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

Table 1: Feature History Table

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

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

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

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

Segment Routing over IPv6 Overview

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

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

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

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

Figure 1: Network Program in the Packet Header
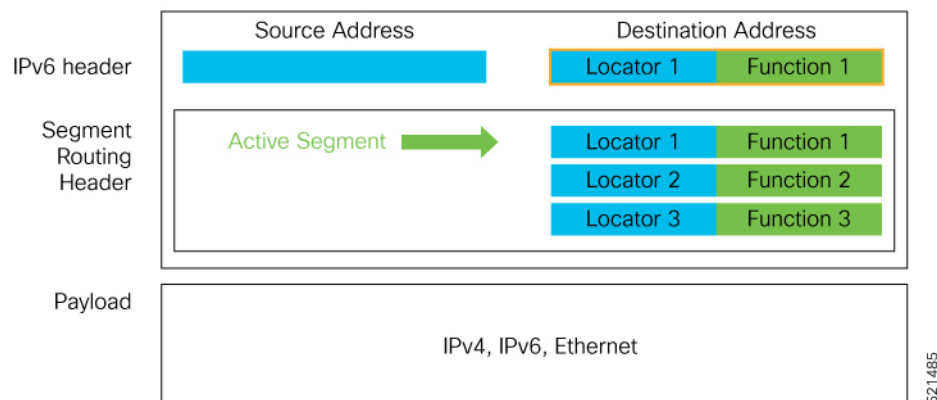
The SRv6 SRH is documented in IETF RFC IPv6 Segment Routing Header (SRH).

The SRH is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Last Entry   |     Flags     |              Tag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128-bit IPv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
```

```
                                  ...
         |                                                       |
         |                                                       |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                                                       |
         |            Segment List[n] (128-bit IPv6 address)     |
         |                                                       |
         |                                                       |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         //                                                     //
         //         Optional Type Length Value objects (variable)  //
         //                                                     //
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The following list explains the fields in SRH:

- Next header—Identifies the type of header immediately following the SRH.

- Hdr Ext Len (header extension length)—The length of the SRH in 8-octet units, not including the first 8 octets.

- Segments left—Specifies the number of route segments remaining. That means, the number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

- Last Entry—Contains the index (zero based) of the last element of the segment list.

- Flags— Contains 8 bits of flags.

- Tag—Tag a packet as part of a class or group of packets like packets sharing the same set of properties.

- Segment list—128-bit IPv6 addresses representing the *n*th segment in the segment list. The segment list encoding starts from the last segment of the SR policy (path). That means the first element of the segment list (Segment list [0]) contains the last segment of the SR policy, the second element contains the penultimate segment of the SR policy and so on.

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

- Locator: This is the first part of the SID with most significant bits and represents an address of a specific SRv6 node.

- Function: This is the portion of the SID that is local to the owner node and designates a specific SRv6 function (network instruction) that is executed locally on a particular node, specified by the locator bits.

- Args: This field is optional and represents optional arguments to the function.

The locator part can be further divided into two parts:

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

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

SRv6 Node Roles

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

- Source node—A node that can generate an IPv6 packet with an SRH (an SRv6 packet), or an ingress node that can impose an SRH on an IPv6 packet.

- Transit node—A node along the path of the SRv6 packet (IPv6 packet and SRH). The transit node does not inspect the SRH. The destination address of the IPv6 packet does not correspond to the transit node.

- Endpoint node—A node in the SRv6 domain where the SRv6 segment is terminated. The destination address of the IPv6 packet with an SRH corresponds to the end point node. The segment endpoint node executes the function bound to the SID

SRv6 Micro-Segment (uSID)

Table 2: 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 Head-End Behaviors

Table 3: Feature History Table

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

- H.Encaps.Red—H.Encaps with Reduced Encapsulation

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

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

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

SRv6 Endpoint Behaviors

SRv6 Endpoint Behaviors

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

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

- End—Endpoint function. The SRv6 instantiation of a Prefix SID [RFC8402].

- End.X—Endpoint with Layer-3 cross-connect. The SRv6 instantiation of an Adj SID [RFC8402].

- End.DX6—Endpoint with decapsulation and IPv6 cross-connect (IPv6-L3VPN - equivalent to per-CE VPN label).

- End.DX4—Endpoint with decapsulation and IPv4 cross-connect (IPv4-L3VPN - equivalent to per-CE VPN label).

- End.DT6—Endpoint with decapsulation and IPv6 table lookup (IPv6-L3VPN - equivalent to per-VRF VPN label).

- End.DT4—Endpoint with decapsulation and IPv4 table lookup (IPv4-L3VPN - equivalent to per-VRF VPN label).

- End.DT46—Endpoint with decapsulation and specific IP table lookup (IP-L3VPN - equivalent to per-VRF VPN label).

- End.DX2—Endpoint with decapsulation and L2 cross-connect (L2VPN use-case).

- End.B6.Encaps—Endpoint bound to an SRv6 policy with encapsulation. SRv6 instantiation of a Binding SID.

- End.B6.Encaps.RED—End.B6.Encaps with reduced SRH. SRv6 instantiation of a Binding SID.

SRv6 Endpoint Behavior Variants

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

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

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

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

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

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

  If Segments Left is 0, then:

  1. Update the Next Header field in the preceding header to the Next Header value of the SRH

  2. Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)

  3. Remove the SRH from the IPv6 extension header chain

  4. Proceed to process the next header in the packet

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

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

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

    1. Remove the outer IPv6 Header with all its extension headers

    2. Submit the packet to the egress IPv6 FIB lookup and transmission to the new destination

    3. Else, if the Upper-layer Header type is 4 (IPv4)

    4. Remove the outer IPv6 Header with all its extension headers

    5. Submit the packet to the egress IPv4 FIB lookup and transmission to the new destination

    6. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

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

    1. Remove the outer IPv6 Header with all its extension headers

    2. Forward the exposed IP packet to the L3 adjacency J

    3. Else, process as per Section 4.1.1 (Upper-Layer Header) of IETF RFC 8986 SRv6 Network Programming

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

SRv6 uSID Terminology

The SRv6 Network Programming is extended with the following terms:

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

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

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

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

  where:

  - uSID Block—An IPv6 prefix that defines a block of SRv6 uSIDs.

  - Active uSID—The first uSID that follows the uSID block.

  - Next uSID—The next uSID after the Active uSID.

  - Last uSID—The last uSID in the carrier before the End-of-Carrier uSID.

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

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

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

SRv6 uSID Carrier Format

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

- **uSID Block**

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

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

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

- **uSID ID**

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

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

SRv6 uSID Allocation Within a uSID Block

Table 4: Feature History Table

Feature Name	Release	Description
Wide LIB uSID Allocation for End.DT46 SRv6 SIDs	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) This feature is now supported on: • 8011-4G24Y4H-I

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

Key Concepts and Terminologies

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

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

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

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

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

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

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

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

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

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

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

Example: uSID Allocation

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

Consider the following example:

- uSID Locator Block length: 32 bits

- uSID Locator Block: FCBB:BB00::/32 (with B being a nibble value picked by operator)

- uSID length (Locator Node ID / Function ID): 16 bits

- uSID: FCBB:BB00:XYWZ::/48 (with XYWZ being variable nibbles)

A uSID FCBB:BB00:XYWZ::/48 is said to be allocated from its block (FCBB:BB00::/32).
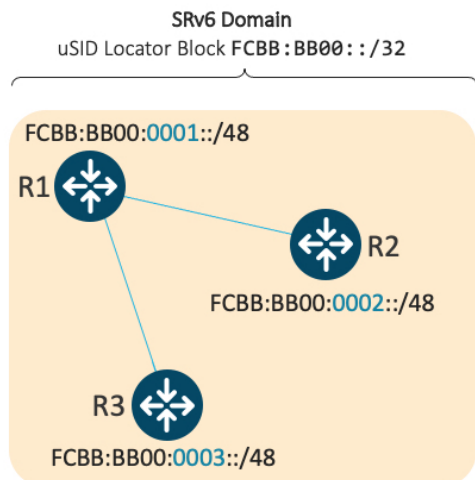
A uSID is allocated from the GIB or LIB of block FCBB:BB00::/32 depending on the value of the "**X**" nibble:

- GIB: nibble **X** from hex(0) to hex(D)

- LIB: nibble **X** hex(E) or hex(F)

XYWZ

GIB 0x0000 ... 0xDFFF	57343 IDs
LIB 0xE000 ... 0xFFFF	8192 IDs
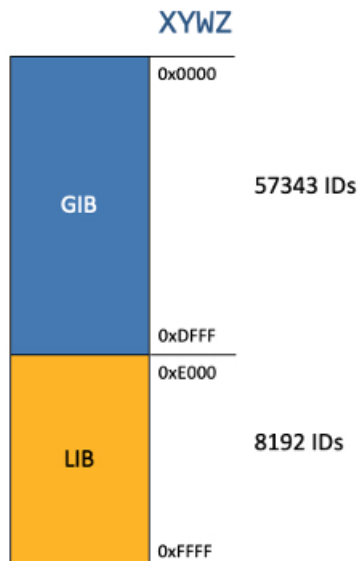
With this allocation scheme, the uSID Block **FCBB:BB00::/32** supports up to 57343 global uSIDs (routers) with each router supporting up to 8192 local uSIDs.

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

SRv6 Domain
uSID Locator Block **FCBB:BB00::/32**

FCBB:BB00:**0001**::/48
R1

R2
FCBB:BB00:**0002**::/48

R3
FCBB:BB00:**0003**::/48

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

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

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

The underlay uSIDs present on R1 are:

- FCBB:BB00:0001::/48

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

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

GIB and LIB – IOS-XR Implementation

In Cisco IOS XR Release 7.5.2 and earlier, the following functionality is supported:
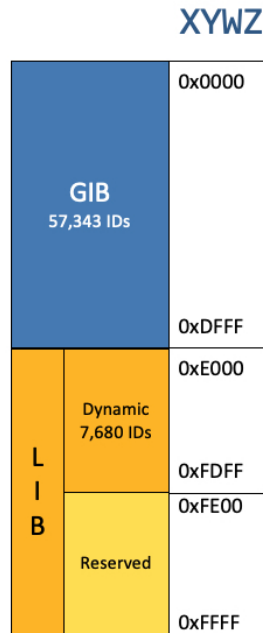
- GIB for user-assigned IDs of global segments (uNs)

- LIB for dynamically assigned IDs of local segments

  - uA end-point behavior

  - Service de-multiplexing end-point behaviors (for example, End.DT, End.DX, End.DX2)

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

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

- The range of IDs in the GIB is 0x000 to 0xDFFF.

- The range of IDs by default in the LIB is divided as follows:

  - Dynamic: 0xE000 to 0xFDFF

  - Reserved: 0xFE00 to 0xFFFF

Figure 2: GIB/LIB

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

- Configurable explicit LIB range

- Explicit LIB for user-assigned IDs of local segments

- Manual uDT46 from explicit LIB

- Wide LIB (W-LIB)

- Configurable explicit W-LIB range

- Explicit W-LIB for user-assigned IDs of local segments

- Manual uDT46 from explicit W-LIB

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

- The range of IDs in the GIB is 0x000 to 0xDFFF.

