Configure Segment Routing for BGP

Border Gateway Protocol (BGP) is an Exterior Gateway Protocol (EGP) that allows you to create loop-free inter-domain routing between autonomous systems. An autonomous system is a set of routers under a single technical administration. Routers in an autonomous system can use multiple Interior Gateway Protocols (IGPs) to exchange routing information inside the autonomous system and an EGP to route packets outside the autonomous system.

This module provides the configuration information used to enable Segment Routing for BGP.

Note For additional information on implementing BGP on your router , see the *Implementing BGP* module in the *Routing Configuration Guide*.

- Segment Routing for BGP, on page 1
- Configure BGP Prefix Segment Identifiers, on page 2
- Segment Routing Egress Peer Engineering, on page 3
- Configure BGP Link-State, on page 8
- Configurable Filters for IS-IS Advertisements to BGP-Link State, on page 13
- Use Case: Configuring SR-EPE and BGP-LS, on page 14
- Configure BGP Proxy Prefix SID, on page 17
- SRv6 double recursion for multilayer BGP underlay , on page 24

Segment Routing for BGP

In a traditional BGP-based data center (DC) fabric, packets are forwarded hop-by-hop to each node in the autonomous system. Traffic is directed only along the external BGP (eBGP) multipath ECMP. No traffic engineering is possible.

In an MPLS-based DC fabric, the eBGP sessions between the nodes exchange BGP labeled unicast (BGP-LU) network layer reachability information (NLRI). An MPLS-based DC fabric allows any leaf (top-of-rack or border router) in the fabric to communicate with any other leaf using a single label, which results in higher packet forwarding performance and lower encapsulation overhead than traditional BGP-based DC fabric. However, since each label value might be different for each hop, an MPLS-based DC fabric is more difficult to troubleshoot and more complex to configure.

BGP has been extended to carry segment routing prefix-SID index. BGP-LU helps each node learn BGP prefix SIDs of other leaf nodes and can use ECMP between source and destination. Segment routing for BGP

simplifies the configuration, operation, and troubleshooting of the fabric. With segment routing for BGP, you can enable traffic steering capabilities in the data center using a BGP prefix SID.

Note BGP flowspec support with SRv6 - Limitations

List of BGP address families interacts with SRv6. There are some supported and unsupported BGP address family for the interaction with SRv6.

Supported address family:

- address-families ipv6.

Unsupported address families:

- address-families ipv4
- vpnv4
- vpnv6

Configure BGP Prefix Segment Identifiers

Segments associated with a BGP prefix are known as BGP prefix SIDs. The BGP prefix SID is global within a segment routing or BGP domain. It identifies an instruction to forward the packet over the ECMP-aware best-path computed by BGP to the related prefix. The BGP prefix SID is manually configured from the segment routing global block (SRGB) range of labels.

Each BGP speaker must be configured with an SRGB using the **segment-routing global-block** command. See the About the Segment Routing Global Block section for information about the SRGB.

Note You must enable SR and explicitly configure the SRGB before configuring SR BGP. The SRGB must be explicitly configured, even if you are using the default range (16000 – 23999). BGP uses the SRGB and the index in the BGP prefix-SID attribute of a learned BGP-LU advertisement to allocate a local label for a given destination.

If SR and the SRGB are enabled after configuring BGP, then BGP is not aware of the SRGB, and therefore it allocates BGP-LU local labels from the dynamic label range instead of from the SRGB. In this case, restart the BGP process in order to allocate BGP-LU local labels from the SRGB.

Note Because the values assigned from the range have domain-wide significance, we recommend that all routers within the domain be configured with the same range of values.

To assign a BGP prefix SID, first create a routing policy using the **set label-index** *index* attribute, then associate the index to the node.

Example

The following example shows how to configure the SRGB, create a BGP route policy using a $SID parameter and **set label-index** attribute, and then associate the prefix-SID index to the node.

```
RP/0/RP0/CPU0:router(config)# segment-routing global-block 16000 23999

RP/0/RP0/CPU0:router(config)# route-policy SID($SID)
RP/0/RP0/CPU0:router(config-rpl)# set label-index $SID
RP/0/RP0/CPU0:router(config-rpl)# end policy

RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 10.1.1.1
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# network 10.1.1.3/32 route-policy SID(3)
RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all
RP/0/RP0/CPU0:router(config-bgp-af)# commit
RP/0/RP0/CPU0:router(config-bgp-af)# end


RP/0/RP0/CPU0:router# show bgp 10.1.1.3/32
BGP routing table entry for 10.1.1.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 74          74
    Local Label: 16003
Last Modified: Sep 29 19:52:18.155 for 00:07:22
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  3
    99.3.21.3 from 99.3.21.3 (10.1.1.3)
      Received Label 3
      Origin IGP, metric 0, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 1, version 74
      Origin-AS validity: not-found
      Label Index: 3
```

Segment Routing Egress Peer Engineering

Segment routing egress peer engineering (EPE) uses a controller to instruct an ingress provider edge, or a content source (node) within the segment routing domain, to use a specific egress provider edge (node) and a specific external interface to reach a destination. BGP peer SIDs are used to express source-routed inter-domain paths.

Below are the BGP-EPE peering SID types:

- PeerNode SID—To an eBGP peer. Pops the label and forwards the traffic on any interface to the peer.
- PeerAdjacency SID—To an eBGP peer via interface. Pops the label and forwards the traffic on the related interface.
- PeerSet SID—To a set of eBGP peers. Pops the label and forwards the traffic on any interface to the set of peers. All the peers in a set might not be in the same AS.

Multiple PeerSet SIDs can be associated with any combination of PeerNode SIDs or PeerAdjacency SIDs.

The controller learns the BGP peer SIDs and the external topology of the egress border router through BGP-LS EPE routes. The controller can program an ingress node to steer traffic to a destination through the egress node and peer node using BGP labeled unicast (BGP-LU).

EPE functionality is only required at the EPE egress border router and the EPE controller.

Usage Guidelines and Limitations

- When enabling BGP EPE, you must enable MPLS encapsulation on the egress interface connecting to the eBGP peer. This can be done by enabling either BGP labeled unicast (BGP-LU) address family or MPLS static for the eBGP peer.

  For information about BGP-LU, refer to the "Implementing BGP" chapter in the *BGP Configuration Guide*.

  For information about MPLS static, refer to the "Implementing MPLS Static Labeling chapter in the *MPLS Configuration Guide*.

Configure Segment Routing Egress Peer Engineering

This task explains how to configure segment routing EPE on the EPE egress node.

Procedure

	Command or Action	Purpose
Step 1	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 1`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 2	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.1.3`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 3	**remote-as** *as-number* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 3`	Creates a neighbor and assigns a remote autonomous system number to it.

	Command or Action	Purpose
Step 4	**egress-engineering** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# egress-engineering`	Configures the egress node with EPE for the eBGP peer.

