Advanced SRv6 Layer 3 features

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

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

- BGP Signaling for co-existence of IP routes with or without SRv6 SID, on page 1
- SRv6 Provider Edge Lite support, on page 7
- Explicit End DT46 SRv6 SIDs, on page 11
- SRv6 SID Information in BGP-LS Reporting, on page 16
- Path maximum transmission unit discovery for SRv6 encapsulated packets, on page 18
- VRF-to-VRF route leaking in SRv6 core, on page 20
- Dual-stack with SRv6 unicast and IPv4 multicast core, on page 25

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

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

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

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
BGP Signaling for co-existence of IP routes	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
BGP Signaling for co-existence of IP routes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
BGP Signaling for co-existence of IP routes	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100]; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) SRv6 with BGP supports the coexistence of IP routes with or without SRv6 SID over an SRv6-enabled core network. This support enables integrating SRv6 capabilities into existing network infrastructures without replacing IP routing completely. This feature enables flexibility and scalability, transition to new technologies, and enhanced network efficiency, making it easier to migrate from MPLS to SRV6. The feature introduces these changes: CLI: • **encapsulation-type srv6 relax-sid**

Configure BGP Signaling over SRv6 Core

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

Procedure

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

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

Example:

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

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

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

Example:

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

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

Example:

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

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

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

Example:

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

Router#show bgp update-group neighbor 192::4
Update group for IPv4 Unicast, index 0.1:
  Attributes:
    Neighbor sessions are IPv6
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    Next-hop-self enabled
    4-byte AS capable
    Advertise routes with local-label via Unicast SAFI
    Send AIGP
    Encapsulation type SRv6 Relax-SID
    Send multicast attributes
    Extended Nexthop Encoding
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 7, replicated: 7
  All neighbor are assigned to sub-group(s)
    Neighbors in sub-group: 0.3, Filter-Groups num:1
     Neighbors in filter-group: 0.3(RT num: 0)
      192::4
```

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

```
Router#show bgp 158.158.58.1/32
BGP routing table entry for 158.158.58.1/32
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                175          175
Last Modified: Dec 13 11:38:31.000 for 00:00:04
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
```

```
    192::4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  60
    16.16.16.3 from 16.16.16.3 (16.16.16.3)
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 175
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Not advertised to any peer
  70
    17.17.17.3 from 17.17.17.3 (17.17.17.3)
      Origin IGP, localpref 100, valid, external, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)
```

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

```
Router#show bgp 157.157.57.1/32
BGP routing table entry for 157.157.57.1/32
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                172          172
    SRv6-VPN SID: cafe:1:1:2:42::/128
    Format: base
Last Modified: Dec 13 11:38:31.000 for 00:02:09
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Path #1: Received by speaker 0
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  50
    15.15.15.3 from 15.15.15.3 (15.15.15.3)
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 172
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Not advertised to any peer
  60
    16.16.16.3 from 16.16.16.3 (16.16.16.3)
      Origin IGP, localpref 100, valid, external, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)
```

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

Example:

```
Router#show bgp 157.157.57.1/32 detail
BGP routing table entry for 157.157.57.1/32
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                172          172
    SRv6-VPN SID: cafe:1:1:2:42::/128
    Format: base
    Alloc Mode/Locator ID: per-vrf/2
```

```
    Flags: 0x00123201+0x61010000+0x00000000; multipath;
Last Modified: Dec 13 11:38:31.000 for 00:04:22
Paths: (2 available, best #1)
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  Path #1: Received by speaker 0
  Flags: 0x3000000001050003+0x00, import: 0x020
  Advertised IPv4 Unicast paths to update-groups (with more than one peer):
    0.2
  Advertised IPv4 Unicast paths to peers (in unique update groups):
    192::4
  50
    15.15.15.3 from 15.15.15.3 (15.15.15.3), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, best, group-best, multipath
      Received Path ID 0, Local Path ID 1, version 172
      Origin-AS validity: (disabled)
  Path #2: Received by speaker 0
  Flags: 0x3000000000010003+0x00, import: 0x020
  Not advertised to any peer
  60
    16.16.16.3 from 16.16.16.3 (16.16.16.3), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, multipath
      Received Path ID 0, Local Path ID 0, version 0
      Origin-AS validity: (disabled)


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

SRv6 Provider Edge Lite support

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

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

Table 2: Feature History Table

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

Benefits of SRv6 Provider Edge Lite support

- Lightweight-PE implementation: It allows for a simplified PE role by only supporting VPN de-multiplexing behaviors and not VPN encapsulation.