- The range of IDs by default in the LIB is divided as follows:

  - Dynamic: 0xE000 to 0xFDFF

  - Explicit: 0xFE00 to 0xFEFF

  - Reserved: 0xFF00 to 0xFFEF and 0xFFF8 to 0xFFFF

- The range of IDs by default in the W-LIB is divided as follows:

  - Reserved: 0xFFF0 to 0xFFF6

  - Explicit: 0xFFF7

Figure 3: GIB/LIB/W-LIB
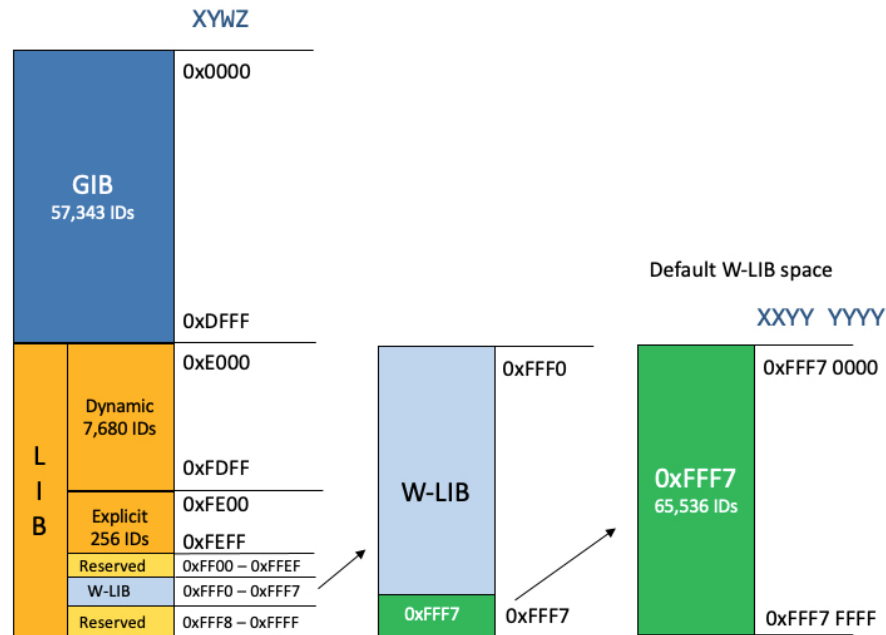
SRv6 Endpoint Behaviors Associated with uSID

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

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

- uA—A short notation for the NEXT-CSID End.X behavior with a pseudocode of shift-and-xconnect, and PSP/USD flavors

- uDT—A short notation for the NEXT-CSID End.DT behavior with the same pseudocode as End.DT4/End.DT6/End.DT46/End.DT2U/End.DT2M

- uDX—A short notation for the NEXT-CSID End.DX behavior with the same pseudocode as End.DX4/End.DX6/End.DX2

SRv6 uSID in Action - Example

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

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

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

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

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

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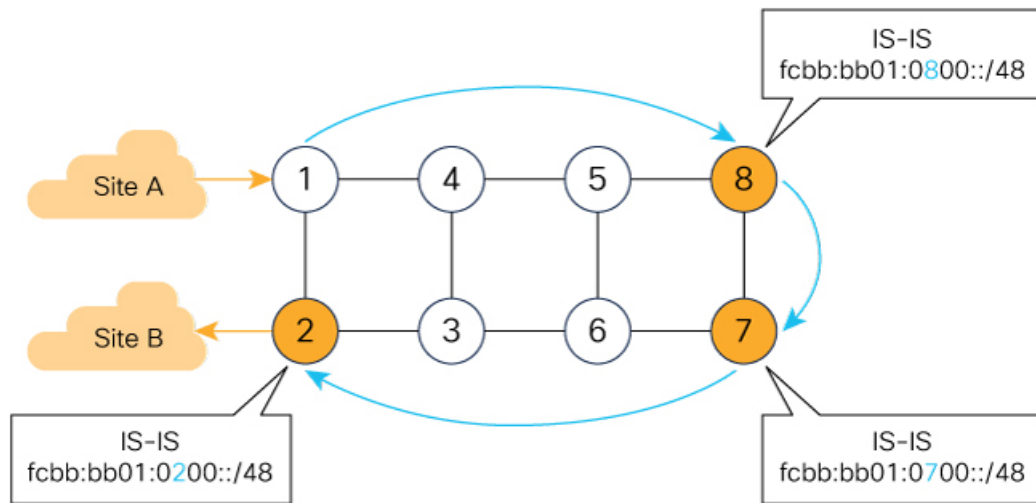
- 16-bit SRv6 ID

  For example:

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

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

The following IGP routes are advertised:

- Node 8 advertises the IGP route fcbb:bb01:**0800**::/48

- Node 7 advertises the IGP route fcbb:bb01:**0700**::/48

- Node 2 advertises the IGP route fcbb:bb01:**0200**::/48

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

- Node 1 encapsulates IPv4 packet from Site A and sends an IPv6 packet with DA = fcbb:bb01:0800:0700:0200:f001:0000:0000

- Traffic engineered path via 8 and 7 using a single 128-bit SRv6 SID

- One single micro-program in the DA is enough

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

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

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

Follow shortest path to
fcbb:bb01:0800::/48

DA = fcbb:bb01:0800:0700:0200:f001:0000:0000

521411
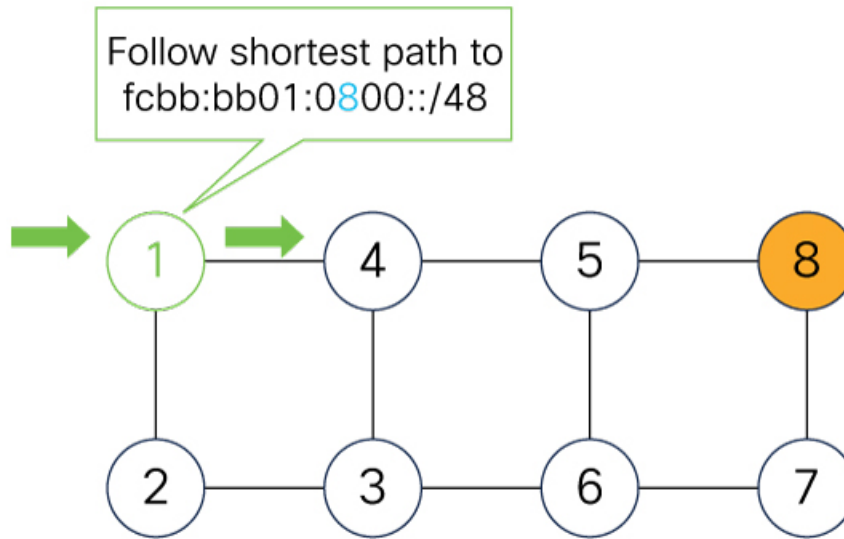
Nodes 4 and 5 simply forward the packet along the shortest path to Node 8, providing seamless deployment through classic IPv6 nodes.

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

Follow shortest path to
fcbb:bb01:0800::/48

Follow shortest path to
fcbb:bb01:0800::/48

DA = fcbb:bb01:0800:0700:0200:f001:0000:0000

521412
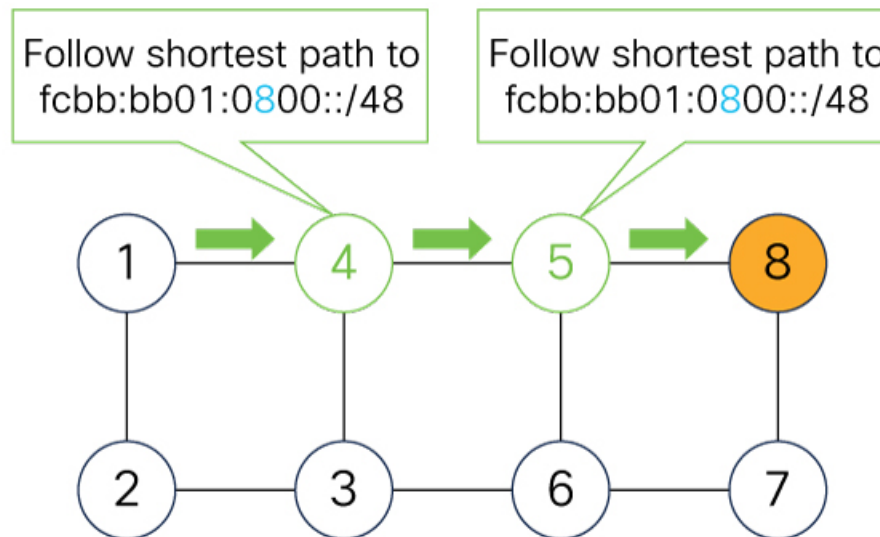
When Node 8 receives the packet, it performs SRv6 uN behavior (shift-and-lookup with PSP/USD). It removes its outer DA (0800) and advances the micro program to the next micro instruction by doing the following:

1. Pops its own uSID (0800)

2. **Shifts** the remaining DA by 16-bits to the left

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

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

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

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

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

Figure 8: Node 7: SRv6 uN Behavior (Shift and Forward)
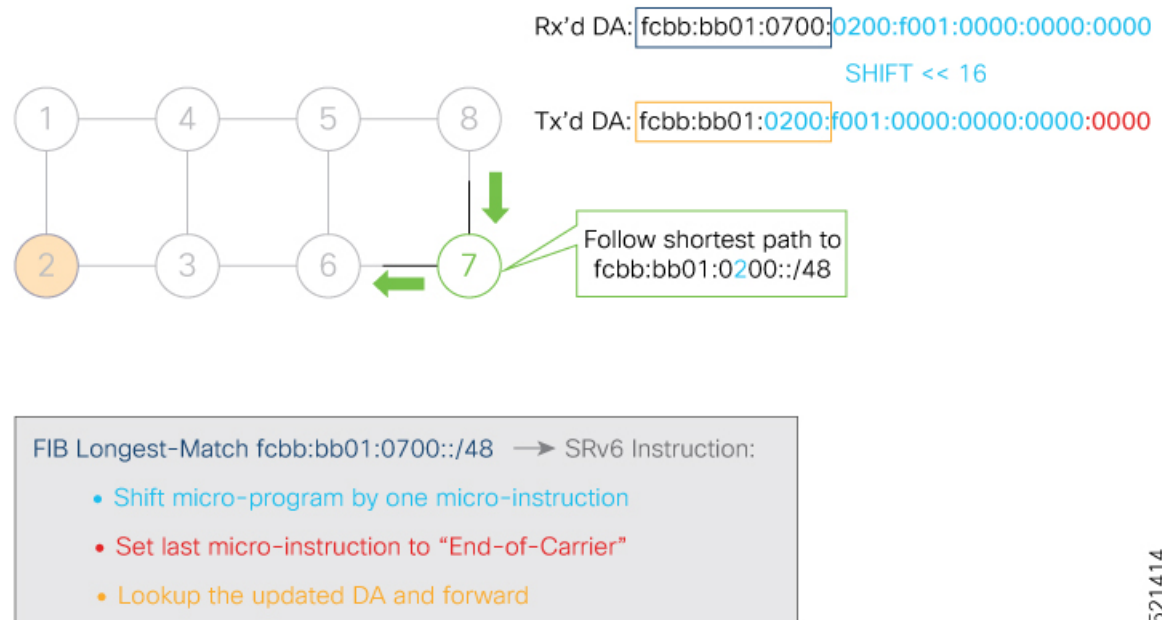
Nodes 6 and 3 simply forward the packet along the shortest path to Node 2, providing seamless deployment through classic IPv6 nodes.