Step 5	**exit**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr)# exit`
	`RP/0/RP0/CPU0:router(config-bgp)# exit`
	`RP/0/RP0/CPU0:router(config)#`	

Step 6	**mpls static** **Example:** `RP/0/RP0/CPU0:router(config)# mpls static`	Configure MPLS static on the egress interface connecting to the eBGP peer.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-mpls-static)# interface GigabitEthernet0/0/1/2`	Specifies the egress interface connecting to the eBGP peer.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Example

Running Config:

```
router bgp 1
 neighbor 192.168.1.3
  remote-as 3
  egress-engineering
  !
 !
mpls static
 interface GigabitEthernet0/0/1/2
 !
!
```

Configuring Manual BGP-EPE Peering SIDs

Configuring manual BGP-EPE Peer SIDs allows for persistent EPE label values. Manual BGP-EPE SIDs are advertised through BGP-LS and are allocated from the Segment Routing Local Block (SRLB). See Configure Segment Routing Global Block and Segment Routing Local Block for information about the SRLB.

Each PeerNode SID, PeerAdjacency SID, and PeerSet SID is configured with an index value. This index serves as an offset from the configured SRLB start value and the resulting MPLS label (SRLB start label + index) is assigned to these SIDs. This label is used by CEF to perform load balancing across the individual BGP PeerSet SIDs, BGP PeerNode SID, or ultimately across each first-hop adjacency associated with that BGP PeerNode SID or BGP PeerSet SID.

Configuring Manual PeerNode SID

Each eBGP peer will be associated with a PeerNode SID index that is configuration driven.

```
RP/0/0/CPU0:PE1(config)# router bgp 10
RP/0/0/CPU0:PE1(config-bgp)# neighbor 10.10.10.2
RP/0/0/CPU0:PE1(config-bgp-nbr)# remote-as 20
RP/0/0/CPU0:PE1(config-bgp-nbr)# egress-engineering
RP/0/0/CPU0:PE1(config-bgp-nbr)# peer-node-sid index 600
```

Configuring Manual PeerAdjacency SID

Any first-hop for which an adjacency SID is configured needs to be in the resolution chain of at least one eBGP peer that is configured for egress-peer engineering. Otherwise such a kind of “orphan” first-hop with regards to BGP has no effect on this feature. This is because BGP only understands next-hops learnt by the BGP protocol itself and in addition only the resolving IGP next-hops for those BGP next-hops.

```
RP/0/0/CPU0:PE1(config)# router bgp 10
RP/0/0/CPU0:PE1(config-bgp)# adjacencies
RP/0/0/CPU0:PE1(config-bgp-adj)# 10.1.1.2
RP/0/0/CPU0:PE1(config-bgp-adj)# adjacency-sid index 500
```

Configuring Manual PeerSet SID

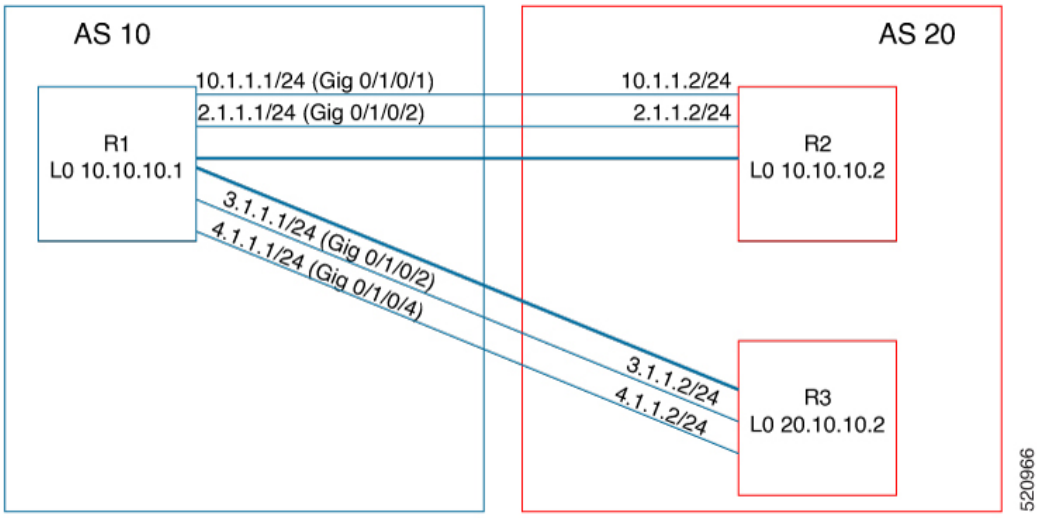
The PeerSet SID is configured under global Address Family. This configuration results in the creation of a Peer-Set SID EPE object.

```
RP/0/0/CPU0:PE1(config)# router bgp 10
RP/0/0/CPU0:PE1(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:PE1(config-bgp-afi)# peer-set-id 1
RP/0/0/CPU0:PE1(config-bgp-peer-set)# peer-set-sid 300
```

Example

Topology

The example in this section uses the following topology.



In this example, BGP-EPE peer SIDs are allocated from the default SRLB label range (15000 – 15999). The BGP-EPE peer SIDs are configured as follows:

- PeerNode SIDs to 10.10.10.2 with index 600 (label 15600), and for 20.10.10.2 with index 700 (label 15700)
- PeerAdj SID to link 10.1.1.2 with index 500 (label 15500)
- PeerSet SID 1 to load balance over BGP neighbors 10.10.10.1 and 20.10.10.2 with SID index 300 (label 15300)
- PeerSet SID 2 to load balance over BGP neighbor 20.10.10.2 and link 10.1.1.2 with SID index 400 (label 15400)

Configuration on R1

```
router bgp 10
 address-family ipv4 unicast
  peer-set-id 1
   peer-set-sid index 300
  !
  peer-set-id 2
   peer-set-sid index 400
  !
 !
 adjacencies
  10.1.1.2
   adjacency-sid index 500
   peer-set 2
  !
 !
 neighbor 10.10.10.2
  remote-as 20
  egress-engineering
  peer-node-sid index 600
  peer-set 1
```

```
!
neighbor 20.10.10.2
 egress-engineering
 peer-node-sid index 700
 peer-set 1
 peer-set 2
!
```

To further show the load balancing of this example:

- 15600 is load balanced over {10.1.1.1 and 2.1.1.1}
- 15700 is load balanced over {3.1.1.1 and 4.1.1.1}
- 15500 is load balanced over {10.1.1.1}
- 15300 is load balanced over {10.1.1.1, 2.1.1.1, 3.1.1.1 and 4.1.1.1}
- 15400 is load balanced over {10.1.1.1, 3.1.1.1 and 4.1.1.1}

