x2001d (Ref: 6)    <=========  Stats-NHID is allocated for prefixes with
 locator flag
      Backup path id:2
  Route version is 0x68 (104)
  No local label
  IP Precedence: Not Set
```

```
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_NON_RECURSIVE_LOW (8) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 2, Download Version 39779
  No advertising protos.
```

Configuring Telemetry Data

Configure the sensory path to retrieve the accounting data using telemetry:

```
Router#configure
Router(config)#grpc
Router(config-grpc)#port 57400
Router(config-grpc)#no-tls
Router(config-grpc)#commit
Router(config-grpc)#exit
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group s1
Router(config-model-driven-snsr-grp)#sensor-path
Cisco-IOS-XR-fib-common-oper:cef-accounting/vrfs/vrf[vrf-name='default']/af$
Router(config-model-driven-snsr-grp)#exit
Router(config-model-driven)#subscription sub1
Router(config-model-driven-subs)#sensor-group-id s1 sample-interval 30000
Router(config-model-driven-subs)#commit
Router(config-model-driven-subs)#root
Router(config)#exit
Router#
```

Running Configuration for Configuring Telemetry Data

The following shows the show running configuration:

```
Router#show run
grpc
 port 57400
 no-tls
!
telemetry model-driven
 sensor-group s1
  sensor-path
Cisco-IOS-XR-fib-common-oper:cef-accounting/vrfs/vrf[vrf-name='default']/afis/afi[afi-type=ipv6]/pfx/srv6locs/srv6loc

 !
 subscription sub1
  sensor-group-id s1 sample-interval 30000
 !
!
```

Verification for Configuring Telemetry Data

Verify the counters using the telemetry data. The following example shows the accounting data with the number of packets and the bytes transmitted through the interface.

```
{
"Cisco-IOS-XR-fib-common-oper:cef-accounting": {
  "vrfs": {
   "vrf": [
    {
     "vrf-name": "default",
```

```
     "afis": {
      "afi": [
       {
        "afi-type": "ipv6",
        "pfx": {
         "srv6locs": {
          "srv6loc": [
           {
            "ipv6-address": " fccc:cc00:1::",
            "prefix-length": 48,
            "ipv6-prefix": " fccc:cc00:1::",
            "ipv6-prefix-length": 48,
            "accounting-information": [
             {
              "number-of-tx-packets": "1500000",           <=========  Accounting data
              "number-of-tx-bytes": "378000000",          <=========  Accounting data
              "path-index": 0,
              "outgoing-interface": "Bundle-Ether1201",
              "nexthop-addr": "fe80::2/128"
             },
             {
              "number-of-tx-packets": "1000000",           <=========  Accounting data
              "number-of-tx-bytes": "252000000",          <=========  Accounting data
              "path-index": 1,
              "outgoing-interface": "TenGigE0/0/0/22",
              "nexthop-addr": "fe80::2/128"
             }
            ],
            "total-number-of-packets-switched": "2500000",
            "total-number-of-bytes-switched": "630000000"
           }
          ]
         }
        }
       ]
      }
     }
    ]
   }
}
}
```

Run **show cef ipv6 accounting** command to display the packets per bytes:

