Nokia Service Router Linux
7215 Interconnect System
7220 Interconnect Router
7250 Interconnect Router
7730 Service Interconnect Router

Release 26.3

Routing Protocols Guide

3HE 22262 AAAA TQZZA
Edition: 01
March 2026

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia's customers ("You"/"Your") in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia's site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Table of contents

1 About this guide....9

1.1 Precautionary and information messages....9

1.2 Conventions....9

2 What's new....11

3 OSPF....12

3.1 OSPF global configuration....12

3.1.1 Configuring basic OSPF parameters....13

3.1.2 Configuring the router ID....14

3.1.3 Configuring an area....15

3.1.4 Configuring a stub area....15

3.1.5 Configuring a Not-So-Stubby area....16

3.1.6 Configuring an interface....16

4 BGP....18

4.1 BGP best path selection....18

4.2 BGP global configuration....19

4.2.1 Configuring an ASN....19

4.2.2 Configuring the router ID....19

4.3 Configuring a BGP peer group....20

4.4 Configuring BGP neighbors....20

4.5 BGP peer import and export policies....21

4.6 eBGP multihop....25

4.6.1 Configuring eBGP multihop....25

4.7 AS path options....26

4.7.1 Configuring allow-own-as....26

4.7.2 Configuring replace-peer-as....27

4.7.3 Configuring remove-private-as....27

4.8 BGP MED....29

4.8.1 Configuring always-compare-med....29

4.9 BGP AIGP metric....30

4.9.1 Configuring BGP AIGP metric....30

4.10 Route reflection....31

4.10.1 Configuring route reflection....31

4.11 BGP route flap damping....32

4.11.1 Configuring route flap damping parameters for the network instance.... 33

4.11.2 Enabling route flap damping on neighbors or peer groups....33

4.11.3 Displaying route flap damping status and statistics....34

4.11.4 Clearing route flap damping history....36

4.12 BGP add-path....37

4.12.1 Configuring BGP add-path....38

4.12.2 Configuring BGP add-path policy controls....40

4.13 BGP graceful restart....41

4.13.1 Configuring graceful restart....42

4.14 BGP unnumbered peering.... 43

4.14.1 Configuring BGP unnumbered peering....45

4.15 Prefix limits for BGP peers.... 46

4.15.1 Configuring the pre-policy prefix limit for BGP peers.... 47

4.15.2 Configuring the post-policy prefix limit for BGP peers....48

4.15.3 Configuring the prefix limit restart timer for BGP peers....49

4.16 RT constrained route advertisement....50

4.16.1 RT-constrain on SR Linux for various router types.... 50

4.16.2 Best-path selection and RTC route re-advertisement....51

4.16.3 Using RTC routes to filter advertised routes.... 52

4.16.4 BGP RIB YANG model for RTC Routes....53

4.17 BGP configuration management....53

4.17.1 Modifying an ASN.... 54

4.17.2 Deleting a BGP neighbor....54

4.17.3 Deleting a BGP peer group.... 54

4.17.4 Resetting BGP peer connections....54

4.18 BGP shortcuts.... 55

4.18.1 Configuring BGP shortcuts over segment routing.... 55

4.19 BGP TCP MSS....57

4.19.1 Configuring BGP TCP MSS....57

4.20 Error handling for BGP update messages....59

4.21 BGP multipath.... 59

4.21.1 Configuring BGP multipath.... 59

4.22 BGP weighted ECMP using the link-bandwidth EC....60

4.22.1 BGP link bandwidth extended community....61

4.22.2 Configuring BGP wECMP routes using link bandwidth EC.. 62
4.22.3 Aggregating link bandwidth values.. 63
4.22.4 Automatic generation of link-bandwidth EC based on interface speed........................ 64
4.22.4.1 Configuring automatic generation of link-bandwidth EC...................................65
4.23 BGP fast reroute.. 65
4.23.1 Configuring BGP FRR...66
4.24 Configuring BGP send-community type..66
4.25 BGP Link-State...67
4.25.1 Supported NLRI and link-state attributes in BGP-LS..69
4.25.2 Configuring BGP-LS..75
4.25.3 Enabling the import of link-state information into the TE database............................. 76
4.25.4 Excluding a level from being imported into the TE database...................................... 76
4.25.5 Configuring BGP-LS route export to reachable-only.. 77

5 Seamless MPLS with BGP labeled unicast (BGP-LU)..78
5.1 Seamless MPLS with BGP-LU configuration...78
5.2 Initial configuration for Seamless MPLS...79
5.2.1 Configuring interfaces.. 79
5.2.2 Configuring IS-IS...81
5.2.3 Configuring MPLS label blocks..82
5.2.4 Configuring LDP..82
5.3 BGP configuration for Seamless MPLS..83
5.3.1 Configuring BGP on ABRs.. 84
5.3.2 Configuring BGP on the core RR...85
5.3.3 Configuring BGP on ANs toward ABRs...86
5.4 Export policy configuration for Seamless MPLS...87
5.4.1 Configuring export policies on ANs and ABRs..87
5.5 BGP-LU selective install..88
5.5.1 Configuring BGP-LU selective install..89
5.6 Advertisement of BGP-LUv4 routes with IPv6 next hops.. 89
5.6.1 Enabling IPv6 next-hops on BGP-LUv4 routes.. 90

6 Segment routing with BGP-LU prefix SID... 92
6.1 Segment routing with BGP-LU prefix SID topology example.. 93
6.2 Configuring segment routing with BGP-LU prefix SID...95
6.3 LSP statistics for BGP-LU...99

6.3.1 Enabling statistics collection for BGP-LU routes........ 99

6.4 Forwarding entries........ 100

7 IS-IS........ 102

7.1 Basic IS-IS configuration........ 104

7.1.1 Enabling an IS-IS instance........ 104

7.1.2 Configuring the router level........ 105

7.1.3 Configuring the Network Entity Title........ 105

7.1.4 Configuring global parameters........ 106

7.1.5 Configuring interface parameters........ 107

7.1.6 Configuring authentication keys........ 107

7.2 IS-IS graceful restart........ 108

7.2.1 Configuring IS-IS graceful restart........ 109

7.3 Displaying IS-IS information........ 110

7.4 Clearing IS-IS information........ 114

7.5 IS-IS weighted ECMP........ 114

7.5.1 Enabling weighted ECMP........ 115

7.5.2 Configuring weighted load-balancing over interface next-hops........ 116

7.5.3 Normalizing datapath weights........ 117

7.6 Multi-instance IS-IS........ 118

7.6.1 Configuring the IID........ 118

7.6.2 Enabling the IID-TLV........ 118

7.6.3 Displaying MI-ISIS information........ 119

7.7 Multi-Topology IS-IS MT0 and MT2........ 120

7.7.1 Enabling IPv6 unicast routing for an IS-IS instance........ 120

7.7.2 Configuring the default MT2 topology........ 121

7.8 IS-IS extensions for Traffic Engineering (TE)........ 121

7.8.1 Enabling advertisement of IS-IS TE TLVs/sub-TLVs........ 122

7.8.2 TE router ID TLV........ 122

7.8.2.1 Advertising IPv4 TE router ID TLV........ 124

7.8.2.2 Advertising IPv6 TE router ID TLV........ 125

7.8.3 Advertising TE attributes using legacy and ASLA TLVs........ 126

7.8.3.1 Enabling advertisement of TE attributes in legacy mode........ 127

7.9 Non-Stop Routing (NSR) IS-IS........ 128

7.10 Multi-part TLVs in IS-IS........ 128

7.10.1 Enabling MP-TLVs........ 129

8 IP ECMP Load Balancing...130
8.1 Avoiding hash polarization...131
8.2 Configuring IP ECMP load balancing...133
8.3 Resilient ECMP hashing...135
8.3.1 Configuring resilient hashing...137
8.4 Dynamic Load Balancing...137
8.4.1 Configuring dynamic load balancing...138

9 Protocol authentication...142
9.1 Configuring protocol authentication...142

10 Routing policies...145
10.1 Routing policy match conditions...145
10.1.1 Specifying match conditions in a routing policy...148
10.1.2 Routing policy subroutines...149
10.2 Routing policy actions...150
10.2.1 Specifying actions in a routing policy...153
10.2.2 Specifying a default action...155
10.2.3 Policy actions for setting MED in BGP routes...156
10.2.4 Policy action to disable IP FIB installation...158
10.3 Applying a routing policy...159
10.3.1 Applying a default policy to eBGP sessions...160
10.3.2 Replacing an AS path...160
10.4 Resequencing statements in a routing policy...161
10.5 AS path sets...165
10.5.1 Regex modes for AS path sets...165
10.5.2 Configuring an AS path set...169
10.6 BGP community-sets...170
10.6.1 Hybrid community-sets...171
10.6.1.1 Configuring a hybrid community-set...173
10.6.2 Standard community-sets...174
10.6.2.1 Configuring a standard community-set...175
10.6.3 Extended community-sets...176
10.6.3.1 Configuring an extended community-set...180
10.7 Prefix sets...182

10.7.1 Configuring a prefix set....182

11 Network instance route limits....183

11.1 Protocol behavior when route limit is reached....183

11.2 Configuring network instance route limits....184

12 Table connections for route redistribution....186

12.1 Supported table-connections in SR Linux....187

12.2 Enabling table-connections....189

12.3 Configuring table-connections....190

1 About this guide

This document describes routing protocols used with Nokia Service Router Linux (SR Linux). Examples of commonly used commands are provided.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how the router is configured.

Note:
This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Software Release Notes* for information on features supported in each load.

Configuration and command outputs shown in this guide are examples only; actual displays may differ depending on supported functionality and user configuration.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

The Nokia SR Linux documentation uses the following command conventions:

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.

- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

The following table outlines platform grouping conventions used in the SR Linux documentation suite.

Note: Some platforms in the 7250 IXR support mixed systems. For more information about mixed system support, see "Chassis types" in the *Configuration Basics Guide*.

Table 1: Platform grouping legend

Platform group	Description
7215 IXS	7215 IXS-A1
7220 IXR	All 7220 IXR platforms
7220 IXR-D*x*	7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D2L, 7220 IXR-D3, 7220 IXR-D3L, 7220 IXR-D4, 7220 IXR-D5
7220 IXR-H*x*	7220 IXR-H2, 7220 IXR-H3, 7220 IXR-H4, 7220 IXR-H4-32D, 7220 IXR-H5-32D, 7220 IXR-H5-64D, 7220 IXR-H5-64O
7250 IXR[1]	7250 IXR platforms
7250 IXR Gen 2	7250 IXR-6, 7250 IXR-10
7250 IXR Gen 2c+	7250 IXR-6e with IMM2, 7250 IXR-10e with IMM2, 7250 IXR-X1b, 7250 IXR-X3b
7250 IXR Gen 3	7250 IXR-6e with IMM3, 7250 IXR-10e with IMM3, 7250 IXR-18e, 7250 IXR-X4
7250 IXR-6e/10e (mixed system)	7250 IXR-6e (mixed system)[2], 7250 IXR-10e (mixed system)[2]
7730 SXR	7730 SXR-1-32D, 7730 SXR-1d-32D, 7730 SXR-1x-44S

[1] References to the 7250 IXR platform group may be appended with (including mixed systems) or (excluding mixed systems) to indicate mixed system support.

[2] References to this platform as part of 7250 IXR (mixed system) indicate mixed system support of 7250 IXR Gen 2c+ (IMM2) and 7250 IXR Gen 3 (IMM3). That is, the 7250 IXR-6e and 7250 IXR-10e can hold and support both IMM2 and IMM3 at the same time.

2 What's new

This section lists the changes that were made in this release.