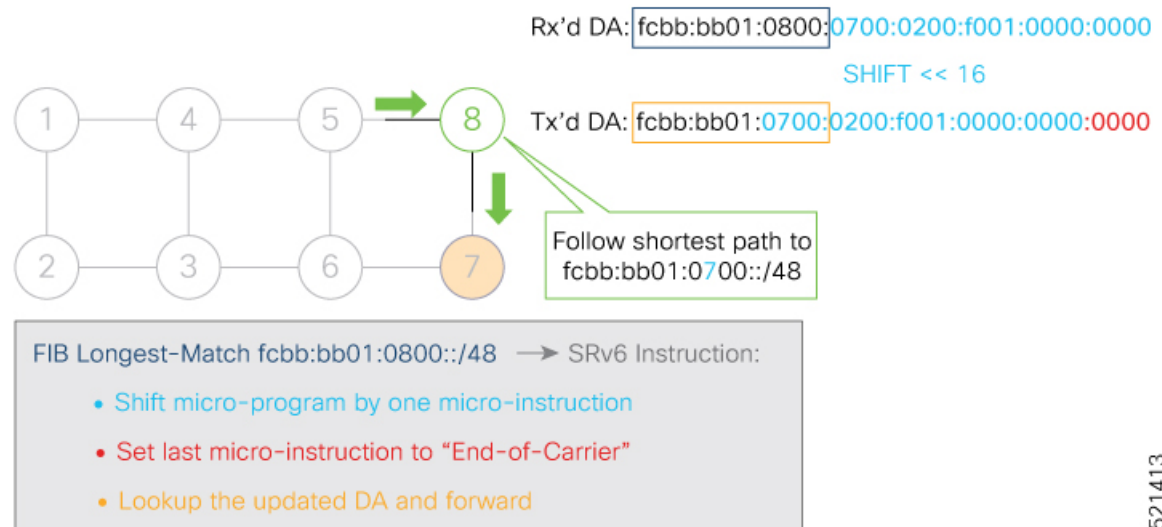*Figure 9: Node 6 and Node 3: Classic IPv6 Nodes*
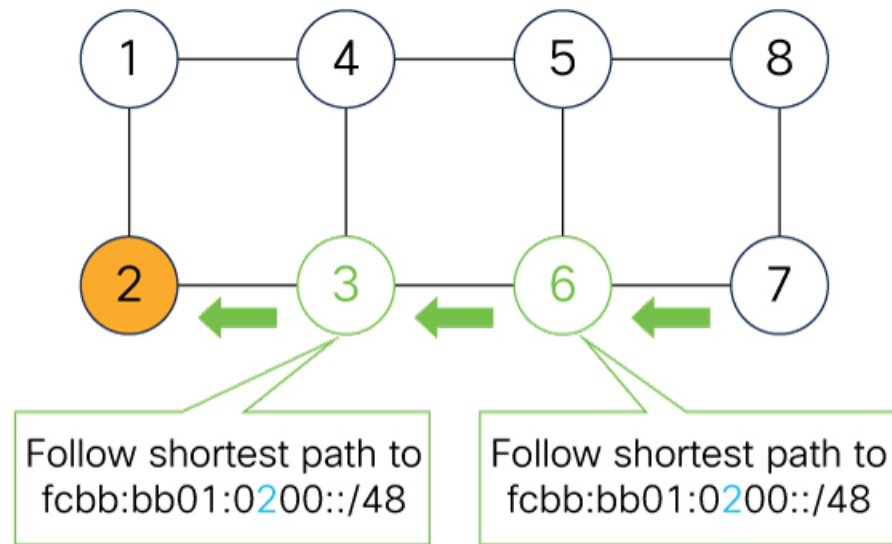
When Node 2 receives the packet, it performs an SRv6 uDT4 behavior (End.DT4—Endpoint with decapsulation and IPv4 table lookup) to VPNv4 Site B.

Figure 10: Node 2: SRv6 uDT4 Behavior
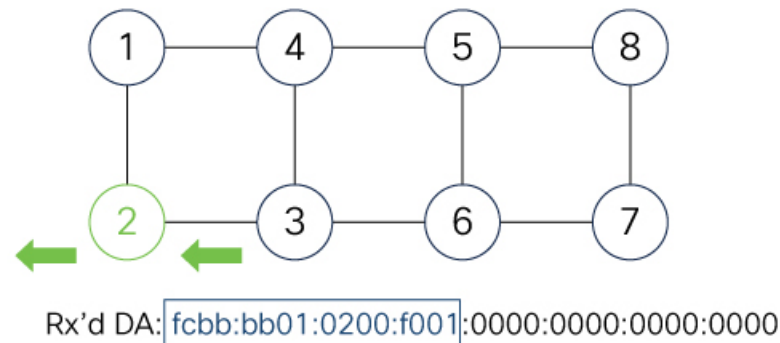
To recap, this example showed an integrated VPN and Traffic Engineering use-case, where VPNv4 site A connected to Node 1 sent packets to VPNv4 site B connected to Node 2 alongside a traffic engineered path via Node 8 and Node 7 using a single 128-bit SRv6 SID:

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

- @4 & @5: classic IP forwarding, outer DA unchanged

- @8: SRv6 uN behavior: shift and lookup, outer DA becomes fcbb:bb01:**0700**:**0200**:f001:0000:0000:0000

- @7: SRv6 uN behavior: shift and lookup, outer DA becomes fcbb:bb01:**0200**:f001:0000:0000:0000:0000

- @6 & @3: classic IP forwarding, outer DA unchanged

- @2: SRv6 End.DT4: Decapsulate and IPv4 table lookup

Usage Guidelines and Limitations

General Guidelines and Limitations

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

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

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

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

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

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

- SRv6 Underlay support includes:

    - IGP redistribution/leaking between levels

    - Prefix Summarization on ABR routers

    - IS-IS TI-LFA

    - Microloop Avoidance

    - Flex-algo

- SRv6 over GRE interface is not supported

- SRv6 over BVI interface is not supported

uSID Allocation Recommendation

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

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

For example:

- Using /24 from FC::/8 ULA

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

- SRv6 uSID Block = FC*BB*:*BBVV*/32, with *VV* indicating a nibble value picked by the operator.

  - 256 /32 uSID blocks possible from this allocation, from block 0 (FC*BB*:*BB00*/32) to block 255(FC*BB*:*BBFF*/32)

  - A network slice is assigned a /32 uSID block:

    - FCBB:BB**00**/32 for min-cost slice (shortest path based on minimum IS-IS cost)

    - FCBB:BB**08**/32 for min-delay slice (shortest path based on minimum latency using Flex Algo instance 128)

Platform-Specific Guidelines and Limitations

- SRv6 is supported on the following Cisco 8000 series Q200-based line cards and fixed-port routers:

  - Cisco 8800 with 88-LC0-36FH-M, 88-LC0-36FH, 88-LC0-34H14FH line cards

  - Cisco 8201-32FH

  - Cisco 8102-64H, 8101-32-FH

- SRv6 is not supported on Q100-based line cards and fixed-port routers.

- Egress marking on the outer header during SRv6 encapsulation operations (TI-LFA) is not supported.

- OAM: Ping and traceroute are supported.

- Cisco 8000 series routers support the following SRv6 uSID behaviors and variants:

  - **Endpoint behaviors**:

    - uN with PSP/USD

    - uA with PSP/USD

    - uDT4

    - uDT6

    - uDT46

  - **Head-end behaviors**:

    - H.Encap.Red (1 uSID carrier with up to 6 uSIDs)

- **Encapsulation Capabilities and Parameters**

  The following describes the Cisco 8000 series router capabilities for setting or propagating certain fields in the outer IPv6 header for SRv6 encapsulated packets:

  - **Source address**: Cisco 8000 series routers support a single source address (SA) for SRv6 encapsulated packets. The SA is derived from the SRv6 global configuration; if not configured, it is derived from the IPv6 Loopback address.

  - **Hop limit**:

    - Overlay encapsulation

Default: propagate=No

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

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

Manual configuration of the hop-limit value in the outer IPv6 header is not supported. See #unique_ 24 unique_24_Connect_42_section_mw5_4w5_qtb.

- **Traffic-class**:

  - Overlay encapsulation

  Default: propagate=No

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

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

  Manual configuration of the traffic-class value in the outer IPv6 header is not supported. See #unique_ 24 unique_24_Connect_42_section_mw5_4w5_qtb.

- **Flow Label**:

  - Cisco 8000 series routers use the flow-label from the incoming IPv6 header. In case of USD operations, flow-label is used from the inner IPv6 header.

  - During H.Encap.Red operations, if the inner packet has a flow label (non-zero value), the Cisco 8000 series routers propagate it to the outer IPv6 header. If the flow label is not present (zero), it is computed.

- P role:

  - Underlay H-Encap: 6 sids (1 carrier with 6 sids per carrier)

- **PE role:**

  - Underlay H-Insert: 3 sids (1 carrier with 3 sids per carrier)

  - Overlay H-Encaps: 3 sids (1 carrier with 3 sids per carrier)

Configuring SRv6

Enabling SRv6 involves the following high-level configuration steps:

- Configure SRv6 locator(s)

- Enable SRv6 under IS-IS

- Enable SRv6 Services under BGP

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

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

- **segment-routing srv6 locators locator** *locator*—Globally enable SRv6 and configure the locator.

- **segment-routing srv6 locators locator** *locator* **prefix** *ipv6_prefix*/*length*—Configure the locator prefix value.

- **segment-routing srv6 locators locator** *locator* **micro-segment behavior unode psp-usd**—Specifies the locator as a micro-segment (uSID) locator as well as specifies that IGP underlay uSID (uN/uA) variant is PSP-USD for this locator.

(Optional) Configure Algorithm Associated with Locator

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

  For additional information about SRv6 Flexible Algorithm, see Configuring SRv6 Flexible Algorithm under IS-IS, on page 25.

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

(Optional) Configure Anycast Locator

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

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

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

The following behaviors apply to Anycast Locator:

- Unlike a normal locator, IS-IS does not program or advertise uA SIDs associated with an Anycast locator.

- uN SIDs allocated from Anycast locators will not be used in constructing TI-LFA backup paths or Microloop Avoidance primary paths. TI-LFA backup and Microloop Avoidance paths for an Anycast locator prefix may terminate on any node advertising that locator, which may be different from the node terminating the original primary path.

- SRv6 anycast locators may have non-zero algorithm (Flexible Algorithm) values.