```
Router#show 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
 21ac packets count
```

Path Maximum Transmission Unit (MTU) Discovery for SRv6 Encapsulated Packets

Table 28: Feature History Table

Feature Name	Release Information	Feature Description
Path MTU Discovery for SRv6 Packets on Ingress Provider Edge (PE) Routers and Egress (PE) Router	Release 24.2.1	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. This feature is enabled by default.

Earlier, routers did not account for the SRv6 encapsulated packets while checking the MTU of a link along a given data path in the egress core interface. When the path MTU of a link along a given data path was not large enough to accommodate the size of the encapsulated packets from a source, the router silently dropped the packets without notifying the source.

With this configuration, the ingress PE router supports Path MTU discovery for SRv6 encapsulated packets. The router does not drop the packets along a given data path without notifying the source. The router sends an ICMP type 3 or type 2 error message for IPv4 or IPv6 links respectively. The configuration enables the source to learn to use a smaller MTU for packets sent to a destination.

Note Starting from Cisco IOS XR Release 24.2.1, Path MTU discovery for SRv6 encapsulated packets support Ingress PE routers, Egress PE routers, and Provider Core Router or Transit nodes.

For example, the maximum allowed MTU for an IPv4 link is 1500 bytes. Consider a source that sends an IPv4 packet of size 1480 bytes with an SRv6 encapsulation of 40 bytes. The overall IPv4 packet size is increased to 1520 bytes, which is greater than the maximum MTU allowed on the IPv4 link. In this case, the router sends an ICMP Type 3 error message to the source to request the packet originator to adjust the size of the packet.

We calculate the maximum allowed MTU on IPv4 and IPv6 links using the following formula:

Maximum MTU = Egress Interface MTU + SRv6 Encapsulation Size (maximum 64 bytes) + size of L2 Header

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

- The SRv6 uSID (F3216) format supports the feature.
- Does not support L2 services over SRv6 (L2VPN).
- When ECMP paths have different MTU sizes configured for each of these paths, then the ICMP P2B generation is not guaranteed. This is due to the difference of the software and hardware load balance hashing algorithm. The MTU exceeded punt packet points to a different path due to this mismatch in the software and hardware hashing. This causes the P2B generation with a wrong MTU value or it gets re-injected to a different interface path.
- This feature is not supported on BVI interfaces.
- **Ingress**
  - On the Cisco ASR 9000 with IOS XR 64 bit, you can set the Ingress Interface MTU to a maximum of 9436 bytes for a single SRv6 encapsulation.
  - The SRv6 Full-length SID format does not support Path MTU discovery.



Note Starting from Cisco IOS XR Release 24.2.1, Ingress and Provider Core Router or Transit nodes with IPv6 are supported.

  - SRv6 encapsulation supports the following scenarios:
    - IPv4/IPv6 over SRv6
    - SRv6-TE
    - H insert
    - TI-LFA for Single Carrier and Multi Carrier
- **Egress**
  - Starting from Cisco IOS XR Release 24.2.1, you can configure this feature on the Egress PE router.
  - Only supported for the following functions:
    - Decapsulation and specific IPv4 table lookup (DT4), Decapsulation and specific IPv6 table lookup (DT6), and Decapsulation and specific IP table lookup (DT46)

- Decapsulation and IPv4 cross-connect (DX4) and Decapsulation and IPv6 cross-connect (DX6)

- Does not support Decapsulation and L2 table lookup (DT2) and Decapsulation and L2 cross-connect (DX2) functions (because L2 payload is not a use-case).

- **Provider Core Router or Transit Node**
  - Starting from Cisco IOS XR Release 24.2.1, path MTU discovery supports P node that is in IPv6-only.
  - Path MTU discovery now supports SRv6 enabled endpoint is a Provider Core Router or a Transit node. This is possible when the destination address of the packet is a SID.

- **Provider Core Router or Transit Node**
  - Supports IPv6 forwarding without SID termination or encapsulation.
  - It supports IPv6-only forwarding
  - The P node requires basic configuration that includes the SRv6 locator.
  - Supports single IPv6 header encapsulation.
  - Supports this packet format: IPv6 (Encap) > with and without SRH (at most single SRH) > IPv4 or IPv6 Header (Inner Header) > Payload
  - MTU adjustments:
    - IPv6 forwarding
    - SRv6 TILFA Encap and IPv6 forwarding

VRF-to-VRF route leaking in SRv6 core

VRF-to-VRF route leaking is an SRv6 core feature that allows communication between separate VRFs by sharing specific routes while keeping others isolated. This is achieved by configuring import and export route targets in each VRF, ensuring that only the selected routes are exchanged.

Table 29: Feature History Table

Feature Name	Release Information	Feature Description
VRF-to-VRF route leaking in SRv6 core	Release 24.4.1	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.

Route leaking between VRFs in an SRv6 core network extends the MPLS-based VRF route leaking functionalities to SRv6, by taking advantage of the SRv6 flexibility for path selection and optimization. You can configure the destination VRF to send leaked routes to a remote PE or RR across an SRv6 core. The feature enables VRF-to-VRF communication 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.

SRv6 VRF-to-VRF route leaking workflow

The different steps in the VRF-to-VRF route leaking workflow are explained in detail:

- Route import: A VRF imports a prefix from another VRF, which belongs to another service and creates a route leak between them.
- SID allocation: The destination VRF allocates a unique SRv6 SID to the imported prefix. The SID ensures that the traffic to the imported prefix is correctly routed within the SRv6 core. The SID allocation type is either per-VRF or per-VRF-46 depending on the destination VRF configuration.
- Prefix advertisement: The destination VRF advertises the imported prefix, along with the associated SRv6 SID to remote neighbors in the SRv6 network through BGP VPNv4 or VPNv6 or EVPN RT5.

- Routing within the SRv6 core: The remote neighbors now have the information that is required to route traffic to the imported prefix using the SRv6 SID. The SRv6 SID allows for efficient routing and traffic engineering, ensuring the traffic reaches the correct VRF and destination.

Usage guidelines and limitations for VRF-to-VRF route leaking in SRv6

The usage guidelines and limitations that are listed apply:

- 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.

Configure VRF-to-VRF route leaking in SRv6

Before you begin

Enable Segment Routing over IPv6 under BGP in the source and destination VRFs.

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
```

BGP SID locator tracking for SRv6 services

BGP SID locator tracking is an SRv6 feature that

- allows BGP to monitor the reachability and metric of SRv6 SID locators in the RIB, in addition to validating the BGP next-hop, rather than relying solely on the next-hop
- automatically tracks these locators and reports locator- and next-hop–related reachability or metric changes to the BGP routing process as they are updated in the RIB, and
- improves overall BGP convergence and ensures consistent control-plane and data-plane behavior in SRv6 deployments.