- Efficient traffic steering: It effectively steers SRv6-encapsulated traffic across an SR-MPLS backbone after performing a VPN lookup.
- High availability: Achieved through the use of Anycast SRv6 locators, ensuring that traffic can still be handled by other nodes if a specific SRv6 PE lite node fails.
- SLA-Driven transport: It enables the provision of desired transport Service Level Agreements (SLAs) to applications by leveraging BGP VPN overlay routes with color extended communities and SR-TE policies.
- Interoperability: It facilitates the carrying of SRv6-encapsulated traffic over IP-only metro domains and SR-MPLS backbones, bridging different network technologies and domains.

How SRv6 Provider Edge Lite processes traffic

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

Summary

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

Workflow

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

1. Control Plane Setup and Signaling:
   - SRv6 PE Lite nodes are configured with SRv6 locators and explicit service de-multiplexing end-point behaviors (e.g., End.DT46 functions).
   - Data center gateways advertise prefixes into the backbone via multiprotocol BGP, including color extended communities to indicate desired transport SLAs.
   - SRv6 PE Lite nodes acting as SR-TE head-ends establish SR policies associated with specific colors and egress PE end-points for steering.
2. Ingress Data Center Gateway Encapsulation:
   - Data center gateways receive traffic destined for a remote data center.
   - The Data Center Gateway encapsulates this traffic into an outer IPv6 header, where the destination address is an SRv6 network program (e.g., `FCBB:BB00:100:FE01::`) determined by a gateway controller for a desired transport SLA.
3. Metro Domain Transport:

   Transit nodes in the IP-only metro domain perform a longest-prefix-match lookup for the outer IPv6 destination address and forward the packet along the shortest path to the SRv6 PE Lite node.
4. SRv6 PE Lite Decapsulation and VPN Lookup:
   - The SRv6 PE Lite node receives the encapsulated traffic.

- The SRv6 PE Lite node removes the SRv6 encapsulation and performs a lookup of the original encapsulated packet's IP destination address in the routing table of the corresponding MPLS VPN (e.g., VRF 200).

5. SRv6 PE Lite Label Imposition and Steering:
   - Based on the VPN lookup and associated SLA, the SRv6 PE Lite node imposes the necessary MPLS VPN label and additional transport labels/SIDs.
   - This steers the traffic over the native LSP path for best-effort traffic or over an SR policy path for SLA-specific traffic.
6. SR MPLS Backbone Transport

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

Configuration for SRv6 PE Lite Node

Procedure

Step 1 Configure SRv6.

Example:

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

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

Example:

```
router isis core
 address-family ipv4 unicast
  metric-style wide level 1
  router-id Loopback0
```

```
  segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16011
  !
 !
!
```

Step 3 Configure interface Loopback0.

Example:

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

Step 4 Configure the SR policy.

Example:

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

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

Example:

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

```
end-set
!
route-policy SET-COLOR-1000
  set extcommunity color COLOR-1000
end-policy
!
```

Step 6 Configure BGP.