Table 2: What's new in Release 26.3.1

Topic	Location
BGP graceful restart notification support (RFC 8538)	BGP graceful restart
BGP fast reroute support for IPv4/IPv6 VPN address families	BGP fast reroute
Per-network instance route limits	Network instance route limits
ECMP hashing enhancements	IP ECMP Load Balancing Configuring IP ECMP load balancing
Non-anchored mode for regular expressions in AS path sets	Regex modes for AS path sets
Automatic generation of link-bandwidth extended community based on interface speed	Automatic generation of link-bandwidth EC based on interface speed
Support for generic extended community sets	Extended community-sets
Support for Group Policy ID extended community sets for micro-segmentation	Extended community-sets
New route property modification action for setting operation to use when applying LOCAL_PREF	Route property modification actions

3 OSPF

Note: OSPF is supported on 7215 IXS, 7220 IXR, 7250 IXR, and 7730 SXR platforms.

Open Shortest Path First (OSPF) is a hierarchical link state protocol. OSPF is an interior gateway protocol (IGP) used within large Autonomous Systems (ASs). OSPF routers exchange state, cost, and other relevant interface information with neighbors. The information exchange enables all participating routers to establish a network topology map. Each router applies the Dijkstra algorithm to calculate the shortest path to each destination in the network. The resulting OSPF forwarding table is submitted to the routing table manager to calculate the routing table.

When a router is started with OSPF configured, OSPF, along with the routing-protocol data structures, is initialized and waits for indications from lower-layer protocols that its interfaces are functional.

The hierarchical design of OSPF allows a collection of networks to be grouped into a logical area. An area's topology is concealed from the rest of the AS which significantly reduces OSPF protocol traffic. With the correct network design and area route aggregation, the size of the route table can be greatly reduced, resulting in decreased OSPF route calculation time and topological database size.

Routers that belong to more than one area are called area border routers (ABRs). An ABR maintains a separate topological database for each area it is connected to. Every router that belongs to the same area has an identical topological database for that area.

Key OSPF areas are:

- Backbone Areas – The backbone distributes routing information between areas.
- Stub areas – A designated area that does not allow external route advertisements. Routers in a stub area do not maintain external routes. A single default route to an ABR replaces all external routes.
- Not-So-Stubby Areas (NSSAs) – NSSAs are similar to stub areas in that no external routes are imported into the area from other OSPF areas. External routes learned by OSPF routers in the NSSA area are advertised as type-7 LSAs within the NSSA area and are translated by ABRs into type-5 external route advertisements for distribution into other areas of the OSPF domain.

3.1 OSPF global configuration

The minimal OSPF parameters that should be configured to deploy OSPF are:

- OSPF version

  SR Linux supports OSPF version 2 and version 3. The OSPF version number must be specified in the configuration. If configuring OSPFv3, you must also specify the address family to be used, either IPv4 or IPv6.

- OSPF instance ID (when configuring multiple instances)

  An OSPF instance ID must be defined when configuring multiple instances or the instance being configured is not the base instance. If an instance ID is not configured, the default instance IDs are as follows:

  – 0 for OSPFv2

  - 0 for OSPF v3 with address family IPv6 unicast
  - 64 for OSPF v3 with address family IPv4 unicast
- Router ID

  Each router running OSPF must be configured with a unique router ID. The router ID is used by both OSPF and BGP routing protocols in the routing table manager.

  When you configure a new router ID, OSPF is automatically disabled and re-enabled to initialize the new router ID.
- An area

  At least one OSPF area must be created. An interface must be assigned to each OSPF area.
- Interfaces

  An interface is the connection between a router and one of its attached networks. An interface has state information associated with it, which is obtained from the underlying lower level protocols and the routing protocol itself. An interface to a network has associated with it a single IP address and mask (unless the network is an unnumbered point-to-point network). An interface is sometimes also referred to as a link.

3.1.1 Configuring basic OSPF parameters

Procedure

To create a basic OSPF configuration, the minimal OSPF parameters required are the following:

- OSPF version number, either v2 or v3. If configuring OSPFv3, you must specify the address family, either IPv4 or IPv6.
- A router ID
- One or more areas
- Interfaces

Example: Configure basic OSPFv2 parameters

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info with-context ospf
    ospf {
        instance default {
            admin-state enable
            version ospf-v2
            router-id 1.1.1.1
            area 0.0.0.1 {
                interface ethernet-1/1.1 {
                    interface-type broadcast
                }
                interface ethernet-1/2.1 {
                    interface-type broadcast
                }
                interface ethernet-1/16.1 {
                    interface-type broadcast
                }
                interface lo0.1 {
                    interface-type broadcast
                }
            }
        }
    }
```

Example: Configure basic OSPFv3 parameters

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info with-context ospf
    ospf {
        instance default {
            admin-state enable
            version ospf-v3
            address-family ipv6-unicast
            router-id 1.1.1.1
            area 0.0.0.1 {
                interface ethernet-1/1.1 {
                    interface-type broadcast
                }
                interface ethernet-1/2.1 {
                    interface-type broadcast
                }
                interface ethernet-1/16.1 {
                    interface-type broadcast
                }
                interface lo0.1 {
                    interface-type broadcast
                }
            }
        }
    }
```

3.1.2 Configuring the router ID

Procedure

You can configure the router ID either at the network-instance level or at the OSPF protocol level. The router ID, expressed like an IP address, uniquely identifies the router within an AS. In OSPF, routing information is exchanged between ASs, groups of networks that share routing information. It can be set to be the same as the loopback (system interface) address. Subscriber services also use this address as far-end router identifiers when service distribution paths (SDPs) are created. The router ID is used by both OSPF and BGP routing protocols.

When you configure a new router ID, OSPF is automatically disabled and re-enabled to initialize the new router ID.

If a router ID is configured at the OSPF protocol level, OSPF uses it instead of the router ID configured at the network-instance level.

Example: Configure router ID at the network-instance level

OSPF uses this router ID unless a different router ID is configured at the OSPF protocol level.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        router-id 10.10.10.104{
        }
    }
```

Example: Configure router ID for the OSPF instance at the OSPF protocol level

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            ospf {
                instance default {
                    version ospf-v2
                    router-id 2.2.2.2
                }
            }
        }
    }
```

3.1.3 Configuring an area

Procedure

An OSPF area consists of routers configured with the same area ID. To include a router in a specific area, the common area ID must be assigned and an interface identified.

If your network consists of multiple areas, you must also configure a backbone area (0.0.0.0) on at least one router. The backbone comprises the area border routers and other routers not included in other areas. The backbone distributes routing information between areas. The backbone is considered to be a participating area within the autonomous system. To maintain backbone connectivity, there must be at least one interface in the backbone area.

The minimal configuration must include an area ID and an interface. Modifying other command parameters are optional.

Example

The following example configures an area at the OSPF level:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                }
            }
        }
    }
```

3.1.4 Configuring a stub area

Procedure

You can configure stub areas to control external advertisement flooding and to minimize the size of the topological databases on an area's routers. A stub area cannot also be configured as an NSSA. By default, summary route advertisements are sent into stub areas.

Example

The following example configures an OSPF stub area:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                    stub {
                    }
                }
            }
        }
    }
```

3.1.5 Configuring a Not-So-Stubby area

Procedure

You can explicitly configure an area to be a Not-So-Stubby Area (NSSA). NSSAs are similar to stub areas in that no external routes are imported into the area from other OSPF areas. An NSSA has the capability to flood external routes it learns throughout its area and by an area border router to the entire OSPF domain. An area cannot be both a stub area and an NSSA.

Example

The following is an OSPF NSSA configuration example:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                    nssa {
                    }
                }
            }
        }
    }
```

3.1.6 Configuring an interface

Procedure

You can configure an interface to act as a connection between a router and one of its attached networks. An interface includes state information that was obtained from underlying lower level protocols and from the routing protocol itself. An interface to a network is associated with a single IP address and mask (unless the network is an unnumbered point-to-point network). If the address is merely changed, then the OSPF configuration is preserved.

Example

The following is an OSPF interface configuration example:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
    protocols {
        ospf {
            instance default {
                version ospf-v2
                area 1.1.1.1 {
                    interface ethernet-1/2 {
                    }
                }
            }
        }
    }
```

4 BGP

Border Gateway Protocol (BGP) is an inter-AS routing protocol. An Autonomous System (AS) is a network or a group of routers logically organized and controlled by common network administration. BGP enables routers to exchange network reachability information, including information about other ASs that traffic must traverse to reach other routers in other ASs.

ASs share routing information, such as routes to each destination and information about the route or AS path, with other ASs using BGP. Routing tables contain lists of known routers, reachable addresses, and associated path cost metrics for each router. BGP uses the information and path attributes to compile a network topology.

To set up BGP routing, participating routers must have BGP enabled, and be assigned to an AS, and the neighbor (peer) relationships must be specified. A router typically belongs to only one AS.

This section describes the minimal configuration necessary to set up BGP in SR Linux. This includes the following:

- Global BGP configuration, including specifying the Autonomous System Number (ASN) of the router, as well as the router ID.
- BGP peer group configuration, which specifies settings that are applied to BGP neighbor routers in the peer group.
- BGP neighbor configuration, which specifies the peer group to which each BGP neighbor belongs, as well as settings specific to the neighbor, including the AS to which the router is peered.

For information about all other BGP settings, see the SR Linux online WebHelp, as well as the *SR Linux Advanced Solutions Guide* and the *SR Linux Data Model Reference Guide*.

4.1 BGP best path selection

To select a single best path for each NLRI, BGP uses the following sequential list of criteria in the order shown:

1. Valid route > invalid route
2. Lowest route preference
3. Highest local preference
4. Shortest AS path
5. Lowest origin
6. Lowest MED
7. EBGP > IBGP
8. Lowest next-hop cost
9. Lowest next-hop type
10. Route owner comparison (**bgp** > **bgp-label** > **bgp-vpn**)
11. Lowest peer router ID

12. Lowest peer IP address
13. Route owner comparison (**bgp-evpn-ifl-host** > **bgp-evpn** > **bgp-ipvpn**)

4.2 BGP global configuration

Global BGP configuration includes specifying the Autonomous System Number (ASN) of the router and the router ID.

4.2.1 Configuring an ASN

Procedure

An ASN is a globally unique value that associates a router to a specific AS. Each router participating in BGP must have an ASN specified.

Example

The following example configures an ASN for a router:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                autonomous-system 65002
            }
        }
    }
```

4.2.2 Configuring the router ID

Procedure

The router ID, expressed like an IP address, uniquely identifies the router and indicates the origin of a packet for routing information exchanged between ASs. The router ID is configured at the BGP level.

Example

The following example configures a router ID:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                router-id 2.2.2.2
        }
    }
```

4.3 Configuring a BGP peer group

Procedure

As part of BGP configuration, you configure a BGP peer group. A BGP peer group is a collection of related BGP neighbors. The group name should be a descriptive name for the group.

All parameters configured for a peer group are inherited by each peer (neighbor) in the peer group, but a group parameter can be overridden for specific neighbors in the configuration of that neighbor.

Example

The following example configures the administrative state and trace options for a BGP peer group. These settings apply to all of the BGP neighbors that are members of this group, unless specifically overridden in the neighbor configuration.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                group headquarters1 {
                    admin-state enable
                    traceoptions {
                        flag events {
                        }
                        flag graceful-restart {
                        }
                    }
                }
            }
```

4.4 Configuring BGP neighbors

Procedure

After configuring a BGP peer group and assigning options, you add neighbors within the same AS to create internal BGP (iBGP) connections and, or neighbors in a different AS to create external BGP (eBGP) peers. All parameters configured for the peer group to which the neighbor is assigned are applied to the neighbor, but a peer group parameter can be overridden on a specific neighbor basis.

Example

The following example configures parameters for two BGP neighbors. The **peer-group** parameter configures both nodes to use the settings specified for the `headquarters1` peer group. The peer group settings apply unless they are specifically overridden in the neighbor configuration.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    peer-group headquarters1
                    description "default network-instance bgp neighbor to Node A"
                    peer-as 65001
                    local-as as-number 65002 {
                    }
                    multihop {
                        admin-state enable
                        maximum-hops 3
                    }
                    failure-detection {
                        enable-bfd true
                        fast-failover true
                    }
                }
                neighbor 192.168.13.2 {
                    peer-group headquarters1
                    description "default network-instance bgp neighbor to Node C"
                    peer-as 65003
                    local-as as-number 65002 {
                    }
                    failure-detection {
                        enable-bfd true
                        fast-failover true
                    }
                }
            }
        }
    }
```

4.5 BGP peer import and export policies

SR Linux supports BGP import policies and export policies:

- An import policy is a sequence of match conditions and action rules that are run on specific routes received from BGP peers. If a received route is rejected by an import policy rule, then depending on the address family and BGP configuration options, the route may be discarded or it may be stored in the RIB but considered invalid and not considered during best-path selection.
- An export policy is a sequence of match conditions and action rules that are run on routes that have been selected for advertisement to a BGP peer. If a route that would normally be advertised to a peer (RIB-OUT) is rejected by an export policy rule, then the actual advertisement of the route to the peer is blocked.