Table 30: Feature History Table

Feature Name	Release Information	Feature Description
BGP SID locator tracking for SRv6 services	Release 26.1.1	You can now enhance SRv6 path selection by extending BGP next-hop tracking to monitor the reachability and metric of SRv6 SID locators. BGP tracks locators for both Layer 2 and Layer 3 services and updates path selection when a locator becomes unreachable, even if the next-hop remains reachable. The feature introduces these changes: CLI: • **nexthop validation use-locator**

Challenges with traditional BGP next-hop tracking

Traditional BGP next-hop tracking monitors only the reachability and metric of the BGP next-hop. In SRv6 networks, however, forwarding decisions rely on the SID locator, which may use different IGP paths or experience independent routing changes.

This disconnect can lead to several critical issues:

- Traffic blackholing: BGP may consider a path valid even if the associated SID locator is unreachable.
- Suboptimal path selection: BGP might incorrectly select the best path when next-hop metrics do not accurately reflect the locator's metrics.
- Reduced performance: These inconsistencies can result in slower convergence, routing metric mismatches, and loss of connectivity.

Enhancing BGP path selection with SID locator tracking in SRv6 networks

The BGP SID locator tracking feature addresses these challenges by:

- Extending BGP's existing RIB registration mechanism to track and monitor the reachability and metrics of SRv6 SID locators, in addition to traditional next-hop tracking.
- Incorporating locator information into BGP path evaluation and attribute propagation.

With these enhancements, BGP can make more accurate best-path selections and respond faster to network changes that affect SRv6 SID locators.

By default, BGP SID locator tracking is disabled, so BGP relies on traditional next-hop tracking. To enable this feature and ensure that BGP monitors the reachability and metrics of SRv6 SID locators in the RIB, configure the `nexthop validation use-locator` command.

You must enable this feature on all devices of a network to avoid the risk of routing loop. This feature supports both MicroSID and Full-length SID.

SID locator tracking

Table 31: SID Locator Tracking — Behavior Comparison

BGP behavior	Without SID locator tracking feature	With SID locator tracking feature
What BGP tracks	• BGP next-hop reachability • BGP next-hop metric	• BGP next-hop • SRv6 SID locator reachability • SRv6 SID locator metric
RIB registration	BGP registers with RIB for BGP next-hop tracking	BGP extends RIB registration to include SRv6 SID locator.
SID resolvability check	One-time SID resolvability check during path validation.	Continuous tracking of SID locator reachability via RIB notifications.

Benefits of BGP SID locator tracking for SRv6 services

- Faster convergence: Accelerates convergence for SRv6 Layer 2 and Layer 3 services.
- Prevents traffic loss: Avoids loss of traffic when SID locators become unreachable.
- Control and data plane alignment: Ensures consistent reachability and metric information across control and data planes.
- Improved best-path selection: Uses locator RIB metrics for more accurate path selection.
- Better attribute propagation: Supports Accumulated IGP (AIGP) and Multi-Exit Discriminator (MED) based on SID locator metrics.

Restrictions for BGP SID locator tracking for SRv6 services

For BGP SID locator tracking with SRv6 services, follow these restrictions.

Configuration requirements

- You must manually configure BGP SID locator tracking because it is disabled by default.

If SID structure sub-TLVs are missing, BGP cannot extract locator information and will use traditional next-hop tracking (NHT) behavior.

- You must enable this feature on all devices in the network to avoid the risk of a routing loop

Operational restrictions

- Do not use BGP SID locator tracking when SRv6 locator summarization is enabled at ABR nodes. Summarization can hide actual locator reachability and may cause the locator and BGP next-hop RIB metric to appear worse than the BGP next-hop metric

Feature limitations

- This feature does not provide enhancements to Optimal Route Reflection (ORR).

How BGP SID locator tracking for SRv6 services works

BGP SID locator tracking for SRv6 services ensures correct path selection and convergence when SRv6 SID locators follow different IGP paths than the BGP next hop. By tracking locator reachability and metrics in the RIB, BGP can maintain accurate and consistent SRv6 service routing.