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

- **segment-routing srv6 locators locator** *locator* **anycast**—Configure the Anycast locator

- **router isis** *instance-id* **interface Loopback** *instance* **prefix-attributes anycast level** *level*—Advertise the Anycast prefixes associated with an interface.

Example 1:

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

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc1
Router(config-srv6-locator)# micro-segment behavior unode psp-usd
```

```
Router(config-srv6-locator)# prefix 2001:0:8::/48
```

Example 2:

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

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

Example 3:

The following example shows how to configure Anycast locator.

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

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

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

Example 4:

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

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

(Optional) Customize SRv6 Logging for Locator Status Changes

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

(Optional) Customize SRv6 SID Parameters

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

Example 4:

The following example shows how to configure optional SRv6 parameters:

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

Verifying SRv6 Manager

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

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

  H.Encaps.Red
Security rules:
  SEC-1
  SEC-2
  SEC-3
Counters:
  None
Signaled parameters:
  Max-SL : 3
  Max-End-Pop-SRH : 3
  Max-H-Insert : 0 sids
  Max-H-Encap : 2 sids
  Max-End-D : 5
Configurable parameters (under srv6):
  Ranges:
    LIB : Yes
```

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

Verifying SRv6 Locator

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

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

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

Verifying SRv6 SIDs

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

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

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

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

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

Configuring SRv6 under IS-IS

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

SRv6 IS-IS performs the following functionalities:

1. Interacts with SID Manager to learn local locator prefixes and announces the locator prefixes in the IGP domain.

2. Learns remote locator prefixes from other ISIS neighbor routers and installs the learned remote locator IPv6 prefix in RIB or FIB.

3. Allocate or learn prefix SID and adjacency SIDs, create local SID entries, and advertise them in the IGP domain.

Usage Guidelines and Restrictions

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

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

Configuring SRv6 IS-IS

To configure SRv6 IS-IS, 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 IS-IS.

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

Configuring SRv6 Flexible Algorithm under IS-IS

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

SRv6 Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As

a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

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

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

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
```

Configuring SRv6 IS-IS TI-LFA with Flexible Algorithm

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

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

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

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

Verification

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

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

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

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

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

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

```
        Binding Label: None
        Extended communities count: 0
        Path id:65           Path ref count:1
        NHID:0x20002(Ref:19)
        SRv6 Headend: H.Encaps.Red, SID-list {cafe:0:4::}
      fe80::e00:ff:fe3a:c700, from cafe:0:2::2, via HundredGigE0/0/0/0, Protected
        Route metric is 20
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:1        Path ref count:0
        NHID:0x20001(Ref:19)
        Backup path id:65
   Route version is 0x4 (4)
   No local label
   IP Precedence: Not Set
   QoS Group ID: Not Set
   Flow-tag: Not Set
   Fwd-class: Not Set
   Route Priority: RIB_PRIORITY_NON_RECURSIVE_MEDIUM (7) SVD Type RIB_SVD_TYPE_LOCAL
   Download Priority 1, Download Version 66
   No advertising protos.
```

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

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

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

    Load distribution: 0 (refcount 2)

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

Configuring SRv6 IS-IS Microloop Avoidance

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

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

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

Usage Guidelines and Limitations

The following usage guidelines and limitations apply:

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

Configuring SRv6 IS-IS Microloop Avoidance

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

> **Note** Complete the Configuring SRv6, on page 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 BGP-Based Services

Table 5: Feature History Table

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

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

- IPv4 Layer-3 VPNs

- IPv6 Layer-3 VPNs

- IPv4 BGP global

- IPv6 BGP global

- Layer-2 VPNs - Ethernet VPNs (EVPN)

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

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

- uDT4 (Endpoint with decapsulation and IPv4 table lookup)

- uDT6 (Endpoint with decapsulation and IPv6 table lookup)

- uDX4 (Endpoint with decapsulation and IPv4 cross-connect)

- uDX6 (Endpoint with decapsulation and IPv6 cross-connect)

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

Usage Guidelines 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 6: Feature History Table

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

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

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

Usage Guidelines and Limitations

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

Configuring SRv6 based IPv4 L3VPN

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

Use Case 1: Assigning SRv6 Locator Globally

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

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

Running Config

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

Use Case 2: Assigning SRv6 Locator for All VRFs

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

- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6**: Enable SRv6

- **router bgp** *as-number* **address-family vpnv4 unicast vrf all segment-routing srv6 alloc mode {per-vrf}**: 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
```

Verification

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

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

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

```
Node1# show segment-routing srv6 sid

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

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

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

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

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

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

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

```
Node1# show bgp vpnv4 unicast summary

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

BGP is operating in STANDALONE mode.


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


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


Node1# show bgp vpnv4 unicast rd 100:1

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

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

Processed 3 prefixes, 3 paths
```

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

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

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

```
Node1# show bgp vrf vrf_cust1 ipv4 unicast

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

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

Processed 3 prefixes, 3 paths


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

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

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

```
Node1# show route vrf vrf_cust1

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

Gateway of last resort is not set

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


Node1# show route vrf vrf_cust1 12.4.4.4/32

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


Node1# show route vrf vrf_cust1 12.4.4.4/32 detail

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

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

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

```
Node1# show cef vrf vrf_cust1

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

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

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

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

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

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

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

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

Load distribution: 0 1 (refcount 1)

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

SRv6 Services: IPv6 L3VPN

Table 7: Feature History Table

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

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

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally, for all VRFs, for an individual VRF, or per-prefix.

- Per-VRF allocation mode is supported (uDT6 behavior)

- Dual-Stack L3VPN Services (IPv4, IPv6) are supported

- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.

- eBGP, OSPF, Static are supported as PE-CE protocol.

  BGP (iBGP, eBGP), OSPF, Static are supported as PE-CE protocol.

- BGP route leaking between BGP Global and L3VPN is supported. Refer to the Implementing BGP chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers* .

- MPLS L3VPN and SRv6 L3VPN interworking gateway is supported.

- Per-CE allocation mode is not supported (uDX6 behavior)

Configuring SRv6-based IPv6 L3VPN

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

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
```

Verification

The following figure shows a VPNv6 scenario. The sequence of commands included correspond to router Node1 acting as Ingress PE, and routers Node4 and Node5 acting as Egress PEs.
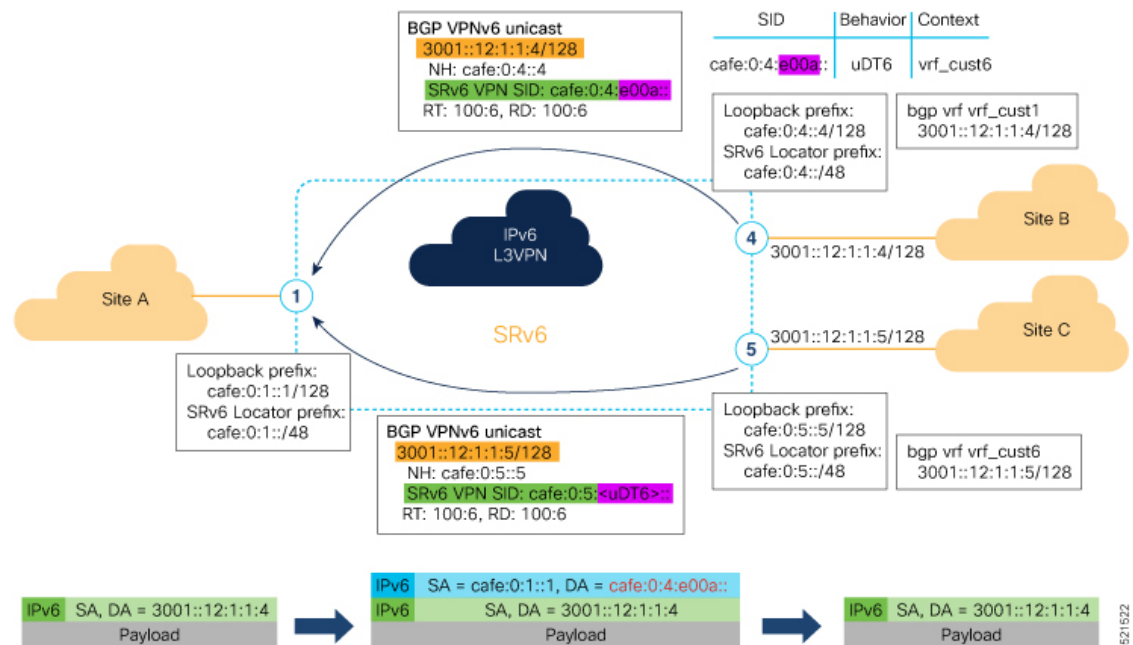
The following examples shows how to verify the SRv6 based L3VPN configurations for an Individual VRF with per VRF label allocation mode.

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

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

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

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