BGP import and export policy rules

You can configure BGP import or export policies at the BGP global, peer-group, and neighbor levels. The policies operate according to the following rules:

For BGP import policies:

- If the configuration of a BGP neighbor explicitly specifies an import policy, then this is the policy used to filter inbound routes from the peer, and import policies defined at higher levels of configuration (group or BGP instance as a whole) are ignored.
- If the configuration of a BGP neighbor does not specify an import policy, but the peer group to which it belongs does specify an import policy, then the peer-group import policy is used to filter inbound routes from the peer, and the import policy defined at the BGP instance level is ignored.

- If the configuration of a BGP neighbor does not specify an import policy, and the peer group to which it belongs also does not specify an import policy, then the BGP-instance import-policy is used to filter inbound routes from the peer.
- If there is no import-policy at any level of configuration that applies to a BGP neighbor, then the handling of received routes depends on the peer session type, as follows:
  - If the peer session type is iBGP, then all received routes are accepted.
  - If the peer session type is eBGP, then all received routes are rejected by default; however, this can be changed by configuring the `ebgp-default-policy import-reject-all` setting to `false`.

For BGP export policies:

- If the configuration of a BGP neighbor explicitly specifies an export policy, then this is the policy used to filter outbound routes sent to the peer, and export policies defined at higher levels of configuration (group or BGP instance as a whole) are ignored.
- If the configuration of a BGP neighbor does not specify an export policy, but the peer group to which it belongs does specify an export policy, then the peer-group export policy is used to filter outbound routes sent to the peer, and the export policy defined at the BGP instance level is ignored.
- If the configuration of a BGP neighbor does not specify an export policy, and the peer group to which it belongs also does not specify an export policy, then the BGP-instance export policy is used to filter outbound routes sent to the peer.
- If there is no export policy at any level of configuration that applies to a BGP neighbor, then the advertisement of routes in the local RIB-IN depends on the RIB-IN type and the peer session type as follows:
  - If the peer session type is iBGP, then all non-imported BGP RIB-INs are accepted and therefore eligible for advertisement.
  - If the peer session type is eBGP, then all routes are rejected by default, meaning that no routes are eligible for advertisement; however, this can be changed by configuring the `ebgp-default-policy export-reject-all` setting to `false`, in which case all non-imported BGP RIB-INs are accepted and eligible for advertisement.

BGP default import policies

For received BGP routes not matching any of the listed peer import policies, the default behavior (without any **default-import-policy** configuration) depends on the context:

- For iBGP peers, the default behavior is `accept`.
- For eBGP peers, the default behavior depends on the configured **import-reject-all** setting.

See the following table for more details. Note that the **default-import-policy** configuration does not have any control over maintenance-mode policy results.

Table 3: Peer import processing

Peer type	Configuration of ebgp-default-policy	Configuration of default-import-policy applied to the peer group and applicable to the AFI-SAFI of the route (and no override at neighbor level)	Configuration of default-import-policy applied to the peer group and applicable to the AFI-SAFI of the route (no configured value means inherit from group)	BGP routes not matched by any of the listed peer import policies
iBGP	N/A	None or `accept`	`accept`	Accepted
		`reject`	`reject`	Rejected
eBGP	`import-reject-all` = true	None or `reject`	`reject`	Rejected
		`accept`	`accept`	Accepted
	`import-reject-all` = false	None or `accept`	`accept`	Accepted
		`reject`	`reject`	Rejected

BGP default export policies

For BGP RIB-INs not matching any of the listed peer export policies, the default behavior (without any `default-export-policy` configuration) depends on the context:

- For iBGP peers, the default is `accept`.
- For eBGP peers, the default depends on the setting for `export-reject-all`.

See the following table for more details. Note that the `default-export-policy` configuration does not have any control over maintenance-mode policy results, and it does not apply to imported, non-BGP routes; to advertise imported routes, they must be matched and accepted by a peer export policy.

Table 4: Peer export processing

Peer type	Configuration of ebgp-default-policy	Configuration of default-export-policy applied to the peer group and applicable to the AFI-SAFI of the route (and no override at neighbor level)	Configuration of default-export-policy applied to the peer group and applicable to the AFI-SAFI of the route (no configured value means inherit from group)	BGP routes not matched by any of the listed peer export policies	Imported non-BGP routes not matched by any of the listed peer export policies
eBGP	`export-reject-all` = true	`accept`	`accept`	Accepted	Rejected
iBGP	N/A	None or `accept`	`accept`	Accepted	Rejected
eBGP	`export-reject-all` = false	None or `accept`	`accept`	Accepted	Rejected
eBGP	`export-reject-all` = true	None or `reject`	`reject`	Rejected	Rejected
iBGP	N/A	`reject`	`reject`	Rejected	Rejected
eBGP	`export-reject-all` = false	`reject`	`reject`	Rejected	Rejected

AFI-SAFI policy attachment rules

The OpenConfig BGP model supports the attachment of AFI-SAFI-specific route policies to selected peers, peer-groups, or to the BGP instance as a whole. This implies, for example, that for one single peer you could have one export policy for IPv4_UNICAST routes advertised to the peer, and a different export policy for IPV6_UNICAST routes advertised to the peer. To accommodate this kind of configuration, SR Linux includes the following contexts supporting attachment of import and export policies:

- `network-instance.protocols.bgp`
- `network-instance.protocols.bgp.afi-safi`
- `network-instance.protocols.bgp.group`
- `network-instance.protocols.bgp.group.afi-safi`
- `network-instance.protocols.bgp.neighbor`
- `network-instance.protocols.bgp.neighbor.afi-safi`

The policy that applies to a route is based on the following order of priority:

1. AFI-SAFI at neighbor level

2. AFI-SAFI at group level
3. AFI-SAFI at instance level
4. General policy at neighbor level
5. General policy at group level
6. General policy at instance level
7. Default policy

4.6 eBGP multihop

External BGP (eBGP) multihop can be used to form adjacencies when eBGP neighbors are not directly connected to each other; for example, when a non-BGP router is between the eBGP neighbors.

BGP TCP/IP packets sent toward an eBGP neighbor by default have a TTL value of 1. If the BGP TCP/IP packets need to pass through more than one router to reach their destination, the TTL decrements to 0, and the packets are dropped.

To prevent this, you can enable multihop for the eBGP neighbor and specify the maximum number of hops for BGP TCP/IP packets sent to the neighbor. This allows the eBGP neighbor to be indirectly connected by up to the specified number of hops.

When multihop is not enabled, the IP TTL for eBGP sessions is set to 1, and the IP TTL for iBGP sessions is set to 64. By enabling multihop and configuring the maximum number of hops to a neighbor, it allows an eBGP session to have multiple hops, and an iBGP session to have a single hop, if required.

If multihop is enabled and the maximum-hops parameter is configured for a BGP peer group, the settings are applied to the members of the group. If the multihop configuration for a neighbor is changed, the session with the neighbor must be disconnected and re-established for the change to take effect.

4.6.1 Configuring eBGP multihop

Procedure

To configure eBGP multihop, you enable it for the eBGP neighbor, and specify a value for the **maximum-hops** parameter. Additionally, the next-hop to the neighbor must be configured so that the two systems can establish a BGP session.

Example: Enable multihop for an eBGP neighbor

The **maximum-hops** parameter is set to 2, which increases the TTL for BGP TCP/IP packets sent toward the eBGP neighbor, allowing the neighbor to be indirectly connected by up to 2 hops.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    multihop {
                        admin-state enable
                        maximum-hops 2
                    }
                }
            }
        }
    }
```

Example: Configure a route to the next-hop toward the eBGP neighbor

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default static-routes
    network-instance default {
        static-routes {
            route 192.168.11.0/24 {
                next-hop-group static-ipv4-grp
            }
        }
    }
--{ * candidate shared default }--[  ]--
# info with-context network-instance default next-hop-groups group static-ipv4-grp
    network-instance default {
        next-hop-groups {
            group static-ipv4-grp {
                nexthop 1 {
                    ip-address 192.168.22.22
                }
            }
        }
    }
```

4.7 AS path options

You can set the following options for handling the AS_PATH in received BGP routes:

- Allow own AS – configures the router to process received routes when its own ASN appears in the AS_PATH.
- Replace peer AS – configures the router to replace the ASN of the peer router in the AS_PATH with its own ASN.
- Remove private AS path numbers – configures the router to either delete private AS numbers, shortening the AS path length, or replace private AS numbers with the local AS number used toward the peer, maintaining the AS path length.

4.7.1 Configuring allow-own-as

Procedure

You can use the **allow-own-as** option to configure the router to process received routes when its own ASN appears in the AS_PATH. Normally, when the ASN of a router appears in the AS_PATH of received routes, it is considered a loop, and the routes are discarded. Specifically, it configures the maximum number of times the global ASN of the router can appear in any received AS_PATH before it is considered a loop and considered invalid. Default is 0.

Example

The following example configures the router to process received routes where its own ASN appears in the AS_PATH a maximum of 1 time:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                autonomous-system 65001
                as-path-options {
                    allow-own-as 1
                }
            }
        }
    }
```

4.7.2 Configuring replace-peer-as

Procedure

You can configure the router to replace the peer ASN in AS_PATH with its own ASN. Normally, two sites having the same ASN would not be able to reach each other directly because the receiving router would see its own ASN in the AS_PATH and consider it a loop. You can overcome this by configuring the router to replace the peer ASN in the AS_PATH with its own ASN. When the **replace-peer-as** option is set to `true`, the router replaces every occurrence of the peer AS number that is present in the advertised AS_PATH with the local ASN used toward the peer.

Example

The following example configures the router to replace the ASN of the peer with its own ASN:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    replace-peer-as true
                    }
                }
            }
        }
    }
```

4.7.3 Configuring remove-private-as

Procedure

You can configure how the router handles private AS numbers: either delete them, shortening the AS path length, or replace private AS numbers with the local ASN used toward the peer, which maintains the AS path length.

You can configure the router to delete or replace private AS numbers that appear before the first occurrence of a non-private ASN in the sequence of most recent ASNs in the AS path. You can also configure the router to ignore private AS numbers when they are the same as the peer ASN.

Example: Configure the router to delete private AS numbers

The following example configures the router to delete private AS numbers (2-byte and 4-byte) from the advertised AS path toward all peers. This shortens the AS path.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode delete
                    }
                }
            }
        }
    }
```

Example

The following example configures the router to replace private AS numbers with the local ASN used toward the peer. This keeps the AS path the same length.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode replace
                    }
                }
            }
        }
    }
```

Example

The following example configures the router to replace only private AS numbers that appear before the first occurrence of a non-private ASN in the sequence of most recent ASNs in the AS path.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode replace
                        leading-only true
                    }
                }
            }
        }
    }
```

Example

The following example configures the router to ignore private AS numbers (not replace them) when they are the same as the peer AS number.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                as-path-options {
                    remove-private-as {
                        mode replace
                        ignore-peer-as true
                    }
                }
            }
        }
    }
```

4.8 BGP MED

The Multi-Exit Discriminator (MED) attribute is an optional attribute that can be added to routes advertised to an eBGP peer to influence the flow of inbound traffic to the AS. The MED attribute carries a 32-bit metric value. A lower metric is better than a higher metric when MED is compared by the BGP decision process.