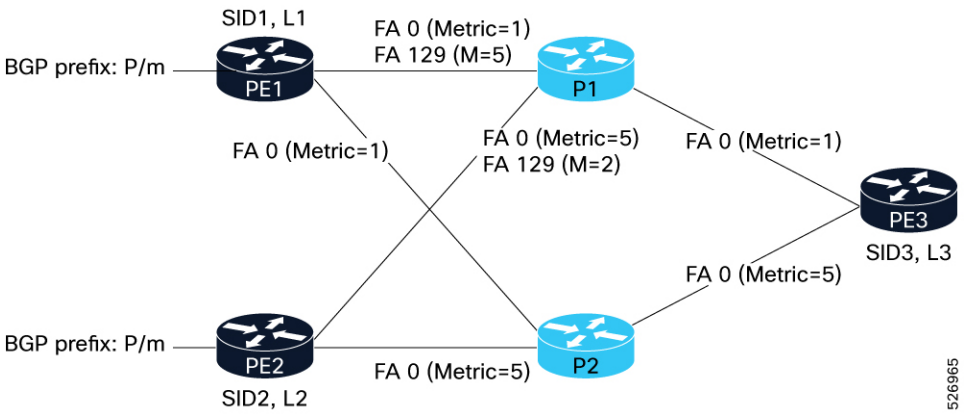
Summary

The process of BGP SID locator tracking for SRv6 services begins with Provider Edge (PE) routers allocating and advertising SRv6 SIDs for local services through BGP. These BGP updates, containing SRv6 SIDs and their locators, are then propagated by Route Reflectors (RRs). IS-IS domains distribute locator reachability information throughout the network. Each PE router actively tracks the reachability of remote SRv6 locators, enabling BGP to automatically withdraw or update affected SID routes in response to topology changes. This mechanism ensures rapid service rerouting and fast convergence, thereby maintaining service continuity and minimizing traffic disruption.

Workflow

These stages describe how BGP SID locator tracking works:

Figure 19: Topology for BGP SID locator tracking



1. SID advertisement: Each PE router allocates SRv6 SIDs for locally hosted services and associates them with IPv6 locators. These SIDs are advertised using BGP so that remote PEs can learn the SRv6 service endpoints.

   For example, PE1 allocates `SID1` with locator `L1` and advertises it via BGP.

2. Locator reachability distribution: IGP domains propagate reachability information for SRv6 locators, ensuring all nodes in the AS 100 backbone can forward packets based on the SIDs.

3. RIB-Based Locator Tracking: Each PE registers with the RIB to track the reachability and metric of SRv6 locators from the received SID advertised by remote PEs, in addition to traditional BGP next-hop tracking.

   When the RIB reports that a locator becomes unreachable or its metric changes due to a link or node failure, BGP automatically re-evaluates path selection and withdraws or updates the SRv6 service routes as needed.

   For example, PE3 receives the BGP route for `SID1/L1` and checks in its RIB if `L1` is reachable and to determine its current IGP cost.

   If a topology change increases the cost or makes L1 unreachable, the RIB notifies BGP, which then updates or withdraws the corresponding SRv6 service routes.

   Table 32: BGP best path calculation example

Underlay FlexAlgo	PE router path view from PE3	Metric calculation	BGP bestpath
FA 0	PE1-P1-PE3	1 + 1 = 2	For FA0, PE1 is the best path.
FA 129	PE1-P1-PE3	1+ 5 = 6	
FA 0	PE2-P1-PE3	1 + 5 = 6	
FA 129	PE2-P1-PE3	1 + 2 = 3	For FA129, PE2 is the best path.

4. Service continuity and fast convergence: When topology changes occur, IGP updates locator reachability, and RIB notifies and BGP promptly withdraws or updates SRv6 service routes. This mechanism ensures services are rerouted or restored quickly, minimizing traffic disruption.

Configure BGP SID locator tracking for SRv6 services

Use this task to enable and validate BGP SID locator tracking for SRv6 services.

Procedure

Step 1 Enable next-hop validation to use the locator for SRv6.

Example:

```
Router(config)#router bgp 200
Router(config-bgp)#nexthop validation use-locator
```

Step 2 Verify that the BGP SID locator tracking is active and providing reachability or metric information.