```
          InUse  Y
```

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

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

BGP is operating in STANDALONE mode.


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


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

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

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

Processed 3 prefixes, 3 paths

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

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

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

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

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

Processed 3 prefixes, 3 paths

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

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

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

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

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

Gateway of last resort is not set

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

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

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

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

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

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

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

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

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

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

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

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

  Load distribution: 0 1 (refcount 1)

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

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

SRv6 Services: IPv4 BGP Global

This feature extends support of SRv6-based BGP services to include IPv4 global BGP by implementing uDX4 and 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 theImplementing BGP chapter in the *Routing Configuration Guide for Cisco 8000 Series Routers*

- Dual-Stack L3 Services (IPv4 BGP global, IPv6 BGP global) are supported.

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

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

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

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

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 modes are per-VRF and per-CE.

- For per-VRF allocation mode, you can also specify the SRv6 locator.

  - If an SRv6 locator is specified in the route policy, BGP will use that to allocate per-VRF SID. If the specified locator is invalid, the SID will not be allocated.

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

- Per-CE allocation mode always uses the default locator to allocate the SID.

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

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

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

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

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

Running Configuration

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

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

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


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

SRv6 Services: IPv6 BGP Global

Table 8: Feature History Table

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

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

Usage Guidelines and Limitations

- SRv6 locator can be assigned globally or under IPv6 unicast address family

- Equal-Cost Multi-path (ECMP) and Unequal Cost Multipath (UCMP) are supported.

- BGP, OSPF, Static are supported as PE-CE protocol.

- Dual-Stack L3 Services (IPv4 BGP global, IPv6 BGP global) are supported.

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

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

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

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

Running Configuration

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

Verification

The following figure shows a IPv6 BGP global scenario. The sequence of commands included correspond to router Node1 acting as Ingress PE, and routers Node4 and Node5 acting as Egress PEs.
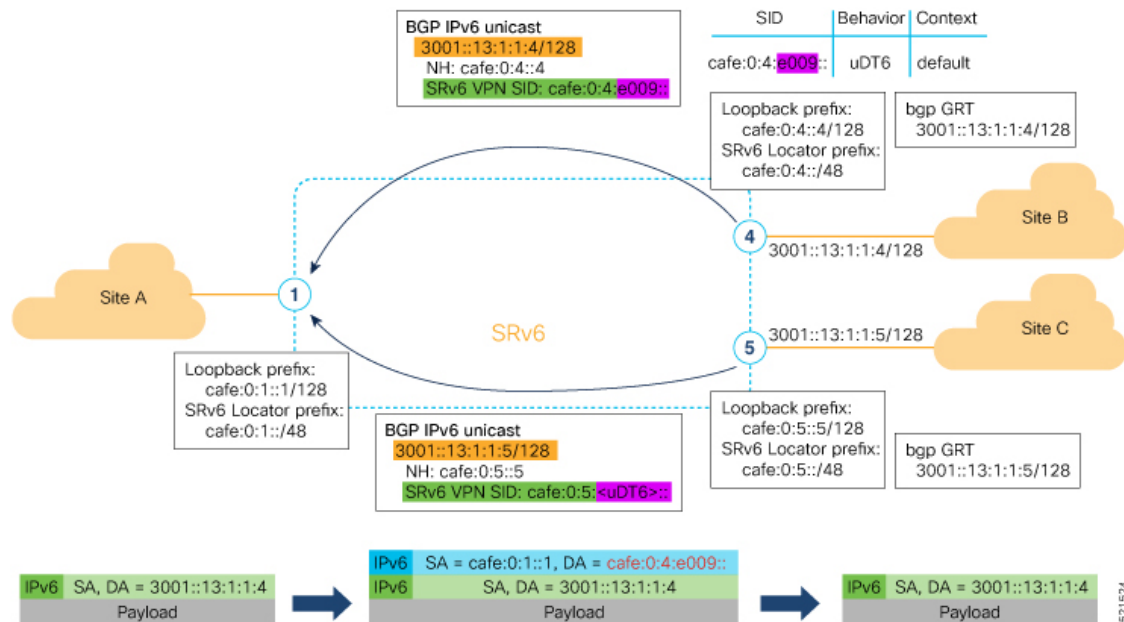
The following examples show how to verify the BGP global IPv6 configuration using the **show bgp ipv6 unicast** commands.

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


BGP is operating in STANDALONE mode.


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


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

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


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

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

Processed 3 prefixes, 3 paths

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

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

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

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

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

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

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

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

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

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

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

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

   Load distribution: 0 1 (refcount 1)

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

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          : 11.1111.1111.1111.1111.1114
  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/MPLS L3 Service Interworking Gateway

Table 9: Feature History Table

Feature Name	Release	Description
SRv6/MPLS L3 Service Interworking Gateway (SRv6 Micro-SID)	Release 7.8.1	This feature enables you to extend L3 services between MPLS and SRv6 domains by providing service continuity on the control plane and data plane. This feature allows for SRv6 L3VPN domains to interwork with existing MPLS L3VPN domains. The feature also allows migration from MPLS L3VPN to SRv6 L3VPN.

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

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

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

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

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

The gateway performs the following actions:

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

SRv6/MPLS L3 Service Interworking Gateway Scenarios

The following scenario is used to describe the gateway functionality:

- Node 1 is an L3VPN PE in the MPLS domain with an SR prefix SID label of 16001 for its Loopback interface 1.1.1.1/32.

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

- Node 3 is an L3VPN PE in the SRv6 domain with SRv6 locator of B:0:3::/48 and Loopback interface B:0:3::3/128.
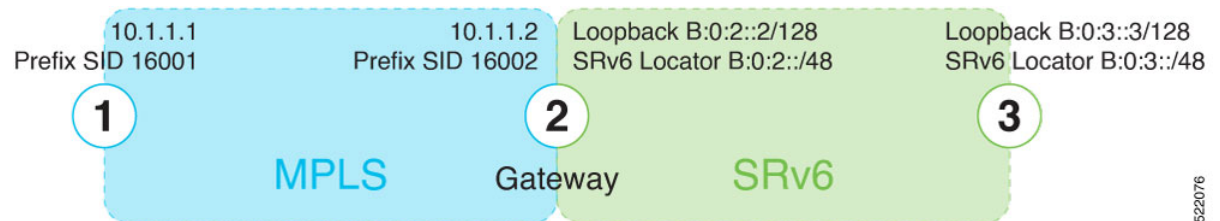
Scenario 1: SRv6-to-MPLS Control-Plane Direction/MPLS-to-SRv6 Data-Plane Direction

The figure below describes the associated control-plane behaviors in the SRv6-to-MPLS direction for traffic in the MPLS-to-SRv6 data-plane direction.
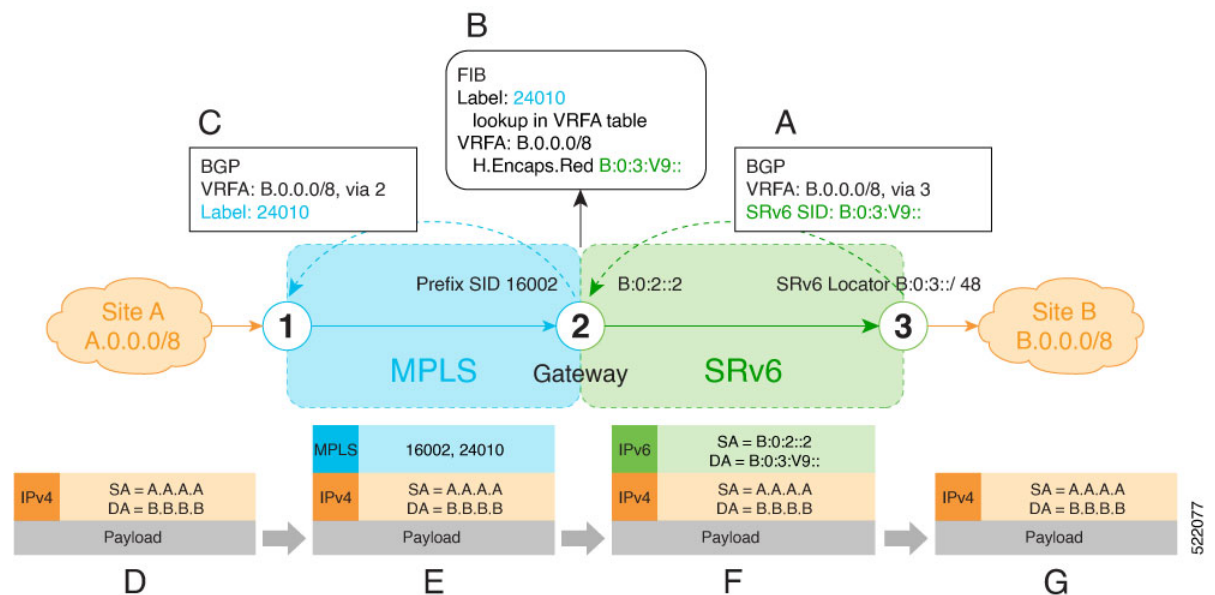
A. Node 3 advertises a BGP L3VPN update for prefix B.0.0.0/8 with RD corresponding to VRFA, including the SRv6 VPN SID (B:0:3:V9::) assigned to this VRF, in the SRv6 domain.

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

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

- MPLS label 24010 is allocated for VRFA

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

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

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

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

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

F. Node 2 performs the following actions:

- Pops the MPLS VPN label and looks up the destination prefix

- Encapsulates the payload in an outer IPv6 header with destination address (DA) equal to the H.Encaps.Red function (B:0:3:V9::)

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

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

The figure below describes the associated control-plane behaviors in the MPLS-to-SRv6 direction for traffic in the SRv6-to-MPLS data-plane direction.
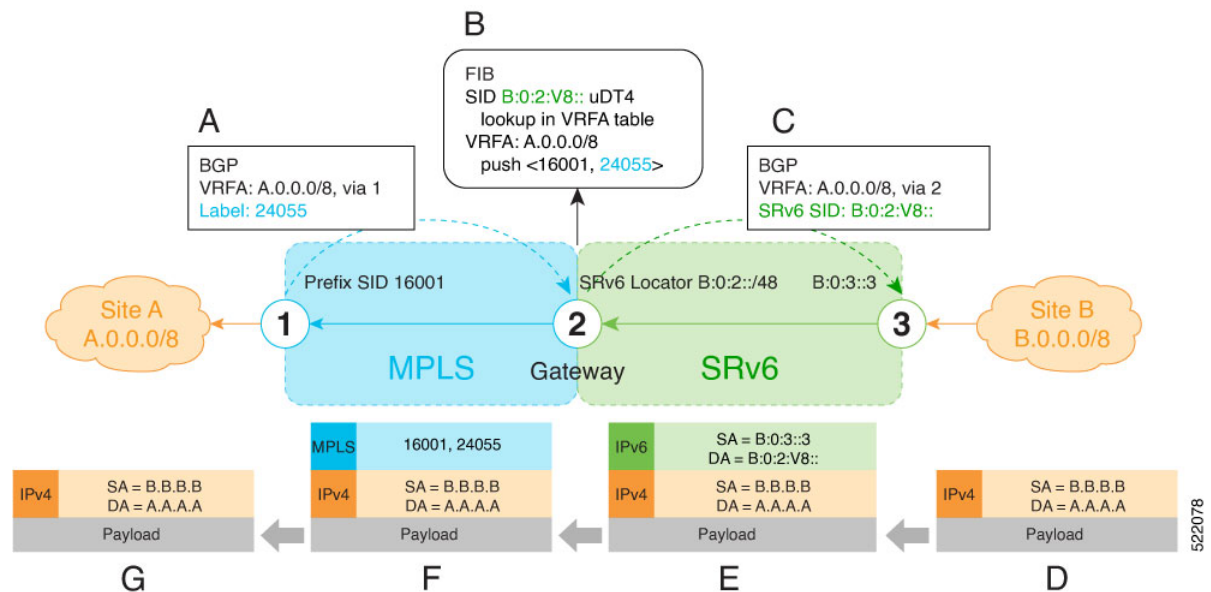
A. Node 1 advertises a BGP L3VPN update for prefix A.0.0.0/8 with RD corresponding to VRFA, including the MPLS VPN label (24055) assigned to this VRF, in the MPLS domain.

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

- Prefix A.0.0.0/8 is programmed to impose an MPLS VPN label (24055) and the prefix SID MPLS label (16001) of the BGP next-hop (Node 1)

- "Endpoint with decapsulation and IPv4 table lookup" function (uDT4) of B:0:2:V8:: is allocated to VRFA

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

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

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

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

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

F. Node 2 performs the following actions:

- Removes the outer IPv6 header and looks up the destination prefix

- Pushes the MPLS VPN label (24055) and the prefix SID MPLS label (16001) of the BGP next-hop (Node 1)

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

Configuration

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

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

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

- MPLS L3VPN RT of 1111:1

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

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

```
Node2(config-vrf-export-rt)# 2222:1 stitching
```

This example shows how to configure BGP SRv6:

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

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

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

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

Example

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

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

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

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

- 1111:1, RT used for MPLS L3VPN

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

```
vrf ACME
 address-family ipv4 unicast
  import route-target
```

```
   1111:1
   2222:1 stitching
  !
 export route-target
  1111:1
  22222:1 stitching
  !
 !
!
```

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