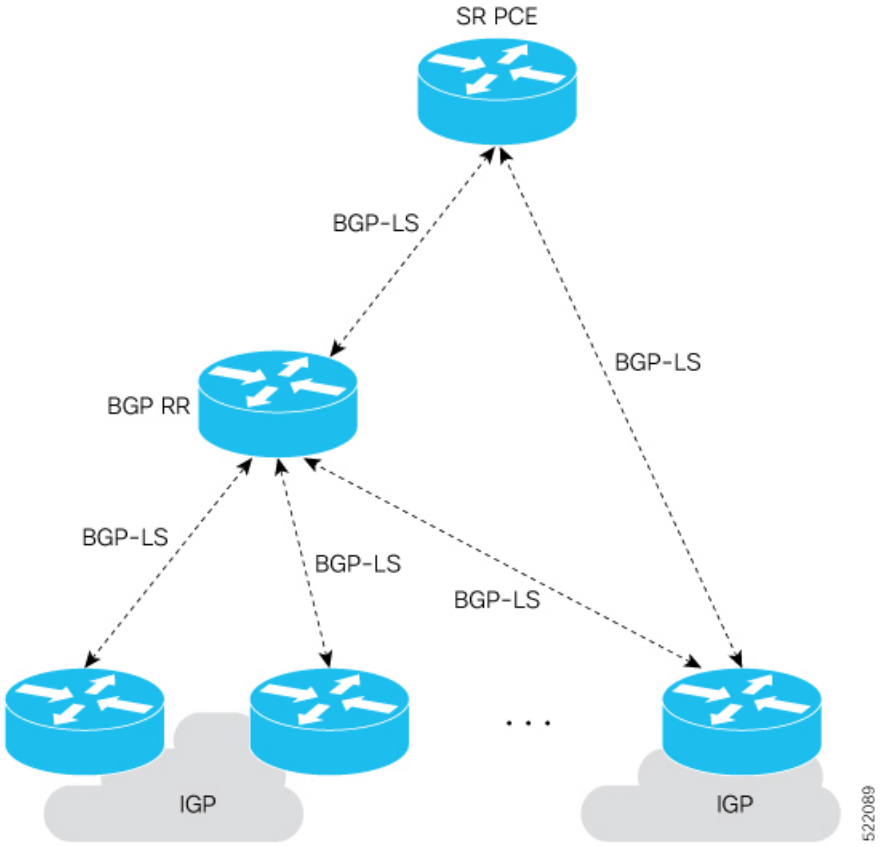
Configure BGP Link-State

BGP Link-State (LS) is an Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) originally defined to carry interior gateway protocol (IGP) link-state information through BGP. The BGP Network Layer Reachability Information (NLRI) encoding format for BGP-LS and a new BGP Path Attribute called the BGP-LS attribute are defined in RFC7752. The identifying key of each Link-State object, namely a node, link, or prefix, is encoded in the NLRI and the properties of the object are encoded in the BGP-LS attribute.

The BGP-LS Extensions for Segment Routing are documented in RFC9085.

BGP-LS applications like an SR Path Computation Engine (SR-PCE) can learn the SR capabilities of the nodes in the topology and the mapping of SR segments to those nodes. This can enable the SR-PCE to perform path computations based on SR-TE and to steer traffic on paths different from the underlying IGP-based distributed best-path computation.

The following figure shows a typical deployment scenario. In each IGP area, one or more nodes (BGP speakers) are configured with BGP-LS. These BGP speakers form an iBGP mesh by connecting to one or more route-reflectors. This way, all BGP speakers (specifically the route-reflectors) obtain Link-State information from all IGP areas (and from other ASes from eBGP peers).



Usage Guidelines and Limitations

- BGP-LS supports IS-IS and OSPFv2.
- The identifier field of BGP-LS (referred to as the Instance-ID) identifies the IGP routing domain where the NLRI belongs. The NLRIs representing link-state objects (nodes, links, or prefixes) from the same IGP routing instance must use the same Instance-ID value.
- When there is only a single protocol instance in the network where BGP-LS is operational, we recommend configuring the Instance-ID value to **0**.
- Assign consistent BGP-LS Instance-ID values on all BGP-LS Producers within a given IGP domain.
- NLRIs with different Instance-ID values are considered to be from different IGP routing instances.
- Unique Instance-ID values must be assigned to routing protocol instances operating in different IGP domains. This allows the BGP-LS Consumer (for example, SR-PCE) to build an accurate segregated multi-domain topology based on the Instance-ID values, even when the topology is advertised via BGP-LS by multiple BGP-LS Producers in the network.
- If the BGP-LS Instance-ID configuration guidelines are not followed, a BGP-LS Consumer may see duplicate link-state objects for the same node, link, or prefix when there are multiple BGP-LS Producers deployed. This may also result in the BGP-LS Consumers getting an inaccurate network-wide topology.

- The following table defines the supported extensions to the BGP-LS address family for carrying IGP topology information (including SR information) via BGP. For more information on the BGP-LS TLVs, refer to Border Gateway Protocol - Link State (BGP-LS) Parameters.

Table 1: IOS XR Supported BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
256	Local Node Descriptors	X	X	—
257	Remote Node Descriptors	X	X	—
258	Link Local/Remote Identifiers	X	X	—
259	IPv4 interface address	X	X	—
260	IPv4 neighbor address	X		
261	IPv6 interface address	X	—	—
262	IPv6 neighbor address	X	—	—
263	Multi-Topology ID	X	—	—
264	OSPF Route Type	—	X	—
265	IP Reachability Information	X	X	—
266	Node MSD TLV	X	X	—
267	Link MSD TLV	X	X	—
512	Autonomous System	—	—	X
513	BGP-LS Identifier	—	—	X
514	OSPF Area-ID	—	X	—
515	IGP Router-ID	X	X	—
516	BGP Router-ID TLV	—	—	X
517	BGP Confederation Member TLV	—	—	X
1024	Node Flag Bits	X	X	—
1026	Node Name	X	X	—
1027	IS-IS Area Identifier	X	—	—
1028	IPv4 Router-ID of Local Node	X	X	—
1029	IPv6 Router-ID of Local Node	X	—	—
1030	IPv4 Router-ID of Remote Node	X	X	—
1031	IPv6 Router-ID of Remote Node	X	—	—
1034	SR Capabilities TLV	X	X	—
1035	SR Algorithm TLV	X	X	—
1036	SR Local Block TLV	X	X	—

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
1039	Flex Algo Definition (FAD) TLV	X	X	—
1044	Flex Algorithm Prefix Metric (FAPM) TLV	X	X	—
1088	Administrative group (color)	X	X	—
1089	Maximum link bandwidth	X	X	—
1090	Max. reservable link bandwidth	X	X	—
1091	Unreserved bandwidth	X	X	—
1092	TE Default Metric	X	X	—
1093	Link Protection Type	X	X	—
1094	MPLS Protocol Mask	X	X	—
1095	IGP Metric	X	X	—
1096	Shared Risk Link Group	X	X	—
1099	Adjacency SID TLV	X	X	—
1100	LAN Adjacency SID TLV	X	X	—
1101	PeerNode SID TLV	—	—	X
1102	PeerAdj SID TLV	—	—	X
1103	PeerSet SID TLV	—	—	X
1114	Unidirectional Link Delay TLV	X	X	—
1115	Min/Max Unidirectional Link Delay TLV	X	X	—
1116	Unidirectional Delay Variation TLV	X	X	—
1117	Unidirectional Link Loss	X	X	—
1118	Unidirectional Residual Bandwidth	X	X	—
1119	Unidirectional Available Bandwidth	X	X	—
1120	Unidirectional Utilized Bandwidth	X	X	—
1122	Application-Specific Link Attribute TLV	X	X	—
1152	IGP Flags	X	X	—
1153	IGP Route Tag	X	X	—
1154	IGP Extended Route Tag	X	—	—
1155	Prefix Metric	X	X	—
1156	OSPF Forwarding Address	—	X	—
1158	Prefix-SID	X	X	—
1159	Range	X	X	—

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
1161	SID/Label TLV	X	X	—
1170	Prefix Attribute Flags	X	X	—
1171	Source Router Identifier	X	—	—
1172	L2 Bundle Member Attributes TLV	X	—	—
1173	Extended Administrative Group	X	X	—

Exchange Link State Information with BGP Neighbor

The following example shows how to exchange link-state information with a BGP neighbor:

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# neighbor 10.0.0.2
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# address-family link-state link-state
Router(config-bgp-nbr-af)# exit
```

IGP Link-State Database Distribution

A given BGP node may have connections to multiple, independent routing domains. IGP link-state database distribution into BGP-LS is supported for both OSPF and IS-IS protocols in order to distribute this information on to controllers or applications that desire to build paths spanning or including these multiple domains.

To distribute IS-IS link-state data using BGP-LS, use the **distribute link-state** command in router configuration mode.