Example:

```
Router#show bgp vrf CE2500 192.0.1.1

BGP routing table entry for 192.0.1.1, Route Distinguisher: 102:2500
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker             472872       472872
Paths: (1 available, best #1)
  Advertised to CE peers (in unique update groups):
    18::1:1:1
  Path #1: Received by speaker 0
  Advertised to CE peers (in unique update groups):
    18::1:1:1
  200, (received & used)
    fccc:cc00:1202::2 (metric 30) from fccc:cc00:1401::1 (10.1.1.2)
      Received Label 0xe0040
     Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, import-candidate,
 imported
      Received Path ID 1, Local Path ID 1, version 472872
      Extended community: Color:2500 RT:62500:2500
      Originator: 10.1.1.2, Cluster list: 10.1.1.5
      PSID-Type:L3, SubTLV Count:1
       SID Locator RIB metric: 30
       SubTLV:
        T:1(Sid information), Sid:fccc:cc05:1052::(Transposed), Behavior:63, SS-TLV Count:1
         SubSubTLV:
          T:1(Sid structure):
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 101:2500


Router#show cef vrf CE2500 192.0.1.1
Thu Dec 24 03:44:02.973 IST
10.19.1.1/32, version 206690, SRv6 Headend, internal 0x5000001 0x30 (ptr 0x90c0df14) [1], 0x0 (0x0),
 0x0 (0x925f7468)
 Updated Dec 24 03:40:21.401
 Prefix Len 32, traffic index 0, precedence n/a, priority 3
  gateway array (0x8fa65a98) reference count 5025, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48441 (0x7a19a758) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Dec 23 05:05:48.103
 LDI Update time Dec 23 05:05:48.103

  Level 1 - Load distribution: 0
  [0] via fccc:cc05:1052::/128, recursive

   via fccc:cc05:1052::/128, 5 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x79ffbd34 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop fccc:cc05:1052::/128 via fccc:cc05:1052::/48
    SRv6 H.Encaps.Red SID-list {fccc:cc05:1052:e004::}

    Load distribution: 0 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   Bundle-Ether1             fe80::28a:96ff:fe4a:20e1
```

Step 3 Confirm BGP is actively tracking the SRv6 locator in the RIB.

Example:

```
Router#show bgp nexthops fccc:cc05:1052::
Nexthop: fccc:cc05:1052::
  VRF: default
  Nexthop ID: 0x6000071, Version: 0
  Nexthop Flags: 0x10000000
  Nexthop Handle: 0x7f0b036d80e8
  Tree Nexthop Handle: 0x7f0b036d80e8

  RIB Related Information:
  Firsthop interface handle 0x00000000
    Gateway TBL Id: 0xe0800000    Gateway Flags: 0x00000080
    Gateway Handle: 0x7f0b7f9e7b68
    Gateway: unreachable, non-Connected route, prefix length 0
    Resolving Route: fccc:cc05:1052::/48 (isis 2)
    SRv6 locator, prefix len: 48
    Paths: 0
    RIB Nexthop Route Type: ISIS level-2
    RIB Nexthop Path Type: any
    RIB Nexthop ID: 0x0
    Nexthop sync slot: 22
    Status: [Unreachable]
    Metric: 4294967295
    Admin Distance: 4294967295, 115 (RIB)
    ORR afi bits: 0x0
    Registration: Synchronous, Completed: 1d00h
    Events: Critical (5)/Non-critical (0)
    Last Received: 00:50:59 (Critical)
    Last gw update: (Crit-notif) 00:50:59(rib)
    Reference Count: 0

    Reachable Notifications:           3 (last at Dec 24 03:00:35.062)
    Unreachable Notifications:         3 (last at Dec 24 03:01:39.121)
    Metric Increase Notifications:     0
    Metric Decrease Notifications:     0
    Nexthop find:                     21
    Most Recent Events:
      Time                 EventType    Metric   RIBAdminDist    Ifhandle     RouteType
PathType
      Dec 24 02:52:21.218  Unreachable  -
      Dec 24 02:53:05.838  Reachable    210      115             0x40032a0    ISIS level-2
any
      Dec 24 02:54:02.226  Unreachable  -
      Dec 24 03:00:35.062  Reachable    210      115             0x40032a0    ISIS level-2
any
      Dec 24 03:01:39.121  Unreachable  -
```

What to do next

You can use these commands to verify the feature or troubleshoot:

- `show bgp <afi safi> rd <rd> <nlri>`
- `show bgp vrf <vrfname> ipv4/ipv6 <prefix>`
- `show bgp ipv4/ipv6 unicast <prefix>`
- `show bgp nexthops <locator>`
- `show cef vrf <vrfname> ipv4/ipv6 <prefix>`

SRv6-TE explicit BSIDs and remote steering

SRv6-TE explicit BSIDs and remote steering is a core SRv6 feature that

- configures a persistent and deterministic binding SID for an SRv6-TE policy,
- enables Remote BSID Steering, allowing the router to match incoming traffic with a destination address equal to a local policy BSID and steer it into the associated SRv6 TE policy, and
- ensures that the binding SID remains consistent across reloads and policy reinstalls.

Table 33: Feature History Table

Feature Name	Release	Description
SRv6-TE explicit BSIDs and remote steering	Release 26.1.1	

Feature Name	Release	Description

		You can now manually configure a deterministic and persistent binding SID, also known as an explicit BSID, for an SRv6-TE policy. The router also supports remote BSID steering, which allows it to recognize traffic whose destination address matches a local policy BSID and steer the traffic directly into the associated SRv6-TE policy.
		
		Previously, the router allocated SRv6-TE policy binding SIDs dynamically and could change the BSID during reloads or policy reinstallation. This behavior limited deterministic policy stitching and integration with external controllers.
		