By default, the MED attribute is compared only if the routes come from the same neighbor AS. You can optionally configure SR Linux to compare the MED value from different ASs when selecting the best route.

4.8.1 Configuring always-compare-med

Procedure

To configure SR Linux to use MED values from different ASs in the BGP decision process (tie-break between routes for the same NLRI), set the **always-compare-med** option to `true`. By default, this option is set to `false`, which uses MED values in the BGP decision process only for routes from the same neighbor AS.

Example

The following example sets the **always-compare-med** option to `true`:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp best-path-selection
    network-instance default {
        protocols {
            bgp {
                best-path-selection {
                    always-compare-med true
                }
            }
        }
    }
```

4.9 BGP AIGP metric

Note: The BGP AIGP metric is supported only on 7250 IXR platforms.

The accumulated IGP (AIGP) metric is an optional non-transitive attribute that can be attached to selected routes to influence the BGP decision process to prefer BGP paths with a lower end-to-end IGP cost, even when the compared paths span more than one AS or IGP instance. AIGP is different from MED in several important ways:

- AIGP is not intended to be transitive between completely distinct autonomous systems (only across internal AS boundaries).
- AIGP is always compared in paths that have the attribute, whether they come from a different neighbor AS or not.
- AIGP is more important than MED in the BGP decision process.
- AIGP is automatically incremented every time there is a BGP next-hop change so that it can track the end-to-end IGP cost, whereas all arithmetic operations on MED attributes must be done manually (for example, using route policies).

In the SR Linux implementation, AIGP is supported only in the base router BGP instance and only for the following types of routes: IPv4 unicast, labeled IPv4 unicast, IPv6 unicast, and labeled IPv6 unicast. When AIGP is enabled for an address family, the AIGP attribute is sent to all peers, except for peers or groups configured with the **block-accumulated-igp** command. For the purpose of best path selection, all routes from a blocked peer are treated as though they were received without any AIGP attribute. If the AIGP attribute is received from a peer that is not configured for AIGP, or if the attribute is received in a non-supported route type, the attribute is discarded and not propagated to other peers (but it is still displayed in BGP **info from state** commands).

When a router receives a route with an AIGP attribute and it re-advertises the route to an AIGP-enabled peer without any change to the BGP next hop, the AIGP metric value is unchanged by the advertisement (RIB-OUT) process. But if the route is re-advertised with a new BGP next hop, the AIGP metric value is automatically incremented by the route table (or tunnel table) cost to reach the received BGP next hop.

Note: No route policy configuration is required to advertise the AIGP attribute to peers. When AIGP is enabled for an address family, the router advertises the AIGP attribute to all peers in that family (unless they are explicitly blocked).

4.9.1 Configuring BGP AIGP metric

Procedure

To enable BGP AIGP for all routes of an address family, use the **protocols bgp afi-safi best-path-selection accumulated-igp** command.

Example: Configure BGP AIGP metric

In the following example, AIGP is enabled for IPv4 unicast and labeled unicast address families. AIGP is also supported for IPv6 unicast and labeled unicast address families.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp
```

```
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-labeled-unicast {
                    best-path-selection {
                        accumulated-igp true
                    }
                }
                afi-safi ipv4-unicast {
                    best-path-selection {
                        accumulated-igp true
                    }
                }
            }
        }
    }
```

4.10 Route reflection

In a standard iBGP configuration, all BGP speakers within an AS must have full BGP mesh to ensure that all externally learned routes are redistributed through the entire AS.

Configuring route reflection provides an alternative to the full BGP mesh requirement: instead of peering with all other iBGP routers in the network, each iBGP router only peers with a router configured as a route reflector.

An AS can be divided into multiple clusters, with each cluster containing at least one route reflector, which redistributes routes to the clients in the cluster. The clients within the cluster do not need to maintain a full peering mesh between each other. They only require a peering to the route reflectors in their cluster. The route reflectors must maintain a full peering mesh between all non-clients within the AS.

4.10.1 Configuring route reflection

Procedure

To configure a route reflector, you assign it a cluster ID and specify which neighbors are clients and which are non-clients. Clients receive reflected routes, and non-clients are treated as a standard iBGP peer.

Example

The following example configures the router to be a route reflector for two clients SRL-1 and SRL-2. The router is assigned cluster ID 0.0.0.1.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                route-reflector {
                    cluster-id 0.0.0.1
                    }
                }
                neighbor SRL-1 {
                    route-reflector {
                        cluster-id 0.0.0.1
                        client true
                    }
                neighbor SRL-2 {
                    route-reflector {
                        cluster-id 0.0.0.1
                        client true
                    }
                }
            }
        }
    }
```

4.11 BGP route flap damping

Note: BGP route flap damping is supported on 7730 SXR, 7250 IXR, and 7220 IXR systems.

Route flap damping (RFD) is a mechanism that is designed to improve the stability of Internet routing by mitigating the impact of route flaps. Route flaps occur when a router alternately advertises a route as reachable and then unreachable, or as reachable through one path and then through another path, in rapid succession. Route flaps can result from hardware errors, software errors, configuration errors, unreliable links, and so on. However, not all perceived route flaps represent a true problem; when a best path is withdrawn, the next-best path may not be immediately known and can trigger a number of intermediate best path selections (and corresponding advertisements) before it is found. These intermediate best path selections can travel at different speeds through different routers because of the effect of the minimum route advertisement interval (MRAI) and other factors.

In SR Linux, RFD is configurable but is disabled by default. It can be enabled on eBGP sessions by including the **route-flap-damping** command in a group or neighbor configuration. The **route-flap-damping** command has no effect on iBGP sessions. When a route for an NLRI of any AFI/SAFI type is received on an eBGP session that has **route-flap-damping** enabled, the feature behavior is as follows:

- If the route changes from reachable to unreachable because of a withdrawal by the RFD peer, damping history is created for the route in the RIB-IN (if it does not already exist). In the damping history, the figure of merit (FOM), an accumulated penalty value, is incremented by 1024. This penalty amount is non-configurable.
- If a reachable route is updated by the RFD peer with new path attribute values, then the FOM is incremented by 1024. (If the update did not change any path attributes, the FOM remains unchanged.)
- In SR Linux, the FOM has a hard, non-configurable upper limit of 21540.
- The FOM value is decayed exponentially as described in RFC 2439. The **half-life** of the decay is 15 minutes by default, however the **half-life** is configurable per instance to any value between 1 and 45 minutes.
- The FOM value at the last time of update can be displayed in state using the **route-flap-damping figure-of-merit** commands under the **info from state network-instance bgp-rib afi-safi** context. The time of last update can be within the preceding 640 seconds; SR Linux does not calculate the current FOM every time the **info from state** command is entered.
- When the FOM reaches the **suppress-threshold**, the route is suppressed, which means that the route is not used locally and not advertised to peers. The **suppress-threshold** is 3000 by default, but can be changed per instance to any value between 1 and 20 000. The route remains suppressed until either the FOM exponentially decays to a value less than or equal to the **reuse-threshold** or the **max-suppress-time** is reached. By default, the **reuse-threshold** is 750 and the **max-suppress-time** is 60

minutes, but these values are configurable per instance: **reuse-threshold** can have a value between 1 and 20 000 and **max-suppress-time** can have a value between 1 and 720 minutes.

4.11.1 Configuring route flap damping parameters for the network instance

About this task

Route flap damping (RFD) applies only to routes received from eBGP peers. It has no effect on routes from iBGP peers. An eBGP peer with RFD enabled is described as an RFD peer. RFD applies to all AFI/SAFI families that are negotiated with the RFD peer.

Procedure

To configure BGP route flap damping parameters for a network instance (default or IP-VRF), use the **network-instance protocols bgp route-flap-damping** command.

Example: Configure global route flap damping parameters

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp route-flap-damping
    network-instance default {
        protocols {
            bgp {
                route-flap-damping {
                    half-life 20
                    max-suppress-time 70
                    reuse-threshold 800
                    suppress-threshold 3500
                }
            }
        }
    }
```

4.11.2 Enabling route flap damping on neighbors or peer groups

Procedure

To enable route flap damping for neighbors or peer groups, use the **route-flap-damping true** command under the BGP **neighbor** or **group** context.

Example: Enable route flap damping on a neighbor

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 192.0.2.1
    network-instance default {
        protocols {
            bgp {
                neighbor 192.0.2.1 {
                    route-flap-damping true
                }
            }
        }
    }
```

Example: Enable route flap damping on a peer group

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp group custom-peer-group
    network-instance default {
        protocols {
            bgp {
                group custom-peer-group {
                    route-flap-damping true
                }
            }
        }
    }
```

4.11.3 Displaying route flap damping status and statistics

Procedure

To display the aggretate route flap damping statistics, use the following **info from state** commands:

- **info from state network-instance protocols bgp statistics total-decayed-routes** – displays the total number of received BGP routes (for all AFI-SAFI families) that are eligible for use but have a FOM greater than 0 and less than the suppress threshold
- **info from state network-instance protocols bgp statistics total-history-routes** – displays the total number of recently withdrawn BGP routes (for all AFI-SAFI families) that are still held in the BGP RIB because their FOM is greater than 0
- **info from state network-instance protocols bgp statistics total-suppressed-routes** – displays the total number of received BGP routes (for all AFI-SAFI families) that are suppressed because their FOM is greater than the suppress threshold
- **info from state network-instance protocols bgp neighbor afi-safi suppressed-routes** – displays the number of routes of each AFI/SAFI family received from the peer that are suppressed because their FOM is greater than the suppress threshold

To display the route flap damping status and statistics for individual routes, use the following **info from state** commands under the **info from state network-instance bgp-rib afi-safi local-rib route** or **info from state network-instance bgp-rib afi-safi rib-in-out rib-in-post route** contexts:

- **route-flap-damping suppressed** – reads true when a non-withdrawn route is suppressed because the FOM is greater than the suppress threshold
- **route-flap-damping history** – reads true when the current FOM for a recently withdrawn route is greater than 0
- **route-flap-damping decayed** – reads true when the current FOM for a non-withdrawn route is greater than 0 but less than the suppress threshold
- **route-flap-damping figure-of-merit** – displays the FOM value, which is updated regularly when there is RFD history (maximum delay between updates is 640 seconds).
- **route-flap-damping reuse-time** – displays the amount of time remaining (in seconds) before a suppressed route can be used again. Displays 0 if the route is not currently suppressed.
- **route-flap-damping flap-count** – displays the number of times that the route flapped. The flap count increments when a route is withdrawn by an RFD peer or when a route is updated by an RFD peer and the update has changed one or more path attributes.