```
Router# configure
Router(config)# router isis isp
Router(config-isis)# distribute link-state instance-id 32
```

To distribute OSPFv2 link-state data using BGP-LS, use the **distribute link-state** command in router configuration mode.

```
Router# configure
Router(config)# router ospf 100
Router(config-ospf)# distribute link-state instance-id 32
```

Configurable Filters for IS-IS Advertisements to BGP-Link State

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Configurable Filters for IS-IS Advertisements to BGP-Link State	Release 7.10.1	This feature allows you to configure a route map to filter IS-IS route advertisements to BGP-Link State (LS). It also provides a per-area configuration knob to disable IS-IS advertisements for external and propagated prefixes. This configuration of filters hence reduces the amount of redundant data for external and interarea prefixes sent to the BGP - LS clients. The feature introduces **exclude-external**, **exclude-interarea**, and **route-policy** *name* optional keywords in the **distribute link-state** command.

In a large IS-IS network, there are multiple routers in different areas distributing their link-state databases through BGP-LS. In addition, other protocols, such as OSPF do their own BGP-LS reporting and have routes that are redistributed into IS-IS. This can result in substantial amounts of redundant data for external and interarea prefixes which are sent to the BGP-LS clients only to be discarded.

Rather than sending redundant information, this feature provides the option of limiting the prefixes for which IS-IS TLV information is sent to BGP-LS.

There are three options to filter prefix Type-Length-Values (TLVs) that are reported in BGP-LS and the operators can specify these options on a per-level basis:

- **exclude-external**: Omits information for external prefixes that are redistributed from a different protocol or instance. These are identified by the “X” bit set in its Extended Reachability Attribute Flags or the ‘X’ bit of TLVs 236 and 237.
- **exclude-interarea**: Omits information for interarea prefixes and summaries. These are identified by the ‘R’ bit set in their Extended Reachability Attribute Flags or the ‘up or down’ bit set in TLVs 135, 235, 236, and 237.
- **route-policy***name*: Allows specification of a route-policy to provide filtering based on a set of destination prefixes.

The filtering is implemented at the point where the individual prefix TLVs are read from a label-switched path to generate updates to BGP-LS. It does not affect the advertisement of a node or the link information.

Configure Filters for IS-IS Advertisements to BGP-LS

Configuration Example

You can configure any of these filters for IS-IS advertisements to BGP-LS:

```
Router#config
Router(config)#router isis 1
```

```
Router(config-isis)#distribute link-state exclude-external
Router(config-isis)#commit


Router#config
Router(config)#router isis 1
Router(config-isis)#ddistribute link-state exclude-interarea
Router(config-isis)#commit


Router# config
Router(config)# router isis 1
Router(config-isis)#distribute link-state route-policy isis-rp-1
Router(config-isis)#commit
```

Note This feature does not introduce any new failure modes to IS-IS.

Running Configuration

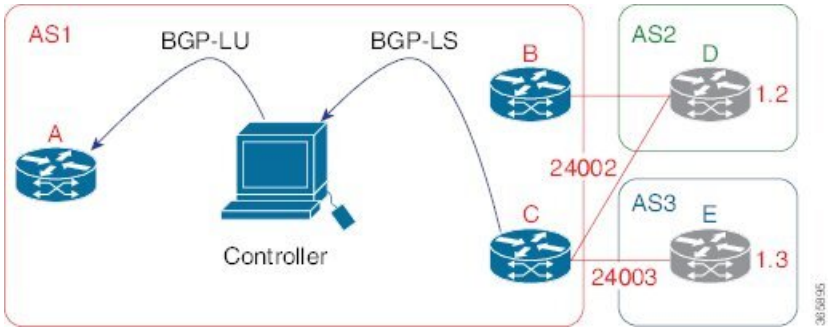
To check the filter for IS-IS advertisements to BGP-LS, you can run the following command:

```
Router# show running-config
router isis 1
 distribute link-state exclude-external
 commit
 !
!


router isis 1
 distribute link-state exclude-interarea
 commit
 !
!


router isis 1
 distribute link-state route-policy isis-rp-1
 commit
 !
!
```

Use Case: Configuring SR-EPE and BGP-LS

In the following figure, segment routing is enabled on autonomous system AS1 with ingress node A and egress nodes B and C. In this example, we configure EPE on egress node C.

Figure 1: Topology



Procedure

Step 1 Configure node C with EPE for eBGP peers D and E.

Example:

```
RP/0/RP0/CPU0:router_C(config)# router bgp 1
RP/0/RP0/CPU0:router_C(config-bgp)# neighbor 192.168.1.3
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# remote-as 3
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# description to E
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# egress-engineering
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_in in
RP/0/RP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_out out
RP/0/RP0/CPU0:router_C(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# exit
RP/0/RP0/CPU0:router_C(config-bgp)# neighbor 192.168.1.2
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# remote-as 2
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# description to D
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# egress-engineering
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_in in
RP/0/RP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_out out
RP/0/RP0/CPU0:router_C(config-bgp-nbr-af)# exit
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# exit
```

Step 2 Configure node C to advertise peer node SIDs to the controller using BGP-LS.

Example:

```
RP/0/RP0/CPU0:router_C(config-bgp)# neighbor 172.29.50.71
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# description to EPE_controller
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# address-family link-state link-state
RP/0/RP0/CPU0:router_C(config-bgp-nbr)# exit
RP/0/RP0/CPU0:router_C(config-bgp)# exit
```

Step 3 Configure MPLS static on the egress interfaces connecting to the eBGP peer.

Example:

```
RP/0/RP0/CPU0:router_C(config)# mpls static
RP/0/RP0/CPU0:router_C(config-mpls-static)# interface TenGigE 0/3/0/0
```

```
RP/0/RP0/CPU0:router_C(config-mpls-static)# interface TenGigE 0/1/0/0
RP/0/RP0/CPU0:router_C(config-mpls-static)# exit
```

Step 4 Commit the configuration.

Example:

```
RP/0/RP0/CPU0:router_C(config)# commit
```

Step 5 Verify the configuration.

Example:

```
RP/0/RP0/CPU0:router_C# show bgp egress-engineering

 Egress Engineering Peer Set: 192.168.1.2/32 (10b87210)
     Nexthop: 192.168.1.2
     Version: 2, rn_version: 2
       Flags: 0x00000002
   Local ASN: 1
  Remote ASN: 2
   Local RID: 10.1.1.3
  Remote RID: 10.1.1.4
   First Hop: 192.168.1.2
        NHID: 3
       Label: 24002, Refcount: 3
     rpc_set: 10b9d408