		The feature introduces these changes:
		
		CLI:
		
		• **show segment-routing traffic-eng database explicit-bsid srv6 [summary | detail | private]**
		
		• **clear segment-routing srv6 sid** <*address*>
		
		• The **enforce** keyword is introduced as an explicit option in SR-TE SRv6 submode.
		
		• The following show commands are updated to display the binding SID behavior:
		
		  • **show segment-routing traffic-eng policy**
		
		  • **show segment-routing srv6 static-endpoint**
		
		YANG Data Models:
		
		• `Cisco-IOS-XR-infra-xtc-agent-oper.yang`

Feature Name	Release	Description
		• Cisco-IOS-XR-segment-routing-srv6-cfg.yang (see GitHub, YANG Data Models Navigator)

SRv6-TE binding SID enhancements

The feature enhances SRv6-TE by enabling deterministic binding SID allocation and allowing the router to steer traffic into policies from remote nodes.

The feature consists of two primary components:

- Explicit BSID (Static BSID) — The router allows you to manually configure a binding SID for an SRv6-TE policy. Previously, the router allocated BSIDs dynamically. Manual configuration ensures that the BSID remains persistent and deterministic, which is critical when an upstream policy uses BSID as a segment.
- Remote steering — The router can receive a packet whose destination address matches a local SRv6-TE policy BSID and steer the traffic directly into the corresponding policy.

Benefits of SRv6-TE explicit BSIDs and remote steering

- Hierarchical networking: The router supports policy stitching by steering packets into a single policy, and then forwards them to a midpoint where another policy, identified by its BSID, continues forwarding.
- Scalability: The feature enables the router to support complex topologies such as transit policies and static network anchors by allowing hosts to use a specific IP address, the BSID, for transport.
- Service steering: The router enables ABF, PBR, and FlowSpec to steer traffic over SRv6-TE policies.

Guidelines and limitations for SRv6-TE explicit BSIDs and remote steering

Follow these guidelines and take note of the limitations when configuring SRv6-TE explicit BSIDs and remote steering.

Guidelines

- The router supports 256 explicit BSIDs by default. Increase the explicit SID range in the SID Manager to support more SIDs.
- Use the uSID F3216 format for explicit BSID and remote steering deployments.

Limitations

- In Release 26.1.1, the router does not fully support Path MTU for remote steering.
- The `binding-sid` command syntax is not backward compatible. Review existing configurations before upgrading.

Configure SRv6-TE explicit BSIDs and remote steering

You can configure SRv6-TE explicit BSIDs and remote steering using these steps.

Procedure

Step 1 Run the **segment-routing srv6** command to configure a static BSID in the SID Manager.

Example:

Define the SID in the central SID Manager database and associate it with the specific policy color and endpoint before the SR-TE policy uses an explicit BSID.

```
Router(config)# segment-routing
Router(config-sr)# srv6
Router(config-sr-srv6)# static
Router(config-sr-srv6-static)# endpoint
Router(config-sr-srv6-static-endpoint)# sid fccc:ccc1:1:fe00:: behavior end-ub6-insert-red
identification policy color 100 endpoint 1:1::11
Router(config-sr-srv6-static-endpoint)# commit
```

Step 2 Run the **binding-SID** command for the SR-TE policy to use an explicit BSID.

The command ensures that if you delete the explicitly configured SID, or if it conflicts with another allocation, the router keeps the policy in a down state and does not allocate a dynamic SID.

Example:

```
Router(config)# segment-routing
Router(config-sr)# traffic-engineering
Router(config-sr-te)# policy MY_SRV6_POLICY
Router(config-sr-te-policy)# srv6
Router(config-sr-te-policy-srv6)# binding-sid explicit enforce
Router(config-sr-te-policy-srv6)# commit
```

Step 3 Run the **install-endpoint-node-sid** command to enable remote steering for the node.

The command ensures that the last uSID, the BSID, is not suppressed so that the node can correctly steer traffic received from remote nodes. By default, SR-TE suppresses this behavior.

Example:

```
Router(config-sr-te-policy)# srv6
Router(config-sr-te-policy-srv6)# install-endpoint-node-sid
Router(config-sr-te-policy-srv6)# commit
```

Step 4 Run the **show segment-routing traffic-eng database explicit-bsid srv6** command to verify the summary or detailed information about the explicit BSIDs.