Example:

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

Explicit End DT46 SRv6 SIDs

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

Table 3: Feature History Table

Feature Name	Release	Description
Support for End.DT46 SRv6 Endpoint Behavior	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release	Description
Support for End.DT46 SRv6 Endpoint Behavior	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Support for End.DT46 SRv6 Endpoint Behavior	Release 7.5.3	This feature adds support for the “Endpoint with decapsulation and specific IP table lookup” SRv6 end-point behavior (End.DT46). The End.DT46 behavior is used for dual-stack L3VPNs. This behavior is equivalent to the single per-VRF VPN label (for IPv4 and IPv6) in MPLS.
Support for Explicit End.DT46 SRv6 SIDs	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
Support for Explicit End.DT46 SRv6 SIDs	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release	Description
Support for Explicit End.DT46 SRv6 SIDs	Release 7.5.3	This feature allows you to configure explicit SIDs associated with SRv6-based L3VPN/Internet BGP services. In previous releases, these SIDs were only allocated dynamically by BGP. Explicit End.DT46 SRv6 SIDs are persistent over reloads and restarts. The feature introduces these changes: CLI: • The **segment-routing srv6 static endpoint sid prefix behavior end-udt46** command mode is introduced.

Multiple explicit uDT46 IDs allocation examples

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

Example: Explicit uDT46 (LIB)

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

Example: Explicit uDT46 (W-LIB)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Configure explicit End DT46 SRv6 SIDs

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

Procedure

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

Use the **allocation-context vrf** command to associate an explicit uDT46 ID allocated from the LIB or W-LIB to a VRF. Use the **forwarding** keyword if VRF-lite (the deployment of VRFs without BGP/MPLS) is enabled.

Example:

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

Configure explicit SRv6 uSID allocation start range

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

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

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

Before you begin

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

Procedure

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

Example:

```
Router# show segment-routing srv6 manager
Fri Sep  9 18:31:06.033 UTC
Parameters:
  SRv6 Enabled: Yes
  SRv6 Operational Mode: None
```

```
  Encapsulation:
    Source Address:
      Configured: ::
      Default: ::
    Hop-Limit: Default
    Traffic-class: Default
  SID Formats:
    f3216 <32B/16NFA> (2)
      uSID LIB Range:
        LIB Start   : 0xe000
        ELIB Start  : 0xe064 (configured)
      uSID WLIB Range:
        EWLIB Start : 0xfff0 (configured)

. . .
```

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

Example:

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

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

Example:

```
RP/0/RP0/CPU0:ios# show segment-routing srv6 manager
Fri Sep  9 18:31:06.033 UTC
Parameters:
  SRv6 Enabled: Yes
  SRv6 Operational Mode: None
  Encapsulation:
    Source Address:
      Configured: ::
      Default: ::
    Hop-Limit: Default
    Traffic-class: Default
  SID Formats:
    f3216 <32B/16NFA> (2)
      uSID LIB Range:
        LIB Start   : 0xe000
        ELIB Start  : 0xe064 (configured)
      uSID WLIB Range:
        EWLIB Start : 0xfff0 (configured)

. . .
```

SRv6 SID Information in BGP-LS Reporting

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

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

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 SID Information in BGP-LS Reporting	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.
SRv6 SID Information in BGP-LS Reporting	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can use BGP Link-State (BGP-LS) to report the domain topology with nodes, links, and prefixes. This feature supports reporting SRv6 Segment Identifier (SID) Network Layer Reachability Information (NLRI). *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Network Layer Reachability Information (NLRI)

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

For example, NLRI is used to describe various types of SRv6-specific information reported by BGP-LS. These NLRI has been added to the BGP-LS protocol to support SRv6:

- Node NLRI: SRv6 Capabilities, SRv6 MSD types
- Link NLRI: End.X, LAN End.X, and SRv6 MSD types
- Prefix NLRI: SRv6 Locator
- SRv6 SID NLRI (for SIDs associated with the node): Endpoint Function, BGP-EPE Peer Node/Set

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

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

Configure SRv6 SID Information in BGP-LS Reporting

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

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

Procedure

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

Example:

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

Path maximum transmission unit discovery for SRv6 encapsulated packets

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

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

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
Path MTU discovery for SRv6 Packets on Ingress Provider Edge (PE) Routers, Egress (PE) Routers, and P Role Transit Nodes	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) You can measure and monitor the packet loss information when one SRv6-enabled router sends an oversized packet to another. This functionality enables a router to send an ICMP error message to the source in such cases, prompting the sender to resend a packet whose size is within the MTU value, thus ensuring the packet moves ahead. The feature is critical for SRv6-enabled routers as these routers do not support packet fragmentation. Previously, a router dropped oversized packets without notifying the source, resulting in packet loss. This feature is now supported on: • 8011-4G24Y4H-I
Path MTU discovery for SRv6 Packets on Ingress Provider Edge (PE) Routers, Egress (PE) Routers, and P Role Transit Nodes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Path MTU discovery for SRv6 Packets on Ingress Provider Edge (PE) Routers, Egress (PE) Routers, and P Role Transit Nodes	Release 24.1.1	You can measure and monitor the packet loss information when one SRv6-enabled router sends an oversized packet to another. This functionality enables a router to send an ICMP error message to the source in such cases, prompting the sender to resend a packet whose size is within the MTU value, thus ensuring the packet moves ahead. The feature is critical for SRv6-enabled routers as these routers do not support packet fragmentation. Previously, a router dropped oversized packets without notifying the source, resulting in packet loss. The **hw-module** configuration is not required, this feature is enabled by default.

VRF-to-VRF route leaking in SRv6 core

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

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

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
VRF-to-VRF route leaking in SRv6 core	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) This feature is now supported on: • 8712-MOD-M • 8011-4G24Y4H-I
VRF-to-VRF route leaking in SRv6 core	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: Q200, P100])(select variants only*) VRF-to-VRF route leaking enables sharing of routes between VRFs while maintaining their isolation. This feature allows the source VRF to send leaked routes to remote PEs or Route Reflectors (RRs) across an SRv6 core network, similar to an MPLS core network, enabling communication between different service tenants or administrative domains without compromising VRF isolation. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

SRv6 VRF-to-VRF route leaking capabilities

SRv6 VRF-to-VRF route leaking extends MPLS-based VRF route leaking functionalities to SRv6. This feature leverages SRv6 flexibility for path selection and optimization, allowing you to configure the destination VRF to send leaked routes to a remote Provider Edge (PE) or Route Reflector (RR) across an SRv6 core. It enables

communication between VRFs in an SRv6-enabled environment while maintaining control over routing and traffic engineering decisions.

Benefits of VRF-to-VRF route leaking in SRv6

The key benefits of the feature are:

- Improved traffic management: The feature allows routes in the source VRF to use SRv6 SIDs from a best-effort locator and routes in the destination VRF to use SIDs from a low-latency locator. This setup enables differentiated traffic treatment in the SRv6 core.
- Enhanced flexibility: The feature leaks routes between VRFs and advertises them as VPNv4 or VPNv6 or EVPN RT5 prefixes to remote PE routers, providing better flexibility in managing network traffic and inter-VRF communication.
- Scalability: The feature dynamically leaks routes that help to scale the network by automating the redistribution process between VRFs.
- Security and isolation: The feature uses route targets and policies to control route leaking, ensuring that it only occurs between intended VRFs, maintaining both security and isolation.

Limitations for VRF-to-VRF route leaking in SRv6

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

How SRv6 VRF-to-VRF route leaking works

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

Summary

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

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

- BGP VPNv4/VPNv6/EVPN RT5: The BGP extensions used by the destination VRF to advertise the prefixes and SIDs.