 Egress Engineering Peer Set: 192.168.1.3/32 (10be61d4)
     Nexthop: 192.168.1.3
     Version: 3, rn_version: 3
       Flags: 0x00000002
   Local ASN: 1
  Remote ASN: 3
   Local RID: 10.1.1.3
  Remote RID: 10.1.1.5
   First Hop: 192.168.1.3
        NHID: 4
       Label: 24003, Refcount: 3
     rpc_set: 10be6250
```

The output shows that node C has allocated peer SIDs for each eBGP peer.

Example:

```
RP/0/RP0/CPU0:router_C# show mpls forwarding labels 24002 24003
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24002  Pop         No ID              Te0/0/0/1    192.168.1.2     0
24003  Pop         No ID              Te0/0/0/2    192.168.1.3     0
```

The output shows that node C installed peer node SIDs in the Forwarding Information Base (FIB).

Configure BGP Proxy Prefix SID

To support segment routing, Border Gateway Protocol (BGP) requires the ability to advertise a segment identifier (SID) for a BGP prefix. A BGP-Prefix-SID is the segment identifier of the BGP prefix segment in a segment routing network. BGP prefix SID attribute is a BGP extension to signal BGP prefix-SIDs. However, there may be routers which do not support BGP extension for segment routing. Hence, those routers also do not support BGP prefix SID attribute and an alternate approach is required.

BGP proxy prefix SID feature allows you to attach BGP prefix SID attributes for remote prefixes learnt from BGP labeled unicast (LU) neighbours which are not SR-capable and propagate them as SR prefixes. This allows an LSP towards non SR endpoints to use segment routing global block in a SR domain. Since BGP proxy prefix SID uses global label values it minimizes the use of limited resources such as ECMP-FEC and provides more scalability for the networks.

BGP proxy prefix SID feature is implemented using the segment routing mapping server (SRMS). SRMS allows the user to configure SID mapping entries to specify the prefix-SIDs for the prefixes. The mapping server advertises the local SID-mapping policy to the mapping clients. BGP acts as a client of the SRMS and uses the mapping policy to calculate the prefix-SIDs.

Configuration Example:

This example shows how to configure the BGP proxy prefix SID feature for the segment routing mapping server.

```
RP/0/RSP0/CPU0:router(config)# segment-routing
RP/0/RSP0/CPU0:router(config-sr)# mapping-server
RP/0/RSP0/CPU0:router(config-sr-ms)# prefix-sid-map
RP/0/RSP0/CPU0:router(config-sr-ms-map)# address-family ipv4
RP/0/RSP0/CPU0:router(config-sr-ms-map-af)# 10.1.1.1/32 10 range 200
RP/0/RSP0/CPU0:router(config-sr-ms-map-af)# 192.168.64.1/32 400 range 300
```

This example shows how to configure the BGP proxy prefix SID feature for the segment-routing mapping client.

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# address-family ip4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# segment-routing prefix-sid-map
```

Verification

These examples show how to verify the BGP proxy prefix SID feature.

```
RP/0/RSP0/CPU0:router# show segment-routing mapping-server prefix-sid-map ipv4 detail
Prefix
10.1.1.1/32
    SID Index:      10
    Range:          200
    Last Prefix:    10.1.1.200/32
    Last SID Index: 209
    Flags:
Number of mapping entries: 1


RP/0/RSP0/CPU0:router# show bgp ipv4 labeled-unicast 192.168.64.1/32

BGP routing table entry for 192.168.64.1/32
```

```
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               117         117
  Local Label: 16400
Last Modified: Oct 25 01:02:28.562 for 00:11:45Paths: (2 available, best #1)
 Advertised to peers (in unique update groups):
   201.1.1.1
 Path #1: Received by speaker 0  Advertised to peers (in unique update groups):
    201.1.1.1
  Local
   20.0.101.1 from 20.0.101.1 (20.0.101.1)      Received Label 61
   Origin IGP, localpref 100, valid, internal, best, group-best, multipath, labeled-unicast

   Received Path ID 0, Local Path ID 0, version 117
  Prefix SID Attribute Size: 7
  Label Index: 1
```

```
RP/0/RSP0/CPU0:router# show route ipv4 unicast 192.68.64.1/32 detail

Routing entry for 192.168.64.1/32
  Known via "bgp 65000", distance 200, metric 0, [ei]-bgp, labeled SR, type internal
  Installed Oct 25 01:02:28.583 for 00:20:09
  Routing Descriptor Blocks
    20.0.101.1, from 20.0.101.1, BGP multi path
      Route metric is 0
      Label: 0x3d (61)
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
   Route version is 0x6 (6)
  Local Label: 0x3e81 (16400)
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 4, Download Version 242
  No advertising protos.


RP/0/RSP0/CPU0:router# show cef ipv4 192.168.64.1/32 detail
192.168.64.1/32, version 476, labeled SR, drop adjacency, internal 0x5000001 0x80 (ptr
0x71c42b40) [1], 0x0 (0x71c11590), 0x808 (0x722b91e0)
 Updated Oct 31 23:23:48.733
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
 Extensions: context-label:16400
  gateway array (0x71ae7e78) reference count 3, flags 0x7a, source rib (7), 0 backups
                [2 type 5 flags 0x88401 (0x722eb450) ext 0x0 (0x0)]
  LW-LDI[type=5, refc=3, ptr=0x71c11590, sh-ldi=0x722eb450]
  gateway array update type-time 3 Oct 31 23:49:11.720
 LDI Update time Oct 31 23:23:48.733
 LW-LDI-TS Oct 31 23:23:48.733
   via 20.0.101.1/32, 0 dependencies, recursive, bgp-ext [flags 0x6020]
    path-idx 0 NHID 0x0 [0x7129a294 0x0]
    recursion-via-/32
    unresolved
     local label 16400
     labels imposed {ExpNullv6}


RP/0/RSP0/CPU0:router# show bgp labels
BGP router identifier 2.1.1.1, local AS number 65000
BGP generic scan interval 60 secs
```

```
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 245
BGP main routing table version 245
BGP NSR Initial initsync version 16 (Reached)
BGP NSR/ISSU Sync-Group versions 245/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop        Rcvd Label      Local Label
*>i10.1.1.1/32        10.1.1.1        3               16010
*> 2.1.1.1/32         0.0.0.0         nolabel         3
*> 192.68.64.1/32     20.0.101.1      2               16400
*> 192.68.64.2/32     20.0.101.1      2               16401
```

BGP-LU Inter-AS Option-C Interworking with LDP and IGP SR-MPLS using Proxy BGP-SR

Table 3: Feature History Table

Feature Name	Release	Description
BGP-LU Inter-AS Option-C Interworking with LDP and IGP SR-MPLS using Proxy BGP-SR	Release 7.3.2	This feature extends the current Proxy BGP-SR functionality by allowing the BGP-LU ASBR router with Proxy BGP-SR configured to also interconnect attached LDP domains. The Proxy BGP-SR feature allows interconnection of IGP SR-MPLS domains and legacy domains via BGP-LU Inter-AS option-C. It provides a prefix-to-SID mapping for BGP-LU prefixes that are learned without a Prefix-SID.