Example: Display aggregate route flap damping statistics

```
--{ candidate shared default }--[  ]--
# info with-context from state network-instance default protocols bgp statistics
    network-instance default {
        protocols {
            bgp {
                statistics {
                    total-paths 4
                    total-prefixes 8
                    path-memory 13960
                    total-received-routes 8
                    total-active-routes 8
                    total-decayed-routes 4
                    total-history-routes 0
                    total-suppressed-routes 0
                    total-peers 4
                    up-peers 4
                    disabled-peers 0
                }
            }
        }
    }
```

Example: Display route flap damping statistics for a local BGP RIB route

```
# info with-context from state network-instance default bgp-rib afi-safi ipv4-unicast
 ipv4-unicast local-rib route 172.16.0.4/32 neighbor 192.168.0.666 origin-protocol bgp
 path-id 0
    network-instance default {
        bgp-rib {
            afi-safi ipv4-unicast {
                ipv4-unicast {
                    local-rib {
                        route 172.16.0.4/32 neighbor 192.168.0.66 origin-protocol bgp
 path-id 0 {
                            last-modified "2025-03-05T06:08:00.200Z (a minute ago)"
                            used-route true
                            valid-route true
                            best-route true
                            backup-route false
                            stale-route false
                            fib-disabled false
                            pending-delete false
                            neighbor-as 4
                            group-best true
                            tie-break-reason none
                            attr-id 3830
                            route-flap-damping {
                                suppressed false
                                history false
                                decayed true
                                figure-of-merit 1024
                                reuse-time 0
                                flap-count 1
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Display route flap damping statistics for a post-import-policy BGP route

```
# info with-context from state network-instance default bgp-rib afi-safi ipv4-unicast
 ipv4-unicast rib-in-out rib-in-post route 172.16.0.4/32 neighbor 192.168.0.66 path-id 0
    network-instance default {
        bgp-rib {
            afi-safi ipv4-unicast {
                ipv4-unicast {
                    rib-in-out {
                        rib-in-post {
                            route 172.16.0.4/32 neighbor 192.168.0.66 path-id 0 {
                                last-modified "2025-03-05T06:08:00.200Z (3 minutes ago)"
                                used-route true
                                valid-route true
                                best-route true
                                backup-route false
                                stale-route false
                                fib-disabled false
                                pending-delete false
                                neighbor-as 4
                                group-best true
                                tie-break-reason none
                                attr-id 3830
                                route-flap-damping {
                                    suppressed false
                                    history false
                                    decayed true
                                    figure-of-merit 1024
                                    reuse-time 0
                                    flap-count 1
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

4.11.4 Clearing route flap damping history

Procedure

To clear the route flap damping history, use the **route-flap-damping clear-history** tools command in the **bgp** context, or in the **bgp group** or **bgp neighbor** context for a more granular operation.

Example: Clear route flap damping history

```
--{ candidate shared default }--[  ]--
# tools network-instance default protocols bgp route-flap-damping clear-history
```

Example: Clear route flap damping history (specified group)

```
--{ * candidate shared default }--[  ]--
# tools network-instance default protocols bgp group custom-bgp-group route-flap-damping
 clear-history
```

Example: Clear route flap damping history (specified neighbor)

```
--{ * candidate shared default }--[  ]--
# tools network-instance default protocols bgp neighbor 192.0.2.1 route-flap-damping
 clear-history
```

4.12 BGP add-path

Note: BGP add-path is supported on 7730 SXR, 7250 IXR, and 7220 IXR systems.

Normally if the SR Linux device receives an advertisement of an NLRI and path from a specific peer, and that peer subsequently advertises the same NLRI with different path information (a different next hop or different path attributes), the new path effectively overwrites the existing path.

If BGP add-path has been negotiated with the peer, there is a different behavior: the newly advertised path is stored in the RIB-IN along with all of the paths previously advertised (and not withdrawn) by the peer.

For router A to receive multiple paths per NLRI from peer B for a particular address family, the BGP capabilities advertisement during session setup must indicate that peer B needs to send multiple paths for the address family, and that router A is willing to receive multiple paths for the address family.

When the add-path receive capability for an address family has been negotiated with a peer, all advertisements and withdrawals of NLRI within that address family by that peer includes a path identifier.

- If the combination of NLRI and path identifier in an advertisement from a peer is unique (does not match an existing route in the RIB-IN from that peer), the route is added to the RIB-IN.
- If the combination of NLRI and path identifier in a received advertisement is the same as an existing route in the RIB-IN from the peer, the new route replaces the existing one.
- If the combination of NLRI and path identifier in a received withdrawal matches an existing route in the RIB-IN from the peer, that route is removed from the RIB-IN.

BGP add-path is supported by BGP running in the default network-instance and BGP running in any IP-VRF network-instance.

BGP add-path is configurable per address family at the network-instance, group, and neighbor levels. Inheritance of add-path configuration from network-instance to group to neighbor is per address family. The following address families are supported:

- IPv4 unicast
- IPv6 unicast
- Layer 3 VPN IPv4 unicast
- Layer 3 VPN IPv6 unicast
- IPv4 labeled unicast
- IPv6 labeled unicast
- EVPN
- Route target

4.12.1 Configuring BGP add-path

Procedure

SR Linux supports the following add-path options:

- **receive** – Negotiate with a peer to receive multiple path advertisements from a single peer for a single NLRI belonging to the address family.
- **send** – Negotiate with a peer to send multiple path advertisements to a single peer for a single NLRI belonging to the address family.
- **send-max** – Send the best paths for a single NLRI, up to a configured maximum, or as many as possible until there are no more valid paths to send.
- **send-multipath** – Send the used paths for a single NLRI, including all paths that are multipaths.
- **eligible-prefix-policy** – Control add-path send behavior using a routing policy. This option is not supported at the group or neighbor levels.

Example: Enable BGP add-path send for an address family

The following example enables the SR Linux device to negotiate with a BGP peer to send multiple path advertisements for a single NLRI belonging to an address family.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 1.1.1.1
    network-instance default {
        protocols {
            bgp {
                neighbor 1.1.1.1 {
                    afi-safi ipv4-unicast {
                        add-paths {
                            send true
                        }
                    }
                }
            }
        }
    }
```

Example: Send up to a maximum number of paths

The following example enables the SR Linux device to send up to 10 advertisements for a single NLRI belonging to the IPv4 unicast address family.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    add-paths {
                        send-max 10
                    }
                }
            }
        }
    }
```

Example: Use a routing policy to control BGP add-path behavior

The following example configures a routing policy that matches prefixes in a prefix-set with a policy-result action of accept. The routing policy is specified in the add-paths configuration to control the BGP add-path send behavior for matching prefixes.

```
--{ * candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        prefix-set pset1 {
            prefix 10.3.192.0/21 mask-length-range 21..24 {
            }
            prefix 10.3.191.0/21 mask-length-range exact {
            }
        }
        policy ap1 {
            statement st1 {
                match {
                    prefix-set pset1
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    add-paths {
                        eligible-prefix-policy ap1
                    }
                }
            }
        }
    }
```

The routing policy referenced by **eligible-prefix-policy** can have the following match conditions:

- prefix-set
- family
- community-set

The action in the routing policy can be accept, reject, or next-statement. Route property modification actions in the routing policy are ignored.

Note:

- If no routing policy is configured to control add-path send behavior, it is advertised for all prefixes for the specified address family.
- If a routing policy is configured, but there is no match, add-path capability is advertised for the prefix according to the **afi-safi** configuration.
- If the routing policy is matched, and the action is accept, add-path capability is advertised for the prefix according to the **afi-safi** configuration.

- If the routing policy is matched, and the action is reject, add-path capability is not advertised for the prefix.
- If a routing policy to control add-path send behavior is configured with an explicit **default-action policy-result accept** entry, but there is no match, add-paths are advertised for the prefix according to the **afi-safi** configuration.
- If a routing policy to control add-path send behavior is configured with an explicit **default-action policy-result reject** entry, but there is no match, add-paths are not advertised for the prefix according to the **afi-safi** configuration.
- If a routing policy to control add-path send behavior is configured with no **default-action** entry, but there is no match, no add-paths are advertised for the prefix according to the **afi-safi** configuration.

4.12.2 Configuring BGP add-path policy controls

Procedure

To control the prefixes that are eligible to be used as add-paths, you can specify a policy to accept or reject specific prefixes within the add-path using the **routing-policy** command. The specified policy can match on **prefix**, **family**, **standard-community**, and **extended-community**. You can then reference the prefix policy using the **afi-safi add-paths eligibile-prefix-policy** command under the **bgp**, **bgp group**, or **bgp neighbor** contexts.

Example: Configure a routing policy for BGP eligible prefix

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy policy custom-add-path-policy
    routing-policy {
        policy custom-add-path-policy {
            statement 100 {
                match {
                    family [
                        ipv4-unicast
                    ]
                    prefix {
                        prefix-set custom-prefix-set
                    }
                    bgp {
                        standard-community {
                            standard-community-set custom-stndrd-community-set
                            match-set-options any
                        }
                        extended-community {
                            extended-community-set custom-ext-community-set
                            match-set-options all
                        }
                    }
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

Example: Reference the BGP eligible prefix policy

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast add-paths
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    add-paths {
                        eligible-prefix-policy custom-add-path-policy
                    }
                }
            }
        }
    }
```

4.13 BGP graceful restart

BGP graceful restart allows a router whose control plane has temporarily stopped functioning because of a system failure or a software upgrade to return to service with minimal disruption to the network.

To do this, the router relies on neighbor routers, which have also been configured for graceful restart, to maintain forwarding state while the router restarts. These neighbor routers are known as helper routers. The helper routers and the restarting router continue forwarding traffic using the previously learned routing information from the restarting router. Other routers in the network are not notified about the restarting router, so network traffic is not disrupted.

When graceful restart is enabled on the SR Linux router and its neighbor, the two routers exchange information about graceful restart capability, including the Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI) of the routes supported for graceful restart.

While the router restarts, the helper router marks the routes from the restarting router as stale, but continues to use them for traffic forwarding. When the BGP session is reestablished, the restarting router indicates to the helper router that it has restarted. The helper router then sends the restarting router any BGP RIB updates, followed by an End-of-RIB (EOR) marker indicating that the updates are complete. The restarting router then makes its own updates and sends them to the helper router, followed by an EOR marker.

Graceful restart is used in conjunction with the In-Service Software Upgrade (ISSU) feature, which can be used to upgrade 7220 IXR-D2 and 7220 IXR-D3 systems while maintaining non-stop forwarding. During the ISSU, a warm reboot brings down the control and management planes while the NOS reboots, and graceful restart maintains the forwarding state in peers. You can use a **tools** command to validate that the SR Linux and its peers support warm reboot, including graceful restart configuration. See the *SR Linux Software Installation Guide* for more information.

BGP graceful restart notification support per RFC 8538

The SR Linux graceful restart implementation supports RFC 8538 operation by default. This setting is non-configurable.

Without RFC 8538 support, when the peer receives any notification message, a BGP session restart is initiated. With RFC 8538 support, the stale routes can remain in use even after receiving a notification message.

Notification message

When a non-recoverable error related to a BGP session occurs, a notification message is sent to the peer and the session is terminated unless graceful restart with RFC 8538 support is enabled. The notification message provides the following information:

- **error code**

  Indicates the type of error: message header error, open message error, update message error, hold timer expired, finite state machine error, or cease.

- **error subcode**

  Provides more specific information about the error.

Events triggering full restart of the BGP session

With graceful restart enabled, the following events trigger a full restart of the BGP session:

- Receiving a cease notification with any of the following subcodes:
  - subcode 1 (maximum number of prefixes reached)
  - subcode 2 (administrative shutdown)
  - subcode 3 (peer deconfigured)
  - subcode 9 (hard reset)
- Manual stop event (**admin-state disable** of the **bgp neighbor**)
- Automatic stop event (prefix limit exceeded with teardown action enabled)

Events triggering graceful restart of the BGP session

With graceful restart enabled, the following events trigger a graceful restart of the BGP session:

- Receiving any notification other than those listed in the full restart triggers above (assuming N=1 is included in the received capability)
- Hold-timer expiry (**bgp [group | neighbor} timers hold-time**)
- TCP connection failure

4.13.1 Configuring graceful restart

Procedure

You can configure graceful restart for the BGP instance. The SR Linux device operates as a helper router for neighbor routers when they are restarting, assuming graceful restart is also enabled on the neighbors. Enabling graceful restart also indicates to the neighbors that they can serve as helper routers when the SR Linux device itself is restarting.

Example

When operating as a helper router, the SR Linux device marks the routes from the restarting router as stale, but continues to use them for forwarding for a period of time while the neighbor router restarts. After this period expires, the SR Linux device deletes the routes. The **stale-routes-time** parameter configures the amount of time in seconds the routes remain stale before they are deleted.

The **requested-restart-time** parameter configures the amount of time in seconds to wait for a graceful restart-capable neighbor to re-establish a TCP connection. After this period expires, the helper router deletes the stale routes it preserved on behalf of its neighbor routers.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                graceful-restart {
                    admin-state enable
                    stale-routes-time 300
                    requested-restart-time 300
                }
            }
        }
    }
```

Following a restart, by default the system waits 600 seconds (10 minutes) to receive EOR markers from all helper routers for all address families that were up before the restart. After this time elapses, the system assumes convergence has occurred and sends its own EOR markers to its peers. You can configure the amount of time the system waits to receive EOR markers to be from 0 to 3,600 seconds.