```
router bgp 100
 segment-routing srv6
  locator LOC1
 !
 neighbor 1.1.1.1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   route-reflector-client
   advertise vpnv4 unicast re-originated
 !
 neighbor B:0:3::1
  address-family vpnv4 unicast
   import stitching-rt re-originate
   route-reflector-client
   encapsulation-type srv6
   advertise vpnv4 unicast re-originated stitching-rt
 !
 vrf ACME
  address-family ipv4 unicast
   enable label-mode
   segment-routing srv6
```

SRv6/MPLS Dual-Connected PE

Table 10: Feature History Table

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

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

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

- MPLS/IPv4 L3VPN between Site A and Site C

- SRv6/IPv4 L3VPN between Site B and Site C

Configure BGP to Support Dual-Mode

Enable MPLS Label Allocation

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

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

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

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

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

Running Config

```
router bgp 100
 neighbor 192::6
  remote-as 1
  address-family ipv4 unicast
   encapsulation-type srv6
```

```
 !
!
vrf blue
 rd 1:10
 address-family ipv4 unicast
  mpls alloc enable
  label mode per-ce
  segment-routing srv6
   alloc mode per-ce
  !
 !
!
!
```

SRv6 Provider Edge (PE) Lite Support

Table 11: Feature History Table

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

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

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

Note See Configuring Explicit End.DT46 SRv6 SIDs, on page 79 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
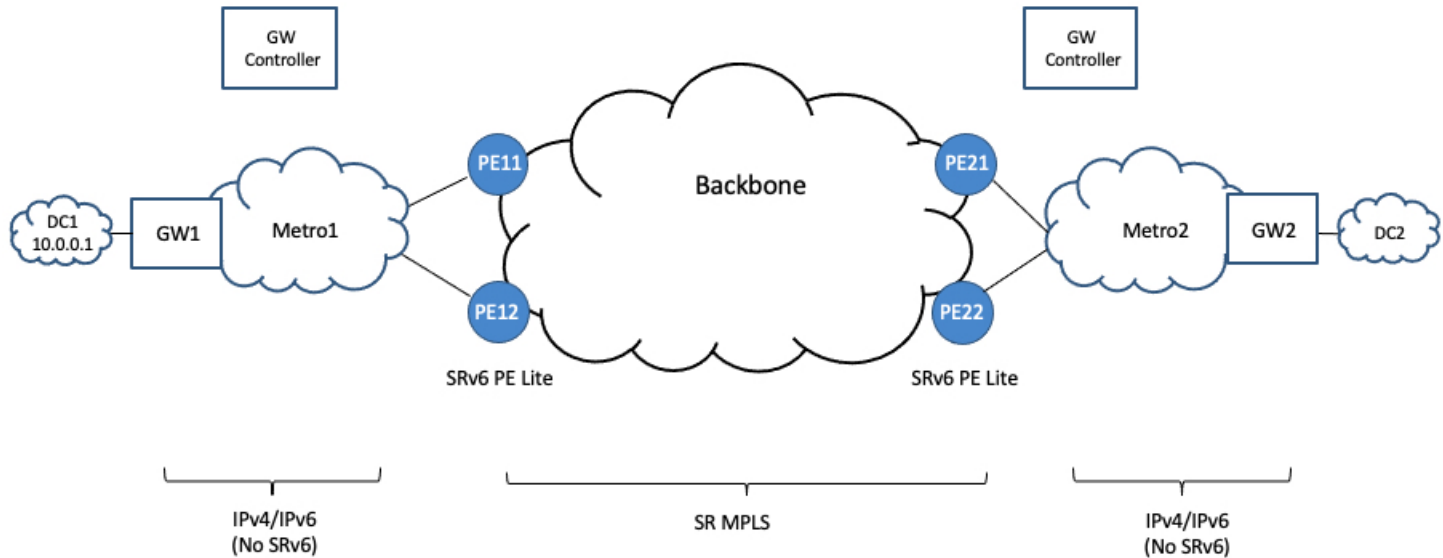
Data center gateways (GW1 and GW2) perform IP-in-IPv6 encapsulation where the outer IPv6 destination address represents an SRv6 network program that leads traffic to the SRv6 PE lite nodes (PE11 and PE12). This outer IPv6 destination address is determined by the gateway controller to provide a desired 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)
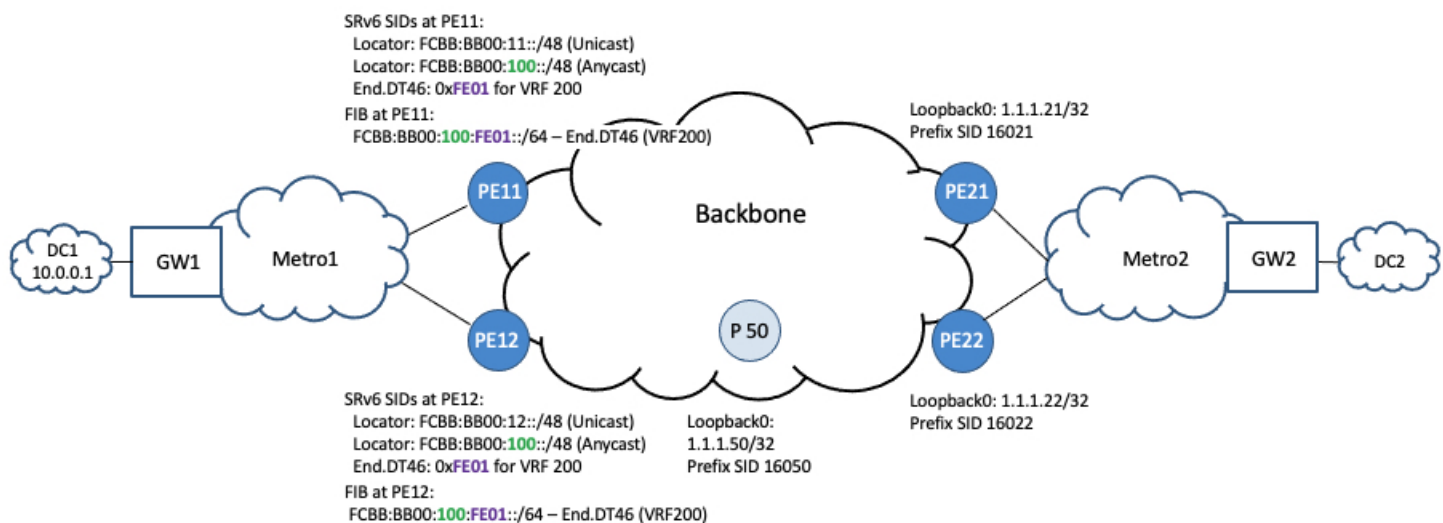

The SRv6 PE lite nodes are configured with SRv6 locators and explicit (manually assigned) service de-multiplexing end-point behaviors to perform decapsulation and VPN table lookup.

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)*
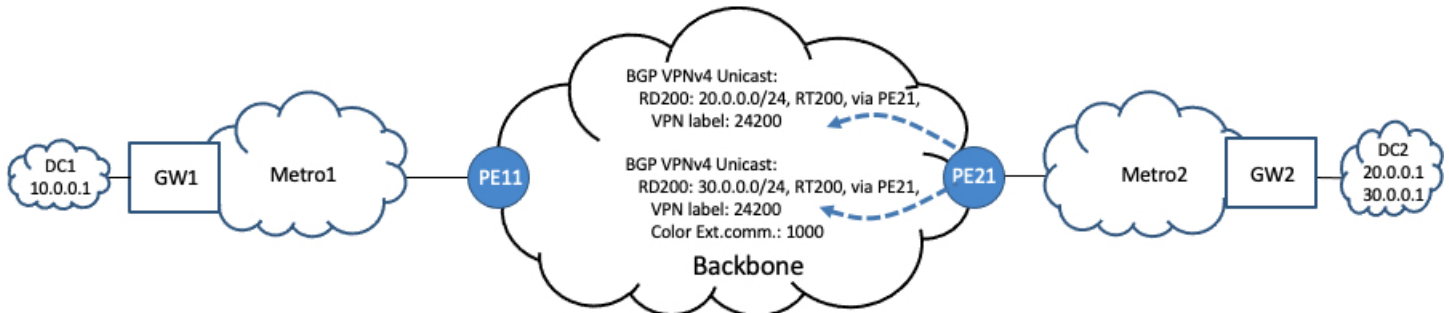
Prefixes from the data center are advertised in the backbone via multiprotocol BGP as part of a VPN. These prefixes can include a color extended community in order to indicate the desired transport SLA. For example, PE21 advertises BGP VPN overlay routes for DC2, 20.0.0.0/24 and 30.0.0.0/24. Prefix 20.0.0.0/24 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)
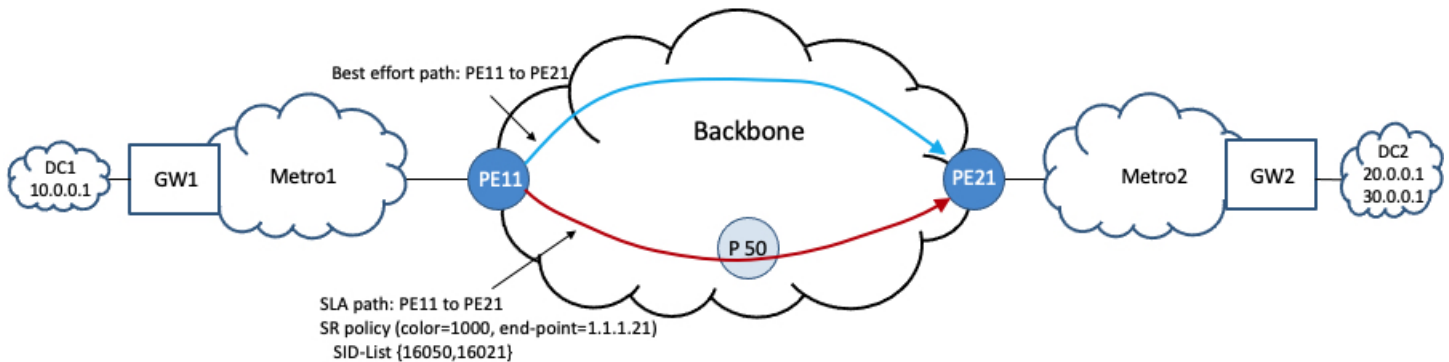
For traffic in the direction DC1 to DC2, the SRv6 PE lite node PE11 is an SR-TE headend of an SR policy associated with color 1000 and end-point of PE21. This SR policy will be used to steer traffic toward BGP service routes with color 1000 advertised by PE21. As an example, this SR policy is associated 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).
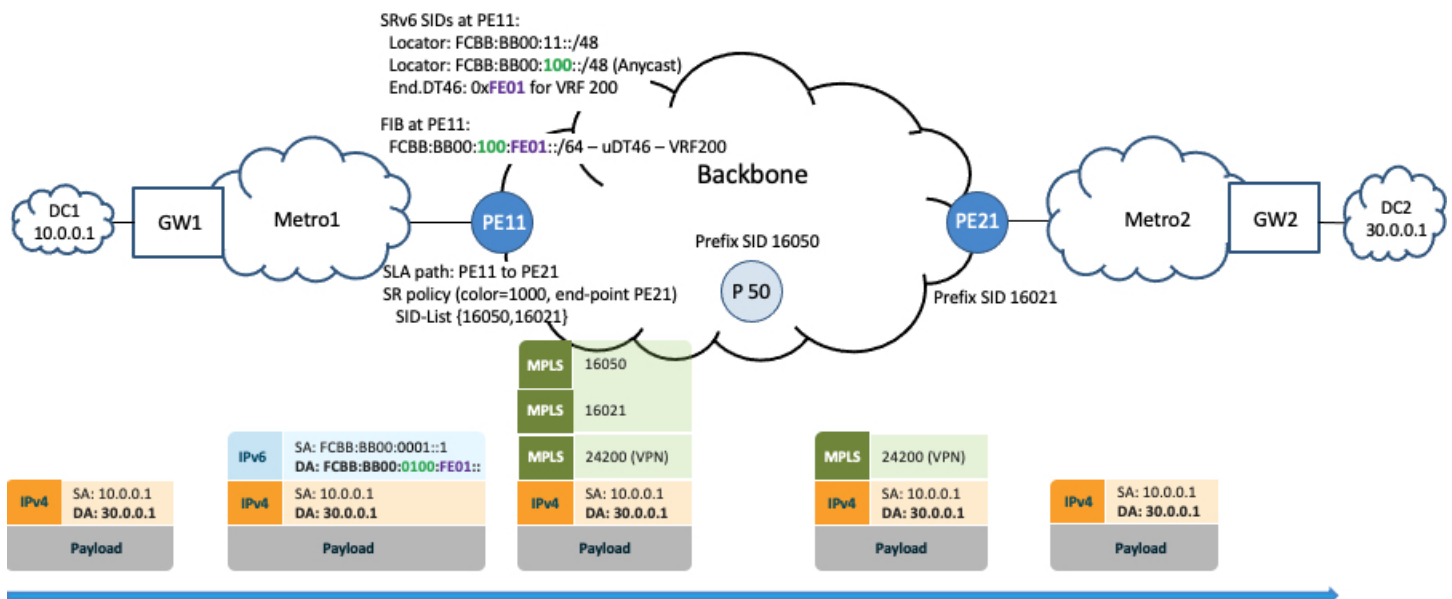
Figure 15: Best Effort Traffic (Traffic Direction – DC1 to DC2)
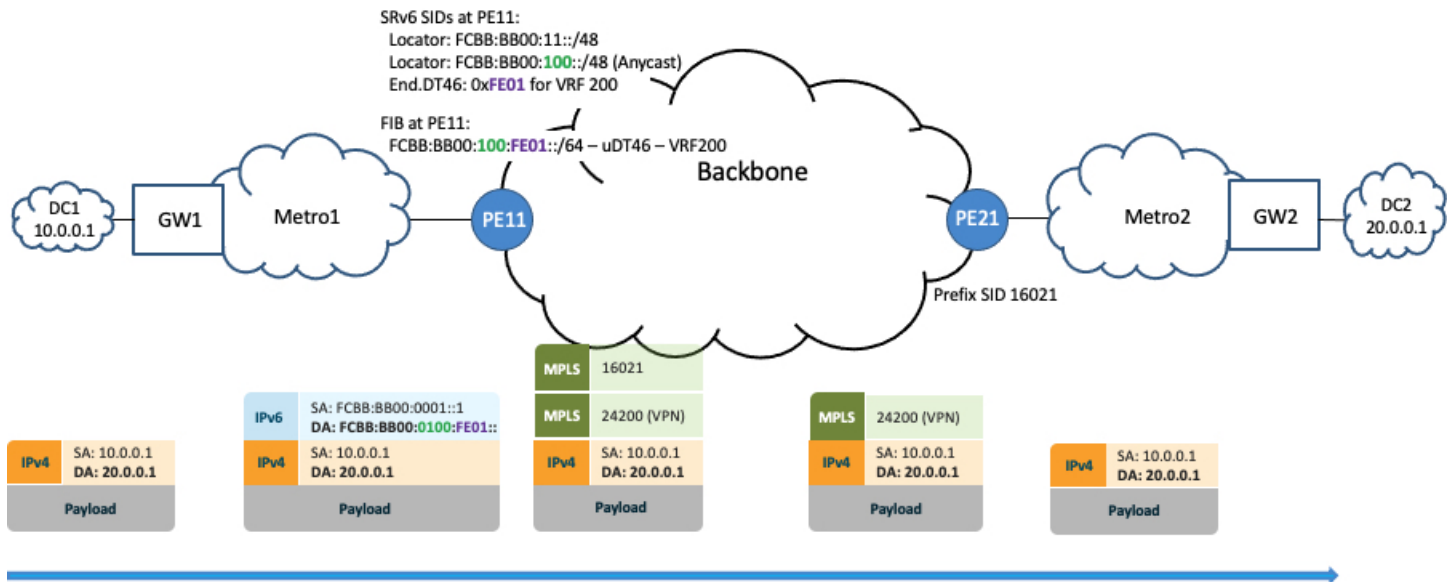
Traffic arriving at GW1 destined for 20.0.0.1 in DC2 is encapsulated into an outer IPv6 header with a destination address of FCBB:BB00:100:FE01::. This address is composed of the Anycast locator at PE11 and the explicit End.DT46 function value of VRF 200. Any transit nodes in the Metro1 domain simply 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
  !
 !
!
```