The Proxy BGP-SR feature allows interconnection of IGP SR-MPLS domains and legacy domains via BGP-LU Inter-AS option-C. It provides a prefix-to-SID mapping for BGP-LU prefixes that are learned without a Prefix-SID. This new feature extends the current functionality by allowing the BGP-LU ASBR router (configured with Proxy BGP-SR) to also interconnect attached LDP domains.

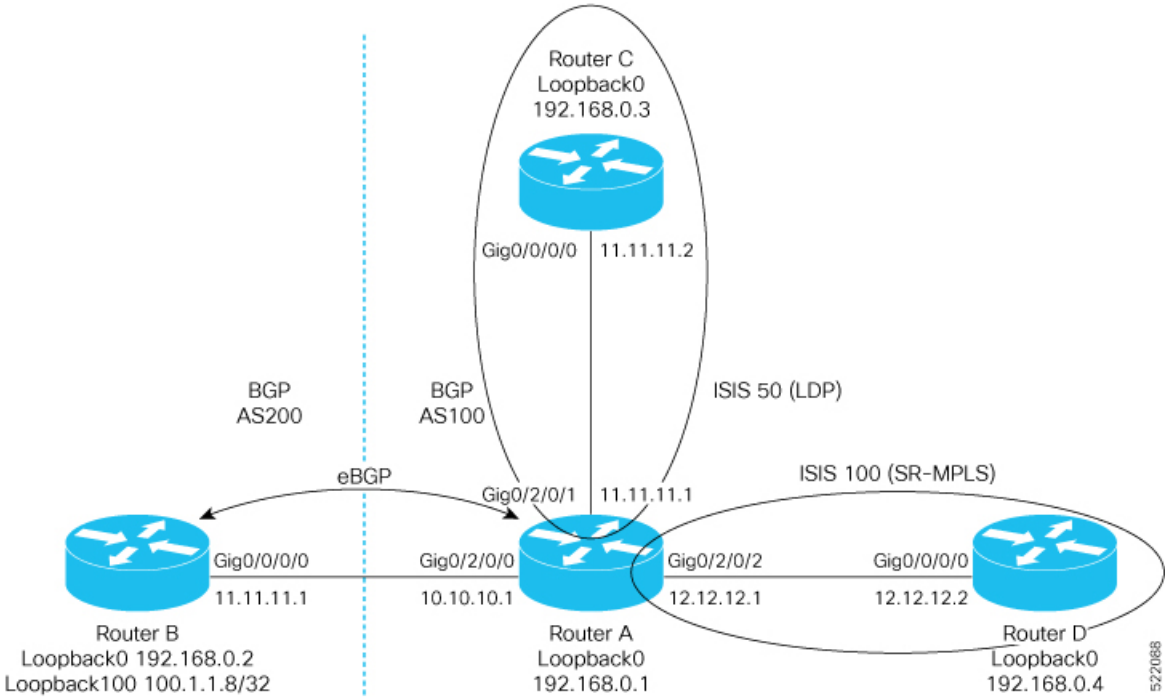
With this enhancement, when performing redistribution from BGP into IGP, LDP would use the same local label assigned by BGP for a prefix learned by BGP-LU. The local label value is based on the SR mapping server configuration (Proxy-BGP SR feature). This behavior allows incoming LDP traffic destined to a redistributed prefix to be switched over to the BGP-LU Inter-AS LSP.

Use Case

In the following figure, Router A does the following:

- Is an ASBR for BGP AS 100 running BGP-LU with BGP AS 200

- Interconnects two IS-IS processes: one running LDP and another running Segment Routing
- Redistributes prefixes learned by BGP-LU from AS 200 into both IS-IS instances
- Runs SR Mapping Server (SRMS) in order to assign mappings to prefixes learned by BGP LU from AS 200 without a prefix SID (proxy BGP-SR) and prefixes learned from the LDP domain



Configuration on Router A - ASBR for AS100

```
prefix-set pfxset-bgplu
  100.1.1.8/32  // The Prefix under test
end-set
!
prefix-set LOOPBACKS
  192.168.0.1,
  192.168.0.2,
  192.168.0.3,
  192.168.0.4,
  192.168.0.8
end-set
!
route-policy Pass
  pass
end-policy
!
route-policy rpl-bgplu
  if destination in pfxset-bgplu then
    pass
  else
    drop
  endif
end-policy
!
route-policy MATCH_LOOPBACKS
```

```
  if destination in LOOPBACKS then
    pass
  else
    drop
  endif
end-policy
!
router static
 address-family ipv4 unicast
  10.10.10.2/32 GigabitEthernet0/2/0/0
 !
!
router isis 50
 is-type level-2-only
 net 49.0001.0000.0000.0001.00
 address-family ipv4 unicast
  metric-style wide
  redistribute bgp 100 route-policy rpl-bgplu  // Redistribute prefixes learned by BGP-LU
into IS-IS LDP domain
 !
 interface Loopback0
  passive
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/2/0/1
  address-family ipv4 unicast
  !
 !
!
router isis 100
 is-type level-2-only
 net 49.0001.0000.0000.0011.00
 distribute link-state
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  redistribute bgp 100 route-policy rpl-bgplu  // Redistribute prefixes learned by BGP-LU
into IS-IS SR domain
  segment-routing mpls
  segment-routing prefix-sid-map advertise-local
 !
 interface Loopback0
  passive
  address-family ipv4 unicast
   prefix-sid index 1
  !
 !
 interface GigabitEthernet0/2/0/2
  address-family ipv4 unicast
  !
 !
!
router bgp 100
 bgp router-id 192.168.0.1
 address-family ipv4 unicast
  segment-routing prefix-sid-map  // SR Proxy SID Configuration
  network 192.168.0.1/32
  redistribute isis 50 route-policy MATCH_LOOPBACKS
  redistribute isis 100 route-policy MATCH_LOOPBACKS
  allocate-label all
 !
 neighbor 10.10.10.2
```

```
  remote-as 200
  address-family ipv4 labeled-unicast
   route-policy Pass in
   route-policy Pass out
  !
 !
!
mpls ldp
 router-id 192.168.0.1
 interface GigabitEthernet0/2/0/1
 !
!
segment-routing
 global-block 16000 23999
 mapping-server
  prefix-sid-map  // SRMS configuration
   address-family ipv4
    100.1.1.8/32 108 range 1  // SRMS mapping - LU prefix 100.1.1.8/32 assigned prefix index
 108
    192.168.0.3/32 3 range 1  // SRMS mapping - LDP prefix Router C assigned prefix index
3
   !
  !
 !
!
```

Configuration on Router B - ASBR for AS200

```
route-policy Pass
  pass
end-policy
!
router static
 address-family ipv4 unicast
  10.10.10.1/32 GigabitEthernet0/0/0/0
 !
!
router bgp 200
 bgp router-id 192.168.0.2
 address-family ipv4 unicast
  network 100.1.1.8/32  // Import/Inject route into BGP
  network 192.168.0.2/32
  allocate-label all
 !
 neighbor 10.10.10.1
  remote-as 100
  address-family ipv4 labeled-unicast
   route-policy Pass in
   route-policy Pass out
  !
 !
!
```