For example, the following configures the amount of time the system waits to receive EOR markers to 270 seconds.

```
--{ * candidate shared default }--[  ]--
# info with-context system warm-reboot
    system {
        warm-reboot {
            bgp-max-wait 270
        }
    }
```

4.14 BGP unnumbered peering

In a typical large-scale data center using BGP, leaf and spine switches are interconnected in a Clos topology, and each device establishes a single-hop eBGP session with each of its physically connected peers. The sessions come up as eBGP because of the ASN allocation scheme; it is common practice to assign a unique ASN to every leaf switch (TOR) in a cluster and a different unique ASN to the set of spine switches to which those TORs are connected. The allocated ASNs are typically private ASNs in the range 4200000000 to 4294967294, although this is not always the case.

For this type of configuration, BGP unnumbered peering can be a useful solution. BGP unnumbered peering is the dynamic setup of one or more single-hop BGP sessions over a network segment that has no globally-unique IPv4 or IPv6 addresses. Each router connected to the network segment is assumed to have an IPv6-enabled interface to the network, and these interfaces have IPv6 link-local addresses that are typically auto-generated by each router from the interface MAC addresses.

How sessions are established using BGP unnumbered peering

The set of BGP speakers configured for BGP unnumbered peering on a network segment discover each other by sending and receiving ICMPv6 router advertisement (RA) messages.

Consider an example of Router A and Router B, which are both connected to an unnumbered interface and configured for BGP unnumbered dynamic session setup. The BGP session between the two routers is established in the following sequence:

1. Router B sends an ICMPv6 RA message on its interface b1.
   Assuming the RA message is unsolicited, the source IP address of this message is the link-local address of interface b1 (fe80::7efe:90ff:fefc:7ad8), and the destination IP address is the all-nodes multicast address.
2. Asynchronously, Router A sends an ICMPv6 RA message on its interface a1.
   The source IP address is the link-local address of interface a1 (fe80::7efe:90ff:fefc:7bd8), and the destination IP address is the all-nodes multicast address.
3. Router A receives the RA message on interface a1, and the software process responsible for ICMPv6 relays the information to BGP, because in the BGP configuration, a1 is a subinterface that is configured as a dynamic neighbor interface; that is, added to the BGP dynamic-neighbors interface list.
4. BGP checks if it already has a BGP session with fe80::7efe:90ff:fefc:7ad8.
   - If BGP already has this session and it is up, or BGP is in the process of establishing this session, then do nothing. Possibly, Router B started the same process moments before Router A.
   - If BGP does not have a session with this link-local address, then a new TCP connection is initiated toward fe80::7efe:90ff:fefc:7ad8.
5. When the TCP connection is established, the BGP OPEN message sent by Router A encodes a local-AS and other capabilities that come from the configuration of the peer-group associated with interface a1.
6. Router A receives a BGP OPEN message from Router B and accepts that OPEN message, proceeding to move toward the BGP established state, if the OPEN message encodes an acceptable peer AS number (in one of the `allowed-peer-as` ranges configured for interface a1). The address families supported by the session are based on the usual MP-BGP negotiation.

BGP dynamic-neighbors interface list

To enable dynamic peering, you add subinterfaces to the BGP dynamic-neighbors interface list in the SR Linux configuration.

When a subinterface is added to the dynamic-neighbors interface list:

- BGP automatically accepts incoming BGP connections to the IPv6 link-local address of that subinterface, subject to the configured `max-sessions` limit for the subinterface.
  For the connection to be accepted, the source address must be an IPv6 link-local address (that may or may not also be a defined neighbor address), and the reported ASN of the peer must match relevant configuration. If the source address does not match a configured neighbor address, the session is set up according to the peer-group associated with the subinterface, not the peer-group associated with the **dynamic-neighbors accept match-***prefix* entry matching the source IPv6 link-local address if a matching entry exists.
- BGP registers for IPv6 RA messages on the subinterface. Whenever the source of one of these RA messages matches an IPv6 link-local address for which there is currently no established BGP session, the system attempts to create a BGP session to that address, as long as this does not exceed the configured `max-sessions` limit for the subinterface. The session is set up according to the configured `peer-group` associated with the subinterface.

When a BGP session is established over a subinterface in the dynamic-neighbors interface list:

- Changes to the `allowed-peer-as` ranges associated with the subinterface only take effect from the next time BGP attempts to establish the sessions.
- Non-arrival of expected ICMPv6 RA messages on the subinterface do not trigger teardown of associated sessions.
- Existing triggers for tearing down a session apply as normal (for example, hold-timer expiration, BFD timeout, **clear bgp neighbor** commands, and so on).
- If the link-local address of a dynamic peer is configured as a static neighbor address, the dynamic session is immediately torn down and replaced by the static session.

When a subinterface is deleted from the dynamic-neighbors interface list, all dynamic sessions associated with that subinterface (excluding sessions set up by static configuration of the neighbor) are torn down immediately.

A BGP session that was previously established on an unnumbered interface and subsequently torn down can only be re-established if the subinterface is configured in the dynamic-neighbors interface list and a recent ICMPv6 RA message is received.

Configuration overrides for dynamic peers on unnumbered interfaces

When a dynamic BGP session is initiated or accepted on an interface that is tied to a peer-group, most of the parameters relevant to that session come from the configuration of that peer-group, with the following exceptions:

- `multihop maximum-hops` is always 1 (for both eBGP and iBGP peers).
- `transport local-address` is always the link-local address of the specified interface.
- `next-hop-self` is always `true`. The neighbor is not presumed to have reachability to off-link destinations.
- `transport passive-mode` is always `false`. BGP always initiates a connection when informed by ICMPv6, unless it already has a connection.
- `afi-safi ipv4-unicast ipv4-unicast receive-ipv6-next-hops` is always `true`.
- `afi-safi ipv4-unicast ipv4-unicast advertise-ipv6-next-hops` and `evpn advertise-ipv6-next-hops` are always `true`.

Peer AS Validation for dynamic peers on unnumbered interfaces

When a BGP OPEN message is received from an unnumbered dynamic neighbor, the reported AS number of the peer is checked to determine if it is acceptable to allow the peering to proceed.

For a dynamic session associated with a subinterface, the peer AS is acceptable only if it matches one of the `allowed-peer-as` elements of the dynamic-neighbors interface list entry for the subinterface, or if the peer AS is equal to the local AS (implying an iBGP session).

4.14.1 Configuring BGP unnumbered peering

Procedure

To configure BGP unnumbered peering, you add subinterfaces to the BGP dynamic-neighbors interface list, and specify the peer autonomous system numbers from which incoming TCP connections to the BGP well-known port are accepted.

Example

The following example adds a subinterface to the BGP dynamic-neighbors interface list.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp dynamic-neighbors interface
 ethernet-1/1.1
    network-instance default {
        protocols {
            bgp {
                dynamic-neighbors {
                    interface ethernet-1/1.1 {
                        peer-group bgp_peer_group_0
                        allowed-peer-as [
                            4294967200
                        ]
                    }
                }
            }
        }
    }
```

In this example, subinterface `ethernet-1/1.1` is added to the BGP dynamic-neighbors interface list. This subinterface must be enabled for IPv6 and configured to accept and send IPv6 RA messages. It does not require any IPv4 addresses or global-unicast IPv6 addresses.

Incoming TCP connections to port 179 received on this subinterface that are sourced from an IPv6 link-local address and destined for the IPv6 link local address of this subinterface are automatically accepted. IPv6 RA messages received on this subinterface automatically trigger BGP session setup toward the sender of these messages, if there is not already an established BGP session.

Peer group `bgp_peer_group_0` is associated with dynamic BGP neighbors on this subinterface. Parameters configured for this peer-group are used for establishing the dynamic BGP session, with the exceptions described in Configuration overrides for dynamic peers on unnumbered interfaces.

ASN 4294967200 is configured as an allowed peer AS for dynamic BGP neighbors on this subinterface. If the BGP OPEN message from a peer on this subinterface contains a MyAS number that is not an allowed peer AS, then a NOTIFICATION is sent to the peer with the indication Bad Peer AS.

4.15 Prefix limits for BGP peers

Note: Prefix limits for BGP peers are supported on 7730 SXR, 7250 IXR, 7220 IXR, and 7215 IXS systems.

SR Linux supports placing limits on the number of IPv4, IPv6, or EVPN route prefixes that can be received from a peer or from individual members of a peer group. When this prefix limit is exceeded, SR Linux tears down the BGP session with the peer and reestablishes the session.

SR Linux supports two different types of prefix limits:

- **pre-policy prefix limit**
  The pre-policy prefix limit applies the prefix limit before import policies are applied to the routes. This limit counts all BGP routes received from the peer including those that the import policies subsequently reject.
- **post-policy prefix limit**

The post-policy prefix limit applies the prefix limit only after import policies are applied to the routes. This limit only counts BGP routes received from the peer that are accepted after all import policies are applied.

For a particular AFI-SAFI family and peer (or peer group), you can configure either a pre- or post-policy prefix limit, but not both.

Prefix limit settings

The following prefix limit settings are configurable per AFI-SAFI family and per peer (or peer group):

- **max-received-routes** – sets the maximum number of routes that can be received from the peer before a BGP session teardown is triggered. The default value is 4294967295.
- **prefix-limit-restart-timer** – defines the number of seconds that the system waits before reestablishing the session after the prefix limit is exceeded and the BGP session is torn down. This setting is common to both pre- and post-policy prefix limits. By default, the BGP session is reestablished immediately.
- **prevent-teardown** – when set to **true**, prevents the BGP session from being torn down when the prefix limit is exceeded. The default value is **false**.
- **warning-threshold-pct** – defines the threshold, as a percentage of **max-received-routes** received from the peer, at which BGP raises a warning log event. The default value is 90.

4.15.1 Configuring the pre-policy prefix limit for BGP peers

About this task

The pre-policy prefix limit settings can be applied to a specific peer or to a peer group. If there is no setting for a specific peer, the setting for the peer group applies. If there is no setting for either the peer or peer group, the system default pre-policy prefix limit settings apply.

Procedure

To configure the pre-policy prefix limit, use the **prefix-limit-received** command to define values for the **max-received-routes** and the **warning-threshold-pct**. To disable the prefix limit for a particular address family, use the **prefix-limit-received prevent-teardown true** command.

Example: Configure pre-policy prefix limit

The following example configures a peer group with pre-policy prefix limit settings. The settings apply to all members of the specified peer group except in cases where an individual peer has different settings applied.