Configuring Explicit End.DT46 SRv6 SIDs

Table 12: Feature History Table

Feature Name	Release	Description
Support for End.DT46 SRv6 Endpoint Behavior	Release 7.5.3	This feature adds support for the "Endpoint with decapsulation and specific IP table lookup" SRv6 end-point behavior (End.DT46). The End.DT46 behavior is used for dual-stack L3VPNs. This behavior is equivalent to the single per-VRF VPN label (for IPv4 and IPv6) in MPLS.
Support for Explicit End.DT46 SRv6 SIDs	Release 7.5.3	This feature allows you to configure explicit SIDs associated with SRv6-based L3VPN/Internet BGP services. In previous releases, these SIDs were only allocated dynamically by BGP. Explicit End.DT46 SRv6 SIDs are persistent over reloads and restarts. The feature introduces these changes: **CLI:** • The **segment-routing srv6 static endpoint sid** *prefix* **behavior end-udt46** command mode is introduced.

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

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

Note See GIB and LIB – IOS-XR Implementation, on page 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 modify the start of the range of IDs available in the explicit LIB and explicit W-LIB.

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

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

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

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

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

Example

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

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

. . .
```

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

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

Running Config

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

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

```
RP/0/RP0/CPU0:ios# show segment-routing srv6 manager
Fri Sep  9 18:31:06.033 UTC
Parameters:
  SRv6 Enabled: Yes
```

```
     SRv6 Operational Mode: None
     Encapsulation:
       Source Address:
         Configured: ::
         Default: ::
       Hop-Limit: Default
       Traffic-class: Default
     SID Formats:
       f3216 <32B/16NFA> (2)
         uSID LIB Range:
           LIB Start   : 0xe000
           ELIB Start  : 0xe064 (configured)
         uSID WLIB Range:
           EWLIB Start : 0xfff0 (configured)


     . . .
```

Configure Seamless Bidirectional Forwarding Detection

Table 13: Feature History Table

Feature Name	Release	Description
Support for Cisco 8000 routers as Seamless BFD reflector	Release 7.5.3	This feature introduces support for Cisco 8000 series routers to act as a Seamless Bidirectional Forwarding Detection (SBFD) reflector. Seamless BFD (SBFD) eliminates many negotiation aspects and thereby provides a simplified approach to using BFD. Benefits of SBFD include quick provisioning, improved control, and flexibility for network nodes that initiate path monitoring. The SBFD reflector is an SBFD session on a network node that listens for incoming SBFD control packets to local entities and generates response SBFD control packets. The reflector is stateless and only reflects the SBFD packets back to the initiator.

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

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

Initiators and Reflectors

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

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

Figure 17: SBFD Initiator and Reflector
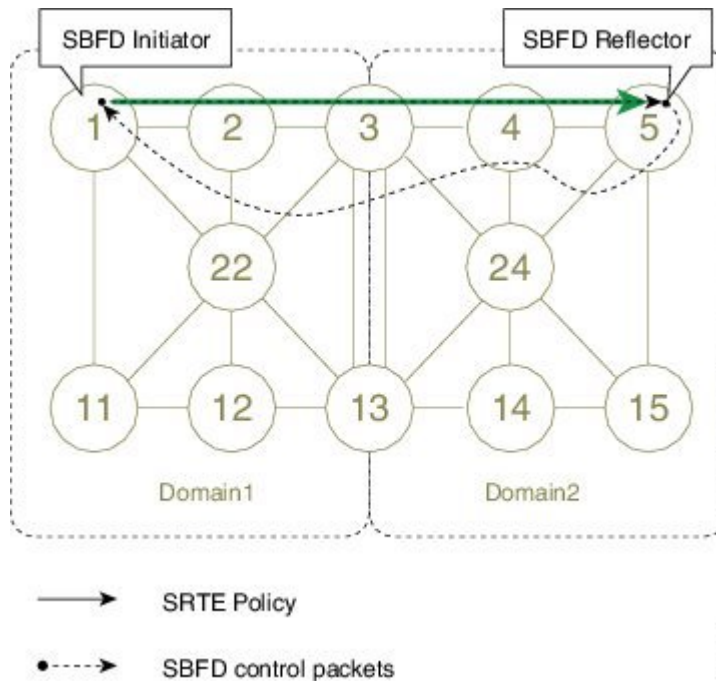
The initiator is an SBFD session on a network node that performs a continuity test to a remote entity by sending SBFD packets. The initiator injects the SBFD packets into the SR-TE policy. The initiator triggers the SBFD session and maintains the BFD state and client context.

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

Note Cisco 8000 series routers support reflector mode only.

Discriminators

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

The SBFD Discriminator must be unique within an administrative domain. If multiple network nodes allocate the same SBFD Discriminator value, then SBFD Control packets falsely terminating on a wrong network node can result in a Reflector BFD session generating a response back because of a matching Your Discriminator value.

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

Configuring the SBFD Reflector

To ensure the SBFD packet arrives on the intended reflector, each reflector has at least one globally unique discriminator. Globally unique discriminators of the reflector are known by the initiator before the session

starts. An SBFD reflector only accepts BFD control packets where "Your Discriminator" is the reflector discriminator.

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

Enable MPLS OAM

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

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

Configure Local Discriminators on the Reflector

Use the **sbfd local-discriminator** {*ipv4-address* | *32-bit-value* | **dynamic** | **interface** *interface*} command to configure a local discriminator for the reflector. The following example shows the different ways to configure a local discriminator.