Configuration on Router C in the LDP Domain

```
router isis 50
 is-type level-2-only
 net 49.0001.0000.0000.0003.00
 address-family ipv4 unicast
  metric-style wide
 !
 interface Loopback0
  passive
```

```
  address-family ipv4 unicast
  !
 !
 interface GigabitEthernet0/0/0/0
  address-family ipv4 unicast
  !
 !
!
mpls ldp
 router-id 192.168.0.3
 interface GigabitEthernet0/0/0/0
 !
!
```

Configuration on Router D in the SR IS-IS Domain

```
router isis 100
 is-type level-2-only
 net 49.0001.0000.0000.0004.00
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls
 !
 interface Loopback0
  passive
  address-family ipv4 unicast
   prefix-sid index 4
  !
 !
 interface GigabitEthernet0/0/0/0
  address-family ipv4 unicast
  !
 !
!
segment-routing
!
```

SRv6 double recursion for multilayer BGP underlay

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
SRv6 double recursion for multilayer BGP underlay	Release 24.4.1	The feature introduces support for SRv6 double recursion, where network services such as BGP VPNs (Layer 2 and Layer 3) require multiple resolution layers. Specifically, one routing layer resolves over another before reaching the final destination. Double recursion is achieved by collapsing the underlay, typically involving protocols like IGP or BGP in the packet forwarding chain. This enables three-level load balancing and an even distribution of traffic across multiple layers of the network stack. The feature is supported on the ingress Provider Edge (PE) router. The feature introduces these changes: **CLI:** • **tag-map tag** <*value*> **map forwarding-hierarchy level-2-used-as-nexthop** • The **show cef ipv6** and **show cef ipv4** commands now include the Layer 2 prefix information, which resolve as nexthop Layer 3 prefixes. **YANG Data Models:** • `Cisco-IOS-XR-um-router-rib-cfg:router` (see GitHub, Yang Data Models Navigator) data model.

SRv6 double recursion and load balancing

SRv6 double recursion refers to scenarios where network service such as BGP VPN (Layer 2/Layer 3) requires multiple layers of resolution, specifically where one routing layer resolves over another before reaching its final destination.

The SRv6 double recursion feature is designed to support three-level load balancing by collapsing the underlay, which typically involves protocols like IGP or BGP in the forwarding chain. This involves specific configurations in RIB, BGP, and static routes to indicate IPv4 and IPv6 unicast routes with Layer 2 prefixes, which resolve as nexthop Layer 3 prefixes.

Without this feature, the traffic drops the Layer 3 prefixes leading to packet loss.

Key benefits of SRv6 double recursion

- Support for complex network scenarios: In traditional service provider (SP) networks, BGP VPN services typically resolve directly over IGP reachability, meaning there is a straightforward, single-level resolution path from the service to the IGP underlay. However, with the evolution of networking, especially in data centers, new architectures have emerged where BGP underlay networks are used. In these scenarios, BGP VPN services resolve over a BGP underlay, which in turn resolves over IGP or directly connected routes.
- Support for enhanced load balancing: SRv6 double recursion allows the routing platform to balance traffic more efficiently across all available paths, enhancing overall network performance and reducing congestion. Proper load balancing at each level of recursion helps in distributing traffic evenly, avoiding bottlenecks, and ensuring that packets take optimal paths even in complex topologies.
- Improved network flexibility: Double recursion allows advanced routing solutions in SP and data center networks, accommodating complex use cases beyond single-layer limitations.

Single and double recursion in BGP VPN services

To optimize the network routing efficiency, it is important to understand the resolution processes of BGP VPN services.

- Single Recursion: BGP VPN service → IGP reachability. In a single recursion, the BGP VPN service directly resolves to IGP (Interior Gateway Protocol) reachability. Single recursion is required when a BGP VPN service can directly resolve to an IGP route without any intermediary steps.
- Double Recursion: BGP VPN service → BGP underlay reachability → IGP reachability. In double recursion, the BGP VPN service first resolves over a BGP underlay, which then leads to IGP reachability. Double recursion is necessary in cases where a BGP VPN service cannot resolve directly to an IGP route but instead must traverse a BGP underlay first. This creates a layered resolution process that standard single-level load balancing cannot handle effectively.

Usage guidelines and limitations for SRv6 double recursion

Limitations

You cannot use a combination of BGP-SU (BGP Service Unicast) and BGP-IP paths at Layer 2.

Usage guidelines

The usage guidelines that are listed apply:

- Configure the locator prefix for IPv6 prefixes that are Layer 2 prefixes.
- If the collapsed BGP paths are a combination of IGP IPv6 and SRv6, the router filters out only the IPv6 paths.
- In a collapsed chain scenario with SRv6 as the IGP, the router configures the encapsulation (Encap) ID, provided it supports SRv6 and BGP extensions for SRv6.

Configure SRv6 double recursion for multilayer BGP underlay

Before you begin

- For Layer 2 IPv6 prefixes, set the locator prefix using the **prefix-set** command in RIB.
- Configure the set of IPv4 and IPv6 Layer 2 prefixes, which resolve as nexthop Layer 3 prefixes using the **prefix-set** command.

Procedure

Step 1 Enable the hardware support for BGP-LU to allow the ingress PE router to advertise and forward the MPLS-labelled unicast routes.

Example:

```
Router#config
Router(config)#hw-module profile cef bgplu enable
```

Step 2 Configure the IPv4 and IPv6 unicast routes in the RIB to ensure that the Layer 2 prefixes can be used as nexthop for Layer 3 prefixes.

Example:

```
Router#config
Router(config)#router bgp 100
Router(config-bgp)#address-family ipv4 unicast
Router(config-bgp-af)#table-policy level2-ipv4-policy
Router(config-bgp-af)#exit
Router(config-bgp)#address-family ipv6 unicast
Router(config-bgp-af)#table-policy level2-ipv6-policy
Router(config-bgp-af)#commit
```

Step 3 Assign the required tag to the IPv4 and IPv6 unicast routes to forward the route information to the Forwarding Information Base (FIB).

For the IPv4 and IPv6 routes, the RIB assigns the tags and forwards the route information to the FIB, indicating that these Layer 2 prefixes resolve as nexthop for Layer 3 prefixes.

Example:

IPv4 Layer 2 prefixes:

```
Router(config)#route-policy level2-ipv4-policy
Router(config-rpl)#if destination in level2_prefixes-ipv4 then
```

```
Router(config-rpl-if)#set tag 100
Router(config-rpl-if)#else
Router(config-rpl-else)#pass
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
Router(config)#commit
```

Example:

IPv6 Layer 2 prefixes:

```
Router(config)#route-policy level2-ipv6-policy
Router(config-rpl)#if destination in level2_prefixes-ipv6 then
Router(config-rpl-if)#set tag 100
Router(config-rpl-if)#endif
Router(config-rpl)#if destination in level2_locators then
Router(config-rpl-if)#set locator-prefix
Router(config-rpl-if)#else
Router(config-rpl-else)#pass
Router(config-rpl-else)#endif
Router(config-rpl)#end-policy
```

Step 4 Map the tags in the RIB for the IPv4 and IPv6 unicast routes to ensure that the routes resolve as nexthop for Layer 3 prefixes.