```
# info with-context network-instance default protocols bgp group custom-peer-group afi-
safi ipv4-unicast ipv4-unicast prefix-limit-received
    network-instance default {
        protocols {
            bgp {
                group custom-pre-policy-group {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            prefix-limit-received {
                                max-received-routes 300000
                                warning-threshold-pct 90
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Disable the pre-policy prefix limit

The following example disables the pre-policy prefix limit for IPv4 routes received from peer 192.0.2.1, so that the BGP session is not torn down when the maximum number of IPv4 routes received from the peer is exceeded.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 192.0.2.1 afi-safi
 ipv4-unicast ipv4-unicast prefix-limit-received
    network-instance default {
        protocols {
            bgp {
                neighbor 192.0.2.1 {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            prefix-limit-received {
                                prevent-teardown true
                            }
                        }
                    }
                }
            }
        }
    }
```

4.15.2 Configuring the post-policy prefix limit for BGP peers

About this task

The post-policy prefix limit settings can be applied to a specific peer or to a peer group. If there is no setting for a specific peer, the setting for the peer group applies. If there is no setting for either the peer or peer group, no post-policy prefix limit is applied.

To enable the post-policy prefix limit, you must configure at least one of the following parameters: **max-received-routes**, **warning-threshold-pct**, or **prevent-teardown**. When one of these parameters is defined, any of the other two parameters that are not otherwise configured have their default values applied.

Procedure

To configure the post-policy prefix limit, use the **prefix-limit-accepted** command to define values for the **max-received-routes** and the **warning-threshold-pct**. To disable the prefix limit for a particular address family, use the **prefix-limit-accepted prevent-teardown true** command.

Example: Configure post-policy prefix limit

The following example configures the maximum number of post-policy IPv4 routes. The settings apply to all members of the specified peer group except in cases where an individual peer has different settings applied.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp group custom-peer-group afi-
safi ipv4-unicast ipv4-unicast prefix-limit-accepted
    network-instance default {
        protocols {
            bgp {
                group custom-post-policy-group {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            prefix-limit-accepted {
                                max-received-routes 300000
                                warning-threshold-pct 90
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Disable the post-policy prefix limit

The following example disables the prefix-limit for IPv4 routes received from peer 192.0.2.2, so that the BGP session is not torn down when the maximum number of IPv4 routes received from the peer is exceeded.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 192.0.2.2 afi-safi
 ipv4-unicast ipv4-unicast prefix-limit-accepted
    network-instance default {
        protocols {
            bgp {
                neighbor 192.0.2.2 {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            prefix-limit-accepted {
                                prevent-teardown true
                            }
                        }
                    }
                }
            }
        }
    }
```

4.15.3 Configuring the prefix limit restart timer for BGP peers

Procedure

When the prefix limit is exceeded the BGP session is torn down. To define the number of seconds the system waits before reestablishing the session, use the **prefix-limit-restart-timer** command. The **prefix-limit-restart-timer** setting applies for both pre- or post-policy prefix limits for a particular AFI-SAFI family and peer (or peer group). The default value is 0 seconds, meaning that the BGP session is reestablished immediately after teardown. If the **prefix-limit-restart-timer** value is set to **forever**, the session remains down indefinitely. In this case, the session can only be reestablished using the **reset-peer** or **soft-clear** command under **tools network-instance protocols bgp**.

Example: Configure the prefix limit restart timer for BGP peers

The following example sets the prefix limit restart timer to 100 seconds.

```
--{ candidate shared default }--[  ]--
```

```
# info with-context network-instance default protocols bgp neighbor 192.0.2.1 timers
    network-instance default {
        protocols {
            bgp {
                neighbor 192.0.2.1 {
                    timers {
                        prefix-limit-restart-timer 100
                    }
                }
            }
        }
    }
```

If the **prefix-limit-restart-timer** is not configured for the peer, the **prefix-limit-restart-timer** setting for the peer group applies. If the **prefix-limit-restart-timer** is not configured for the peer group either, the BGP session with the peer is reestablished immediately after teardown (that is, **prefix-limit-restart-timer** = 0 seconds).

4.16 RT constrained route advertisement

RT constrained route advertisement (RT-constrain) is a mechanism that allows a BGP router to advertise route target (RT) membership information to its BGP peers to indicate interest in receiving only BGP routes tagged with specific RT extended communities. Upon receiving this information, BGP peers restrict the advertised BGP routes to only those requested routes, which can minimize control-plane load for protocol traffic and RIB memory.

The RT membership information is carried in a special type of MP-BGP route called an RTC route; the associated AFI is 1 and the SAFI is 132. For two routers to exchange RT membership NLRI, they must advertise the corresponding AFI/SAFI to each other during capability negotiation. The use of MP-BGP means RT membership NLRI is propagated, loop-free, within an AS and between ASes using well-known BGP route selection and advertisement rules.

Note:
Extended community-based ORF can also be used for RT-based route filtering, but RT-constrain has distinct advantages over extended community-based ORF: RT-constrain is more widely supported, simpler to configure, and its distribution scope is not limited to a direct peer.

4.16.1 RT-constrain on SR Linux for various router types

This section describes how RT-constrain operates on SR Linux for PE, route reflector, and ASBR routers.

PE routers

A PE router originates RT membership NLRI to its peers (often route reflectors) to prevent these peers from sending unnecessary VPN routes to the PE. Usually a PE originates one RTC route for each import route-target associated with a local VPRN service or IP-VRF network-instance. An RT is an extended community with type subcode value 0x02 and type code value 0x00, 0x01, or 0x02.

RTC routes originated by a PE are by default automatically advertised to all RTC peers, without the need for an export policy to accept them. Each RTC route has a prefix (carried in the MP_REACH_NLRI and MP_UNREACH_NLRI attributes) and path attributes (for example, ORIGIN, AS_PATH).

The prefix length (in bits) of an RTC route can be 0 (for the default RTC route), 32, or a number in the range 48 to 96. The prefix value is the concatenation of the origin AS (a 4-byte value representing the 2- or 4-octet AS of the originating router, as configured under **network-instance.protocols.bgp.autonomous-system**) and 0 or 16-64 bits of an RT extended community.

This NLRI format allows RTs originated by the same AS and having the same N most significant bits to be advertised in a single RTC route with prefix length 32+N (N >= 16), but the SR Linux implementation does not make use of this flexibility when originating RTC routes; only the default RTC route with a prefix length of 0, or fully-specified RTC routes with a prefix length of 96, are advertised to RTC peers.

Route reflectors

A route reflector (RR) propagates RTC routes according to a special set of rules listed below. An RR typically advertises the default RTC route to each of its clients so that it receives all VPN routes belonging to the cluster. This is achieved by enabling (setting to `true`) **afi-safi.route-target.send-default-route** in the group or neighbor configuration context. The default RTC route is a special type of RTC route encoded in one of the following ways:

- prefix-length = 0 (SR Linux routers always generate default RTC routes with this encoding)
- prefix-length = 32, encoding only the origin AS value
- prefix-length = 48, encoding the origin AS value, plus 16 bits of an RT type

Sending the default RTC route to a peer conveys a request to receive all VPN routes (regardless of RT extended community) from that peer. The advertisement of the default RTC route to a peer does not suppress other more-specific RTC routes from being sent to that peer.

ASBRs

An ASBR (for example, a model-B ASBR or model-C route reflector) propagates route target membership NLRI to eBGP peers in other autonomous systems to limit received routes to only those needed for services in the local AS and those to be propagated through the local AS. When the RTC route propagation path includes multiple ASNs, SR Linux routers choose only the single best path for reverse advertisement of VPN routes; advertisement of VPN routes using RTC multipaths is not supported.

4.16.2 Best-path selection and RTC route re-advertisement

If multiple RTC routes are received for the same prefix, then standard BGP best-path selection procedures determine the best of these routes. BGP does not check for reachability of the BGP next-hop of RTC routes, so this does not factor into best-path selection.

The best RTC route per prefix is re-advertised to RTC peers based on the following rules:

- The best path for a default RTC route (prefix-length 0, origin AS only with prefix-length 32 or origin AS plus 16 bits of an RT type with prefix-length 48) is never propagated to another peer.
- A PE with only iBGP RTC peers that is neither a route reflector nor an ASBR does not re-advertise the best RTC route to any RTC peer due to standard iBGP split horizon rules.
- A route reflector that receives its best RTC route for a prefix from a client peer re-advertises that route (subject to export policies) to all of its client and non-client iBGP peers (including the originator), per standard RR operation. When the route is re-advertised to client peers the RR should (i) set the ORIGINATOR_ID to its own router ID and (ii) modify the NEXT_HOP to be its local address for the sessions (e.g. system IP).

- A route reflector that receives its best RTC route for a prefix from a non-client peer re-advertises that route (subject to export policies) to all of its client peers, per standard RR operation. Normally no route is advertised to non-client peers in this scenario, but if the RR has a non-best path for the prefix from any of its clients it should advertise the best of the client-advertised paths to all non-client peers. No ORIGINATOR_ID or NEXT_HOP manipulation is required in this case.
- An ASBR which is neither a PE nor a route reflector that receives its best RTC route for a prefix from an iBGP peer re-advertises that route (subject to export policies) to its eBGP peers. It modifies the NEXT_HOP and AS_PATH of the re-advertised route per standard BGP rules. No aggregation of RTC routes is supported.
- An ASBR that is neither a PE nor a route reflector that receives its best RTC route for a prefix from an eBGP peer re-advertises that route (subject to export policies) to its eBGP and iBGP peers. When re-advertised routes are sent to EBGP peers the ABSR modifies the NEXT_HOP and AS_PATH per standard BGP rules. No aggregation of RTC routes is supported.

4.16.3 Using RTC routes to filter advertised routes

When RT-constrain is configured on a session that also supports VPN address families using route targets (for example, VPN-IPv4, VPN-IPv6, MVPN, EVPN), advertisement of the VPN routes is affected as follows:

- When the session comes up, the advertisement of all VPN routes is delayed until the initial set of RTC routes has been received from the peer (that is, all RTC routes in the peer’s RIB-OUT); this is the `waiting` state. The `waiting` state ends when an End-of-RIB marker for AFI/SAFI=1/132 is received from the peer, or a certain amount of time has elapsed since the session was established (this amount of time is hard-coded to 60 seconds and applies when the peer does not support sending the End-of-RIB marker). When the `waiting` state ends, VPN routes are sent to the peer based on received RTC routes (see below), and the session transitions to the `ready` state.
- When the session is in the `ready` state, received RTC routes are acted upon immediately. SR Linux does not expect and wait for a ROUTE REFRESH message from the peer.
  If S1 is the set of routes previously advertised to the peer, and S2 is the set of routes to be advertised based on the most recent received RTC routes, then:
  - the set of routes in S1, but not in S2, are withdrawn immediately (subject to MRAI)
  - the set of routes in S2, but not in S1, are advertised immediately (subject to MRAI)
- If a default RTC route (best or non-best) is received from an eBGP or iBGP peer, the VPN routes advertised to the peer is the set of VPN routes in the LOC-RIB that meet all of the following conditions:
  - are eligible for advertisement to the eBGP or iBGP peer per BGP route advertisement rules
  - have not been rejected by manually configured export policies
  - have not been advertised to the peer
- If an RTC route for a prefix (origin-AS = A1, RT = A2/n, n > 48), best or non-best, is received from an iBGP peer in autonomous system A1, the VPN routes advertised to the iBGP peer is the set of VPN routes in the LOC_RIB that meet all of the following conditions:
  - are eligible for advertisement to the iBGP peer per BGP route advertisement rules
  - have not been rejected by manually configured export policies
  - carry at least one route target extended community with value A2 in the n most-significant bits
  - have not been advertised to the peer

- If the best RTC route for a prefix (origin-AS = A1, RT = A2/n, n > 48) is received from an iBGP peer in autonomous system B, the VPN routes advertised to the iBGP peer is the set of VPN routes in the LOC-RIB that meet all of the following conditions:
  - are eligible for advertisement to the iBGP peer per BGP route advertisement rules
  - have not been rejected by manually configured export policies
  - carry at least one route target extended community with value A2 in the n most-significant bits
  - have not been advertised to the peer
- If the best RTC route for a prefix (origin-AS = A1, RT = A2/n, n > 48) is received from an eBGP peer, the VPN routes advertised to the eBGP peer is the set of VPN routes in the LOC-RIB that meet all of the following conditions:
  - are eligible for advertisement to the eBGP peer per BGP route advertisement rules
  - have not been rejected by manually configured export policies
  - carry at least one RT extended community with value A2 in the n most-significant bits
  - have not been advertised to the peer

4.16.4 BGP RIB YANG model for RTC Routes

The following table lists the information available in the SR Linux BGP RIB YANG model for RTC routes in the RIB-IN and RIB-OUT contexts.

Table 5: RTC route information available in the SR Linux BGP RIB YANG model

`bgp-rib.afi-safi.route-target.rib-in-out.rib-in-pre.routes`	Contains the full set of RTC routes received from all peers.
`bgp-rib.afi-safi.route-target.rib-in-out.rib-in-post.routes`	Contains the full set of RTC routes received from all peers, after import policy modification.
`bgp-rib.afi-safi.route-target.rib-in-out.rib-out-post.routes`	Contains the full set of RTC routes advertised to each peer, after export policy modification.