```
Router_5(config)# sbfd
Router_5(config-sbfd)# local-discriminator 10.1.1.5
Router_5(config-sbfd)# local-discriminator 987654321
Router_5(config-sbfd)# local-discriminator dynamic
Router_5(config-sbfd)# local-discriminator interface Loopback0
```

Running Config

```
mpls oam
!
sbfd
 local-discriminator 10.1.1.5
 local-discriminator 987654321
 local-discriminator dynamic
 local-discriminator interface Loopback0
!
```

Verifying SBFD Reflector

```
Router_5# show bfd target-identifier local

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

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


Router_5# show bfd reflector info detail location 0/0/CPU0

Local Discr    : 2147483649
Remote Discr   : 65576
Source Address : 1.1.1.1
```

```
Last DOWN received Time : (NA)
Last Rx packets timestamps before DOWN
  [NA                    ] [NA                    ] [NA                    ] [NA                    ]
  [NA                    ] [NA                    ] [NA                    ] [NA                    ]
  [NA                    ] [NA                    ]
Last Tx packets timestamps before DOWN
  [NA                    ] [NA                    ] [NA                    ] [NA                    ]
  [NA                    ] [NA                    ] [NA                    ] [NA                    ]
  [NA                    ] [NA                    ]
Last UP sent Time : (Jun  7 14:59:34.763)
Last recent Rx packets timestamps:
  [Jun 7 15:00:18.653 ] [Jun 7 15:00:18.751 ] [Jun 7 15:00:18.837 ] [Jun 7 15:00:18.927 ]
  [Jun 7 15:00:18.085 ] [Jun 7 15:00:18.185 ] [Jun 7 15:00:18.274 ] [Jun 7 15:00:18.372 ]
  [Jun 7 15:00:18.464 ] [Jun 7 15:00:18.562 ]
Last recent Tx packets timestamps:
  [Jun 7 15:00:18.653 ] [Jun 7 15:00:18.751 ] [Jun 7 15:00:18.837 ] [Jun 7 15:00:18.927 ]
  [Jun 7 15:00:18.085 ] [Jun 7 15:00:18.185 ] [Jun 7 15:00:18.274 ] [Jun 7 15:00:18.372 ]
  [Jun 7 15:00:18.464 ] [Jun 7 15:00:18.563 ]
```

SRv6 SID Information in BGP-LS Reporting

BGP Link-State (BGP-LS) is used to report the topology of the domain using nodes, links, and prefixes. This feature adds the capability to report SRv6 Segment Identifier (SID) Network Layer Reachability Information (NLRI).

The following NLRI has been added to the BGP-LS protocol to support SRv6:

- Node NLRI: SRv6 Capabilities, SRv6 MSD types

- Link NLRI: End.X, LAN End.X, and SRv6 MSD types

- Prefix NLRI: SRv6 Locator

- SRv6 SID NLRI (for SIDs associated with the node): Endpoint Function, BGP-EPE Peer Node/Set

This example shows how to distribute IS-IS SRv6 link-state data using BGP-LS:

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

Note It is still possible to ping or trace a SID:

- **ping** B:k:F::

- **traceroute** B:k:F::

It is possible to use a list of packed carriers to ping or trace a SID, to ping or trace route, use <destination SID> via srv6-carriers <list of packed carriers>

Full-Replace Migration to SRv6 Micro-SID

Table 14: Feature History Table

Feature Name	Release	Description
Full-Replace Migration to SRv6 Micro-SID	Release 7.8.1	This feature enables migration of existing SRv6 SID format1 to SRv6 Micro-SIDs (f3216) formats.
		Earlier, only one format was supported at a time, and you had to choose either format1 or Micro-SID format for the deployment of services. Migration from Full-length SIDs to SRv6 Micro-SIDs was not possible.

During the Full-Replace migration, both underlay and services are migrated from format1 to f3216. The underlay migration is done using the *Ship in the night* strategy, where updates into your environment are incremental, thereby phasing out your existing transport protocols when ready. This method minimizes the service disruption, and is recommended for seamless migration. The services migration is done using *swap* procedures, where the incoming transport label is swapped with an outgoing transport label.

The format1 to f3216 migration is seamless, requires minimal configurations, and no IETF signaling extensions. The migration enables preference of Micro-SID f3216 over format1, and minimizes traffic drop with faster convergence.

Cisco 8000 series routers support the following configurations for migration from format1 to Micro-SIDs:

- IS-IS underlay (TILFA, uLoop, FlexAlgo)

The following modes are supported in the context of migration:

- **Base**: SRv6 classic with format1 only.

- **Dual**: SRv6 classic with format1 and SRv6 Micro-SID with f3216 will both coexist.

- **f3216**: Micro-segment format. f3216 represents the format 3216, which is 32-bit block and 16-bit IDs.

The migration starts with SRv6 base format1, and ends with SRv6 uSID f3216. The migration states in detail are:

The migration process involves the following steps:

1. **Prepare for migration**: Upgrade the network nodes to an image that is Micro-SID f3216 capable, and allows the coexistence of format1 and f3216.

2. **Migrate the underlay to Micro-SID**: Enable IS-IS as an underlay protocol on PE nodes. The IS-IS configuration adds f3216 locators to format1 locators. Both format1 and f3216 endpoint SIDs are allocated, installed, and announced during this stage. f3216 is the preferred option over format1 for underlay paths.

   The IS-IS SR headends provide faster convergence to Micro-SID. Faster convergence to f3216 is done on the per-prefix per-path level, does not need any new CLI, and avoids packet drops. The format1 locators

are removed after underlay traffic convergence to f3216 on all nodes. The format1 locators are unconfigured from IS-IS, and deleted from SRv6.

At the end of this step, the migration status of the following P Nodes are:

- Locator reachability: f3216 only
- Underlay endpoint/headends: f3216 only

At the end of this step, the migration status of the following PE Nodes are:

- Locator reachability: format1 and f3216
- Underlay endpoint/headends: f3216 only
- Overlay endpoint/headends: format1

3. **Migrate the overlay to Micro-SID**: Enables overlay f3216 under BGP and EVPN on all PE nodes. The BGP and EVPN configuration replaces format1 by f3216 locators. During this stage, the f3216 Micro-SIDs are allocated, installed, and announced, while the format1 SIDs are deallocated, uninstalled, and withdrawn.

The format1 locators are removed after overlay traffic convergence to f3216 on all nodes. The format1 locators are unconfigured from BGP and EVPN, and deleted from SRv6.

For a transient period, BGP and EVPN might have some paths with format1 and some with f3216.

At the end of this step, the migration status of the following are:

- For P/PE Nodes:
    - Locator reachability: f3216 only
    - Underlay endpoint/headends: f3216 only
    - Overlay endpoint/headends: f3216 only

The migration starts with SRv6 base format1, and ends with SRv6 Micro-SID f3216. The migration states are:

1. **Initial state**: This is the early migration state of a deployment, for the supported features. This state comprises SRv6 base with format1.

This example shows the initial state of migration with SRv6 and configure locator:

```
Router(config)# segment-routing srv6
Router(config-srv6)# locators
Router(config-srv6-locators)# locator myLoc0
Router(config-srv6-locators)# prefix f1bb:bbbb:bb00:0001::/64
```

This example shows the initial state of migration with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myLoc0
```

This example shows the initial state of migration with SRv6 and BGP/EVPN:

```
Router(config)# router bgp 100
```

```
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myLoc0
```

2. **In-migration state**: The migration procedures are initiated, and are in progress. This state comprises SRv6 in dual mode (base with format1, and Micro-SID with f3216).

   This example shows the in-migration state with SRv6 and configure locator:

   ```
   Router(config)# segment-routing srv6
   Router(config-srv6)# locators
   Router(config-srv6-locators)# locator myLoc0
   Router(config-srv6-locators)# prefix f1bb:bbbb:bb00:0001::/64
   Router(config-srv6-locators)# delayed-delete
   Router(config-srv6-locators)# locator myuLoc0
   Router(config-srv6-locators)# micro-segment behavior unode psp-usd
   Router(config-srv6-locators)# prefix fcbb:bb00:0001::/48
   ```

   This example shows the in-migration state with SRv6 and IS-IS:

   ```
   Router(config)# router isis 100
   Router(config-isis)# address-family ipv6 unicast
   Router(config-isis-af)# segment-routing srv6
   Router(config-isis-srv6)# locator myLoc0
   Router(config-isis-srv6)# locator myuLoc0
   ```

   This example shows the in-migration state with SRv6 and BGP/EVPN:

   ```
   Router(config)# router bgp 100
   Router(config-bgp)# bgp router-id 10
   Router(config-bgp)# segment-routing srv6
   Router(config-bgp-srv6)# locator myuLoc0


   Router(config)# evpn
   Router(config-evpn)# segment-routing srv6
   Router(config-evpn-srv6)# locator myuLoc0
   ```

3. **End state**: This is the state of deployment at the end of the migration. At the end state, you can update the network and add new features. The Full-Replace migration end state can be of two modes:

   - **Full-Replace**: Both underlay and overlay are migrated to Micro-SID f3216. Full-Replace is the Cisco recommended migration type.

   - **uF1**: Underlay migrated to Micro-SID f3216, overlay remains format1. The uF1 migration is a transient state of the Full-Replace migration type.

   This example shows the end state with SRv6 and configure locator:

   ```
   Router(config)# segment-routing srv6
   Router(config-srv6)# locators
   Router(config-srv6-locators)# locator myuLoc0
   Router(config-srv6-locators)# micro-segment behavior unode psp-usd
   Router(config-srv6-locators)# prefix fcbb:bb00:0001::/48
   ```

This example shows the end state with SRv6 and IS-IS:

```
Router(config)# router isis 100
Router(config-isis)# address-family ipv6 unicast
Router(config-isis-af)# segment-routing srv6
Router(config-isis-srv6)# locator myuLoc0
```

This example shows the end state with SRv6 and BGP/EVPN:

```
Router(config)# router  bgp 100
Router(config-bgp)# bgp router-id 10
Router(config-bgp)# segment-routing srv6
Router(config-bgp-srv6)# locator myuLoc0


Router(config)# evpn
Router(config-evpn)# segment-routing srv6
Router(config-evpn-srv6)# locator myuLoc0
```

Run the following command to check the result of migration, as shown in the example:

```
RP/0/RSP0/CPU0:Router# sh route ipv6 fc00:cc30:600:e004:: detail
Wed Nov 10 18:57:56.645 UTC

Routing entry for fc00:cc30:600::/48
    Known via "isis 2", distance 115, metric 141, SRv6-locator, type level-2
    Installed Nov 2 18:56:55.718 for 00:01:01
    Routing Descriptor Blocks
      fe80::232:17ff:fec3:58c0, from 7511::1, via TenGigE0/0/0/16.1, Protected
        Route metric is 141
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:1  Path ref count:0
        NHID:0x20006(Ref:193)
        Backup path id:65
      fe80::226:80ff:fe36:7c01, from 7511::1, via TenGigE1/0/9/1.1, Backup (TI-LFA)
        Repair Node(s): 3888::1
        Route metric is 251
        Label: None
        Tunnel ID: None
        Binding Label: None
        Extended communities count: 0
        Path id:65  Path ref count:1
        NHID:0x20007(Ref:163)
        SRv6 Headend:H.Insert.Red [f3216], SID-list {fc00:cc30:700::}
  Route version is 0x0 (8)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-Class: Not Set
  Route Priority:RIB_PRIORITY_NON_RECURSIVE_LOW (8) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 2, Download Version 261731
  No advertising protos.
```