In the following example, the RIB maps all the IPv4 and IPv6 routes tagged with the value 100, which indicates that these routes resolve as nexthop for Layer 3 prefixes. The RIB adds the *FIB_UPDATE_ROUTE_FLAG_EXTN_LVL2_HAS_DEPENDENT* flag when it sends the route update to the FIB.

Example:

```
Router#config
Router(config)#router rib
Router(config-rib)#tag-map tag 100 map forwarding-hierarchy level-2-used-as-nexthop
Router(config-rib)#commit
```

Step 5 Run the **show running-config** command to verify the running configuration.

Example:

```
hw-module profile cef bgplu enable
!
router bgp 100
 address-family ipv4 unicast
  table-policy level2-ipv4-policy
 !
 address-family ipv6 unicast
  table-policy level2-ipv6-policy
 !
!
router rib
 tag-map tag 100 map forwarding-hierarchy level-2-used-as-nexthop
 !
!
```

Step 6 Run the **show cef ipv4** and **show cef ipv6** commands to verify the Layer 2 collapsed prefixes.

In the show output command example, the IPv6 Layer 2 prefixes, which resolve as nexthop for Layer 3 prefixes are collapsed. This is indicated by the **collapsed** keyword in the output. The SRv6 SID lists indicate the different encapsulation layers or hierarchy.

Example:

```
Router#show cef ipv6 2001:DB8:A:B::1/64
Thu Jun  6 12:48:52.399 EDT
2001:DB8:A:B::1/64, version 8, SRv6 Headend, internal 0x1000001 0x0 (ptr 0x63851c98) [1],
0x1400 (0x63851da0), 0x0 (0x638b2128)
 Updated Jun  6 12:41:10.589
 Prefix Len 64, traffic index 0, precedence n/a, priority 0, encap-id 0x11deadbeef
  gateway array (0x61e1a798) reference count 1, flags 0x10, source rib (7), 0 backups
                [2 type 3 flags 0x40008501 (0x63853e38) ext 0x0 (0x0) (collapsed)]
  LW-LDI[type=3, refc=1, ptr=0x63851da0, sh-ldi=0x63853e38]
  gateway array update type-time 1 Jun  6 12:41:10.589
 LDI Update time Jun  6 12:41:10.629
 LW-LDI-TS Jun  6 12:41:10.629
 Accounting: Disabled
   via 2001:DB8::1/128, 1 dependency, recursive [flags 0x3000000]
    path-idx 0 NHID 0x0 [0x63a2c098 0x0]
    next hop 2001:DB8::1/128 via  2001:DB8::1
    SRv6 H.Encaps.Red SID-list {2001:DB8:1:e002::}
      SRv6 H.Insert.Red SID-list {}
      SRv6 H.Insert.Red SID-list {bbbb:bbbb:3:: bbbb:bbbb:4::}
   via 2001:DB8::1/128, 1 dependency, recursive [flags 0x3000000]
    path-idx 1 NHID 0x0 [0x63a2c270 0x0]
    next hop 2001:DB8::1/128 via 2001:DB8::1
    SRv6 H.Encaps.Red SID-list {bbbb:bbbb:2:e002::}
       SRv6 H.Insert.Red SID-list {}

    Load distribution: 0 1 2 2 (refcount 2)

    Hash  OK  Interface                 Address
    0     Y   UNKNOWN intf 0x00000013   10::2
    1     Y   UNKNOWN intf 0x00000014   20::2
    2     Y   UNKNOWN intf 0x00000013   10::2
    3     Y   UNKNOWN intf 0x00000013   10::2
```

The described show output indicates that the IPv4 Layer 2 prefixes, which resolve as nexthop for Layer 3 prefixes are collapsed.

Example:

```
Router#show cef 209.165.201.1 detail
Output received:
Mon Dec  2 08:31:43.765 UTC
209.165.201.1/27, version 47031, internal 0x5000001 0x40 (ptr 0x98246ad8) [1], 0x0 (0x0),
0x0 (0x0)
Updated Dec  2 08:27:35.523
Prefix Len 32, traffic index 0, precedence n/a, priority 4
gateway array (0x98099098) reference count 1, flags 0x2010, source rib (7), 0 backups
[1 type 3 flags 0x40441 (0x98134438) ext 0x0 (0x0)]
LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
gateway array update type-time 1 Dec  2 08:27:35.523
LDI Update time Dec  2 08:31:14.951

Level 1 - Load distribution: 0 1
[0] via 209.165.200.225/27, recursive
[1] via 209.165.200.226/27, recursive

via 209.165.200.225/27, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
path-idx 0 NHID 0x0 [0x982355c8 0x0]
next hop 209.165.200.225/27 via 209.165.200.225/27

Load distribution: 0 1 2 2 (refcount 1)

Hash  OK  Interface                 Address
0     Y   HundredGigE0/0/0/0/2     fe80::2
1     Y   Bundle-Ether1201         fe80::2
```

```
2     Y   Bundle-Ether1301          fe80::3
3     Y   Bundle-Ether1301          fe80::3

via 209.165.200.226/27, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
path-idx 1 NHID 0x0 [0x98235678 0x0]
next hop 209.165.200.226/27 via 209.165.200.226/27

Load distribution: 0 1 2 2 (refcount 1)

Hash  OK  Interface                 Address
4     Y   HundredGigE0/0/0/0/2      fe80::2
5     Y   Bundle-Ether1201          fe80::2
6     Y   Bundle-Ether1301          fe80::3
7     Y   Bundle-Ether1301          fe80::3
```

Step 7 Run the **router static** command to configure and map the tags in RIB for IPv6 or IPv4 static routes.

Example:

```
Router#config
Router(config)#router static
Router(config-static)#address-family ipv6 unicast
Router(config-static-afi)#2001:DB8:8::/48 4::4 tag 100
Router(config-static-afi)#commit
```

Step 8 Run the **show route** command to view the configuration for IPv4 or IPv6 static routes.

Example:

The below show output displays the IPv4 static route configuration.

```
Router#show route 209.165.201.30

Tue Dec  3 18:22:39.579 UTC

Routing entry for 209.165.201.30/27
Known via "bgp 100", distance 200, metric 0, lvl2 has dependent
Tag 100, type internal
Installed Dec  3 18:22:35.820 for 00:00:03
Routing Descriptor Blocks
1::2, from 1::2, BGP multi path
Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
Route metric is 0
1::3, from 1::3, BGP multi path
Nexthop in Vrf: "default", Table: "default", IPv6 Unicast, Table Id: 0xe0800000
Route metric is 0
No advertising protos.
```

Example:

The below show output displays the IPv6 static route configuration.

```
Router#show route ipv6 2001:DB8:A:B::1 detail
Tue Dec  3 18:23:55.390 UTC

Routing entry for 2001:DB8:A:B::1/64
Known via "bgp 100", distance 200, metric 0, lvl2 has dependent
Tag 100, type internal
Installed Dec  3 18:22:45.835 for 00:01:09
Routing Descriptor Blocks
1::2, from 1::2
Route metric is 0
Label: None
Tunnel ID: None
Binding Label: None
Extended communities count: 0
```

```
NHID: 0x0 (Ref: 0)
Path Grouping ID: 100
SRv6 Headend: H.Encaps.Red [f3216], SID-list {fccc:bb01:2:e002::}
Route version is 0x1c (28)
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 17265
No advertising protos.
```