Example:

```
Router# show segment-routing traffic-eng database explicit-bsid srv6
Tue Feb 17 05:02:24.088 +00

               Binding SID                   Behavior      Policy (Color,Endpoint)
-------------------------- -------------------------- ----------------------------
      fc20:253f:11:fe05::    uB6 (PSP/USD Insert.Red)                    50,22::22

Router# show segment-routing traffic-eng database explicit-bsid srv6 private
Tue Feb 17 05:02:33.031 +00
```

```
SRv6-TE Redistributed BSID database:
------------------------------------

Binding SID: fc20:253f:11:fe05::/64 (Last update: Feb 17 05:00:57.924)
  Behavior: uB6 (PSP/USD Insert.Red) (99)
  Policy identifier:
    Color: 50 Endpoint: 22::22
  Locator ID: 1
  Conflict: No
  Stale: No
  Configured: Yes
  Event history (oldest first):
    Time                    Event
    Feb 17 05:00:57.924     Added (SID Mgr configured: true)
```

```
Router# show segment-routing traffic-eng database explicit-bsid srv6 detail
Tue Feb 17 05:02:29.593 +00

SRv6-TE Redistributed BSID database:
------------------------------------

Binding SID: fc20:253f:11:fe05::/64 (Last update: Feb 17 05:00:57.924)
  Behavior: uB6 (PSP/USD Insert.Red) (99)
  Policy identifier:
    Color: 50 Endpoint: 22::22
  Locator ID: 1
  Conflict: No
```

Step 5 Run the **clear segment-routing srv6 sid** <*address*> command to manually clear an explicit SRv6 SID in case of route conflicts.

Example:

```
Router# clear segment-routing srv6 sid fc20:253f:11:fe05::
Tue Feb 17 05:04:09.407 +00
Router#
```

Step 6 Run the **show segment-routing traffic-eng policy color** command to display the detailed operational status, configuration, and attributes of all SR-TE policies associated with a specific route color.

In Release 26.1.1, the command is enhanced to include details to verify if a manually assigned BSID is allocated and programmed.

Example:

```
Router(config-sr-te-policy-srv6-expl-enforce)# show segment-routing traffic-eng policy color 105
Tue Feb 17 05:33:07.117 +00

SR-TE policy database
---------------------

Color: 105, End-point: 22::22
  Name: srte_c_105_ep_22::22
  Status:
    Admin: up  Operational: up for 00:00:37 (since Feb 17 05:32:29.986)
  Candidate-paths:
    Preference: 10 (configuration) (active)
      Name: test_policy
      Requested BSID: explicit
      Constraints:
        Protection Type: protected-preferred
        Maximum SID Depth: 13
      Explicit: segment-list TC1 (valid)
        Weight: 1, Metric Type: TE (IANA PCEP/IGP: 2/2)
          SID[0]: fc20:253f:1::/48
```

```
                Format: f3216
                LBL:32 LNL:16 FL:0 AL:80
          SID[1]: fc20:253f:2::/48
                Format: f3216
                LBL:32 LNL:16 FL:0 AL:80
      SRv6 Information:
        Locator: loc1
        Binding SID requested: fc20:253f:11:fe05::
        Binding SID behavior: uB6 (PSP/USD Insert.Red)
        Explicit binding SID enforced: Yes
        Install Final Node SID: No
  Attributes:
    Binding SID: fc20:253f:11:fe05::
    Forward Class: Not Configured
    Steering services disabled: no
    Steering BGP disabled: no
    IPv6 caps enable: yes
    Invalidation drop enabled: no
    Max Install Standby Candidate Paths: 0
    Path Type: SRV6
```

Step 7 Run the **show segment-routing srv6 static-endpoint** command to verify the SID Manager view of the explicit BSIDs.

Example:

```
Router# show segment-routing srv6 static-endpoint
Tue Feb 17 05:24:01.798 +00
SID/Prefix                 Behavior          Identification                                 State

-------------------------  ----------------  ---------------------------------------------
-----------------------------------------
fc20:253f:11:fe05::/64     uB6 (PSP/USD Ins  'srte_c_50_ep_22::22' (50, 22::22)
Allocated - Locator Up
```

SRv6 Flow Label entropy via source address

SRv6 Flow Label entropy via source address is a core segment routing feature that

- enables consistent end-to-end load balancing in networks when intermediate nodes are unable to utilize the standard IPv6 Flow Label for hash calculations
- serves as a critical workaround for service providers and cloud-based deployments, and
- provides flexible configuration options allowing for platform-specific optimization.

SRv6 entropy encoding in IPv6 source address

In an SRv6 network, the Flow Label in the IPv6 header provides built-in entropy and enables efficient per-flow load balancing. the core nodes use this entropy to perform hashing without accessing all flows arriving at the PE.

Some IPv6 backbone nodes, especially in cloud environments do not use the Flow Label for load balancing, which disrupts end-to-end traffic distribution. To address this, the encapsulating node encodes entropy into the IPv6 source address during SRv6 H.Encaps. It generates a flow-aware source address by combining the Flow Label with the original source address, enabling all network devices to use standard IP-based hashing and ensures consistent load balancing across the network.

Benefits of Flow Label entropy via source address

SRv6 Flow Label entropy using source address includes the listed benefits.