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

Workflow

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

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

Configure VRF-to-VRF route leaking in SRv6

Procedure

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

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

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

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

```
Route(config)#prefix-set allowed-leaked-route
Route(config-pfx)#192.168.1.0/32
Router(config-pfx)#end-set
Router(config)#route-policy export-policy
Router(config-rpl)#if destination in allowed-leaked-route then
Router(config-rpl-if)#set extcommunity rt 1:12
Router(config-rpl-if)#endif
Router(config-rpl)#end-policy
Router(config)#commit
```

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

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

Example:

```
Router(config)#vrf vrf-ef
Router(config-vrf)#address-family ipv4 unicast
Router(config-vrf-af)#import route-target 2:2
Router(config-vrf-af)#import route-target 1:12
Router(config-vrf-af)#export route-target 2:2
```

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

Example:

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

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

Example:

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

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

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

Example:

```
Router#show bgp vrf vrf-be 192.168.1.0/32 detail
Mon Aug 19 14:06:22.668 UTC
BGP routing table entry for 192.168.1.0/32, Route Distinguisher: 1.1.1.1:11
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker            3434714      3434714
    SRv6-VPN SID: fc00:1:4:fff0:7d1::/80
    Format: f3216
    Alloc Mode/Locator ID: per-vrf/1
    Flags: 0x00143001+0x01000000+0x00000000
Last Modified: Aug 19 09:53:33.351 for 04:12:49
Paths: (1 available, best #1)
```

```
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Flags: 0x3000000005040003+0x00, import: 0x31f
  Advertised to update-groups (with more than one peer):
    0.2
  4
    100.4.0.1 from 100.4.0.1 (193.0.0.1), if-handle 0x00000000
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate
      Received Path ID 0, Local Path ID 1, version 3434714
      Extended community: RT:1:10 RT:1:12
```

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

Example:

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

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

```
Router#show bgp vpnv4 unicast rd 1.1.1.1:21 192.168.1.0/32 detail
Fri Dec 13 13:21:29.136 PST
BGP routing table entry for 192.168.1.0/32, Route Distinguisher: 1.1.1.1:21
Versions:
  Process           bRIB/RIB   SendTblVer
  Speaker                690          690
    Flags: 0x00040028+0x00000000+0x00000000
Last Modified: Dec 13 13:14:37.000 for 00:06:52
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x2000000025060005+0x00, import: 0x31f
  Not advertised to any peer
  10
    192::1 (metric 30) from 192::4 (192.168.0.1), if-handle 0x00000000
      Received Label 0x7d10
      Origin IGP, metric 0, localpref 100, valid, internal, best, group-best, import-candidate,
not-in-vrf
      Received Path ID 1, Local Path ID 1, version 690
```

```
     Extended community:RT:2:2
     Originator: 192.168.0.1, Cluster list: 192.168.1.4
     PSID-Type:L3, SubTLV Count:1, R:0x00,
      SubTLV:
T:1(Sid information), Sid:fcc00:2:4:fff0::(Transposed), F:0x00, R2:0x00, Behavior:61, R3:0x00, SS-TLV
Count:1
        SubSubTLV:
         T:1(Sid structure):
          Length [Loc-blk,Loc-node,Func,Arg]:[32,16,32,0], Tpose-len:16, Tpose-offset:64
```

Dual-stack with SRv6 unicast and IPv4 multicast core

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

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

Table 7: Feature History Table

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

Traffic distribution in dual-stack environments using SRv6 and IPv4

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

Benefits of dual-stack with SRv6 unicast and IPv4 multicast

The key benefits of the feature are:

- Combines the benefits of both IPv4 and SRv6 protocols, by providing a streamlined and optimized network experience for both unicast and multicast communication.

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

Limitations for dual-stack with SRv6 and IPv4

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

Enable dual-stack with SRv6 unicast and IPv4 multicast core

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

Procedure

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

The *as-number* range is 1–65535.

Example:

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

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

Example:

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

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

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

Example:

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

```
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 mvpn
Router(config-bgp)#address-family ipv4 mvpn
```

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

Example:

```
router bgp 101
 nsr
 mvpn
 bgp router-id 10.10.10.1
 bgp graceful-restart
 segment-routing srv6
 !
 address-family ipv4 multicast
  redistribute connected
 !
 address-family ipv6 unicast
  segment-routing srv6
  !
  redistribute connected
 !
 address-family ipv4 mvpn
 !
 address-family ipv6 mvpn
 !
!
```