- Ensures effective ECMP distribution across all network segments, including cloud underlays that ignore the Flow Label.
- Prevents traffic polarization and maximizes use of available bandwidth.
- Enables reliable SRv6 deployments over public cloud infrastructures.
- Eliminates reliance on intermediate nodes supporting Flow Label-based hashing.
- Reduces congestion and improves overall application and service reliability.

Usage guidelines of Flow Label entropy via source address

These guidelines and limitations apply when you configure Flow Label entropy via source address:

- This feature supports both SRv6 full-length SID and micro-SIDs.
- The Flow Label uses a fixed size of 20 bits.
- Use either `right-justified` or `relative-offset` mode, with 16, 24, or 32 bits, for Flow Label positioning.
- The feature supports configurable size and offset for flexible entropy encoding.
- Only one source address (explicit or locator-derived) can be configured at a time.
- If you configure an invalid source address or if the locator is down, the feature programs a null source address. It does not revert to the default source address.
- This feature supports only Cisco ASR 9000 Series 5th Generation High-Density Multi-Rate Line Cards.

How SRv6 Flow Label entropy via source address works

Summary

Embedding SRv6 Flow Label entropy into the IPv6 source address allows core network nodes to leverage source address diversity for effective load balancing, even when Flow Label–based entropy is not available.

Key components of the process include:

- Ingress PE device: Receives the original packet and prepares it for SRv6 encapsulation.
- Hardware hash function: Calculates the entropy value (Flow Label) based on packet headers.
- Core node: Performs load-balancing decisions using the entropy in the modified source address.
- Egress PE: Removes the SRv6 header and delivers the original payload.

Workflow

These stages describe how SRv6 Flow Label entropy via source address works:

1. The ingress PE receives the data packet.
2. The hardware computes a hash from the packet headers and generates the Flow Label.
3. The PE applies the SRv6 H.Encaps operation and adds the SRv6 header.
4. The ingress PE uses the configured add-entropy parameters to encode Flow Label bits into the IPv6 source address at the required offset.
5. The ingress PE forwards the packet into the core network. The core nodes use the modified source address to perform load-balancing hashes.
6. The egress PE terminates the SRv6 header and forwards the original payload to the destination.

Configure Flow Label entropy via source address

The configuration enables the router to use a specified locator as the source address. It injects Flow Label bits into this address to enhance load balancing performance in the network.

Follow these steps to configure SRv6 Flow Label entropy via source address.

Before you begin

Ensure that the SRv6 locator is configured and operational.

Procedure

Step 1 Run the **segment-routing srv6 encapsulation** command to configure the encapsulation parameters for SRv6 traffic.

Example:

```
Router(config)#segment-routing
Router(config-sr)#srv6
Router(config-srv6)#encapsulation
```

Step 2 Configure the router to encode Flow Label–based entropy into the IPv6 source address using a locator.

- Relative-offset positioning.

  Use this mode to embed Flow Label entropy at the specified offset within the base source address.

  The example configuration uses locator0 as the base source address and embeds Flow Label entropy at a 16-bit offset to improve load balancing.

  ```
  Router(config-srv6-encap)#source-address locator locator0 add-entropy flow-label position
  relative-offset 16
  Router(config-srv6-encap)#commit
  ```

- Right-justified positioning.

  Use this mode to place the Flow Label entropy at the rightmost end of the IPv6 source address.

  The example configuration uses locator0 as the base source address.

  ```
  Router(config-srv6-encap)#source-address locator locator0 add-entropy flow-label position
  right-justified
  Router(config-srv6-encap)#commit
  ```

Step 3 Enter these show commands to verify that Flow Label–based entropy is encoded in the IPv6 source address.

Example:

```
Router#show segment-routing srv6 manager
Output received:
Sat Jan 17 03:53:03.738 EST
Parameters:
  SRv6 Enabled: Yes
  SRv6 Operational Mode:
    Micro-segment:
      SID Base Block: fccc:cc00::/24
  Encapsulation:
    Source Address:
      Default: 1::1
      Configured: fccc:cc00:1::
        Mode: Locator-Address with Add-Entropy
          Locator: 'locator0' (is up)
          Flow-label: Size 20 bits, Position relative-offset 16 bits
    Hop-Limit: Default
    Traffic-class: Propagate
  SID Formats:
    f3216 <32B/16NFA> (2)
      uSID LIB Range:
        LIB Start   : 0xe000
        ELIB Start  : 0xfe00
      uSID WLIB Range:
        EWLIB Start : 0xfff7

Router#show cef ipv6 glbl segment-routing srv6
    SRv6 Enabled
    ...
    Encap:
      SA: fccc:cc00:1::
        Add-Entropy Flowlabel: size 20 bits, offset Position relative-offset 16 bits
      Hop-Limit: Default
      Traffic-Class: Default
    ...
```