4.17 BGP configuration management

Managing the BGP configuration in SR Linux can include the following tasks:

- Modifying an AS number
- Deleting a BGP neighbor
- Deleting a BGP peer group
- Resetting BGP peer connections

4.17.1 Modifying an ASN

Procedure

You can modify the ASN on the router, but the new ASN does not take effect until the BGP instance is restarted, either by administratively disabling/enabling the BGP instance, or by rebooting the system with the new configuration.

Example

```
--{ * candidate shared default }--[ network-instance default ]--
# protocols bgp autonomous-system 95002
# protocols bgp admin-state disable
# protocols bgp admin-state enable
```

All established BGP sessions are taken down when the BGP instance is disabled.

4.17.2 Deleting a BGP neighbor

Procedure

Use the **delete** command to delete a BGP neighbor from the configuration.

Example

```
--{ * candidate shared default }--[ network-instance default ]--
# delete protocols bgp neighbor 192.168.11.1
```

4.17.3 Deleting a BGP peer group

Procedure

Use the **delete** command to delete the settings for a BGP peer group from the configuration.

Example

```
--{ * candidate shared default }--[ network-instance default ]--
# delete protocols bgp group headquarters1
```

4.17.4 Resetting BGP peer connections

Procedure

To refresh the connections between BGP neighbors, you can issue a hard or soft reset. A hard-reset tears down the TCP connections and returns to IDLE state. A soft-reset sends route-refresh messages to each peer. The hard or soft reset can be issued to a specific peer, to peers in a specific peer-group, or to peers with a specific ASN.

Example: Issue a hard reset

The following command hard-resets the connections to the BGP neighbors in a peer group that have a specified ASN. The hard reset applies both to configured peers and dynamic peers.

```
# tools network-instance default protocols bgp group headquarters1 reset-peer peer-as
 95002
/network-instance[name=default]/protocols/bgp/group[group-name=headquarters1]:
    Successfully executed the tools clear command.
```

Example: Issue a soft reset

The following command soft-resets the connection to BGP neighbors that have a specified ASN. The soft reset applies both to configured peers and dynamic peers.

```
# tools network-instance default protocols bgp soft-clear peer-as 95002
/network-instance[name=default]/protocols/bgp:
    Successfully executed the tools clear command.
```

4.18 BGP shortcuts

Note: BGP shortcuts is supported on 7730 SXR and 7250 IXR platforms.

With BGP shortcuts, SR Linux can include LDP LSPs, segment routing (SR-ISIS) tunnels, or TE Policy SR-MPLS tunnels in the BGP algorithm calculations. In this case, tunnels operate as logical interfaces directly connected to remote nodes in the network. Because the BGP algorithm treats the tunnels in the same way as a physical interface (being a potential output interface), the algorithm can select a destination node together with an output tunnel to resolve the next-hop, using the tunnel as a shortcut through the network to the destination. With BGP shortcuts enabled, next-hop resolution determines whether to use a local interface or a tunnel to resolve the BGP next-hop.

Tunnel resolution mode

As part of the configuration for BGP shortcuts, you must define the tunnel-resolution mode (prefer/required/disabled). This mode determines the order of preference and fallback of using tunnels in the tunnel table to resolve the next-hop instead of using routes in the FIB.

4.18.1 Configuring BGP shortcuts over segment routing

Procedure

Step 1. In the default network instance, define the tunnel-resolution mode for the BGP protocol.

This setting determines the order of preference and the fallback when using tunnels in the tunnel table instead of routes in the FIB. Available options are as follows:

- **require**
  requires tunnel table lookup only
- **prefer**
  prefers tunnel table lookup over FIB lookup
- **disabled (default)**

performs FIB lookup only

Step 2. Set the allowed tunnel types for next-hop resolution.

Example: Configure IPv4 BGP shortcuts

The following example shows the BGP next-hop resolution configuration to allow IPv4 SR-ISIS tunnels, with the tunnel mode set to `prefer`.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast ipv4-
unicast next-hop-resolution ipv4-next-hops tunnel-resolution
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    ipv4-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                tunnel-resolution {
                                    mode prefer
                                    allowed-tunnel-types [
                                        sr-isis
                                    ]
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Configure IPv6 BGP shortcuts

The following example shows the BGP next-hop resolution configuration to allow IPv6 SR-ISIS tunnels, with the tunnel mode set to `prefer`.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols bgp afi-safi ipv6-unicast ipv6-
unicast next-hop-resolution ipv6-next-hops tunnel-resolution
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv6-unicast {
                    ipv6-unicast {
                        next-hop-resolution {
                            ipv6-next-hops {
                                tunnel-resolution {
                                    mode prefer
                                    allowed-tunnel-types [
                                        sr-isis
                                    ]
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

4.19 BGP TCP MSS

BGP uses TCP transport, and BGP messages are carried as TCP segments. SR Linux allows you to control the Maximum Segment Size (MSS) for each TCP segment based on the Path MTU discovery settings.

Path MTU discovery can be enabled or disabled per network instance in SR Linux. The default is enabled.

Within the BGP hierarchy, path MTU discovery can be enabled and disabled at different configuration levels. The supported configuration paths are:

- `network-instance.protocols.bgp.transport.mtu-discovery`
- `network-instance.protocols.bgp.group.transport.mtu-discovery`
- `network-instance.protocols.bgp.neighbor.transport.mtu-discovery`

BGP path MTU discovery by default inherits the value from the network instance for all BGP sessions. It can be overruled by the above config. When an ICMP fragmentation-needed message is received and BGP path MTU discovery is disabled, the system reduces the MTU for the BGP session according to the ICMP message, subject to the lower bound configured under the system-level `min-path-mtu`.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default
    network-instance default {
        mtu {
            path-mtu-discovery true
        }
    }
```

```
--{ * candidate shared default }--[ ]--
# info with-context system mtu
    system {
        mtu {
            min-path-mtu 552
        }
    }
```

4.19.1 Configuring BGP TCP MSS

Procedure

The maximum size of each TCP segment is controlled by configuring the TCP MSS (`tcp-mss`) value.

SR Linux supports configuring TCP MSS at BGP instance, peer group, and neighbor configuration levels. The supported range for the `tcp-mss` value is 536-9446 bytes, and the default value is 1024 bytes.

The value of `tcp-mss` gets inherited down the configuration levels within the BGP hierarchy. If no `tcp-mss` is configured for a BGP neighbor, the `tcp-mss` value is taken from the BGP peer group, if it is configured there, or else is taken from the BGP instance. The default BGP instance `tcp-mss` value is used if neither the BGP peer group or the neighbor has a configured `tcp-mss`.

If the configured or inherited `tcp-mss` value is higher than the BGP path MTU value, the `tcp-mss` value is ignored, and the BGP path MTU value is used as the operational TCP MSS.

Example: Configure BGP instance `tcp-mss`

The following example configures the BGP instance `tcp-mss` value.

```
info with-context from state network-instance default protocols bgp trans
port tcp-mss
    network-instance default {
        protocols {
            bgp {
                transport {
                    tcp-mss 1024
                }
            }
        }
    }
```

Example: Configure BGP peer group `tcp-mss`

The following example configures the BGP peer group `tcp-mss`.

```
info with-context from state network-instance default protocols bgp group trans
port tcp-mss
    network-instance default {
        protocols {
            bgp {
                group test {
                    transport {
                        tcp-mss 1024
                    }
                }
            }
        }
    }
```

Example: Configure BGP neighbor `tcp-mss`

The following example configures the BGP neighbor `tcp-mss` .

```
info with-context from state network-instance default protocols bgp  neighbor 1.1.1.1
 transport tcp-mss

     network-instance default {
        protocols {
            bgp {
                neighbor 192.168.0.1 {
                    transport {
                        tcp-mss 1012
                    }
                }
            }
        }
    }
```

If the configured or inherited `tcp-mss` value is higher than the operational path MTU value, the `tcp-mss` value is ignored and the path MTU value is used as the operational TCP MSS.

4.20 Error handling for BGP update messages

BGP update messages are used to transfer routing information between BGP peers. Errors in some BGP update messages are considered critical; for example, if the Network Layer Reachability Information (NLRI) cannot be extracted and parsed from an update message, it is a critical error. Errors in other BGP update messages are considered non-critical; for example, errors such as incorrect attribute flag settings, missing mandatory path attributes, incorrect next-hop length or format, and so on, are non-critical errors.

In SR Linux, critical errors in BGP update messages trigger a session reset. Non-critical errors are handled using the treat-as-withdraw or attribute-discard approaches to error handling. This error-handling behavior for BGP update messages is not configurable in SR Linux.

4.21 BGP multipath

BGP multipath is the ability to install a BGP route into the FIB so that the ECMP algorithm load-balances traffic across multiple BGP next-hops that come from different multipath-eligible RIB-INs for the same prefix or NLRI address family.

In a network-instance, you can enable BGP multipath for an address family and specify the maximum number of BGP ECMP next-hops for BGP routes that have an NLRI belonging to the address family.

4.21.1 Configuring BGP multipath

Procedure

To configure BGP multipath, set the `allow-multiple-as` parameter to `true`. When you do this, BGP is allowed to build a multipath set using BGP routes with a different neighbor AS (the most recent AS in the AS_PATH).

When `allow-multiple-as` is set to `false` (the default), BGP is only allowed to use non-best paths for ECMP if they meet the multipath criteria, and they have the same neighbor AS as the best path.

The `maximum-paths` parameter configures the maximum number of BGP ECMP next-hops for BGP routes with an NLRI belonging to the specified address family. Note the following:

Note:

- When a BGP prefix is covered by a `resilient-hash-prefix` entry, the maximum number of BGP next-hops used for load balancing is controlled by the `network-instance ip-load-balancing resilient-hash-prefix <ip-prefix> max-paths` value.
- When BGP is resolved by an unweighted, non-resilient-hash IGP route, the maximum number of paths towards the BGP next-hop is controlled by the IGP configuration; for example, the IS-IS `max-ecmp-paths` value.
- When BGP is resolved by a weighted, non-resilient-hash IGP route, the maximum number of paths towards the BGP next-hop is controlled by the IGP configuration; for example, the IS-IS `max-ecmp-hash-buckets-per-next-hop-group` value.
- When BGP is resolved by a static, non-resilient-hash route, the maximum number of paths towards the BGP next-hop is controlled by the static NHG configuration.

Example: Enable BGP multipath

The following example enables BGP multipath for eBGP and specifies the maximum number of BGP ECMP next-hops for BGP routes with an NLRI belonging to the `ipv4-unicast` address family:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance n1 protocols bgp afi-safi ipv4-unicast
    network-instance n1 {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    multipath {
                        allow-multiple-as true
                        ebgp {
                            maximum-paths 10
                        }
                    }
                }
            }
        }
    }
```

4.22 BGP weighted ECMP using the link-bandwidth EC

Note: SR Linux supports wECMP for BGP IPv4 and IPv6 routes on 7250 IXR 6e/10e/18e/X1b/X3b and 7220 IXR H4/H5 systems.

Weighted ECMP (wECMP), also known as UCMP or unequal cost multipath, is the ability to install a multipath route (a route with multiple next-hops) into the FIB so that flow hashing directs flows to each next-hop in a ratio proportional to the weight of each next-hop.

The weight BGP associates with each multipath next-hop comes from the link bandwidth extended community (EC) that is associated with each path (RIB-IN). The link bandwidth EC can be added to a BGP route by an upstream originator/advertiser of the route, or it can be added by the receiver of the route.

When the feature is enabled for eBGP or iBGP for the AFI-SAFI associated with a route:

- If the best path for the prefix was received from an eBGP/iBGP peer
- and the eBGP/iBGP peer is not part of a peer-group for which the feature is disabled for the AFI-SAFI associated with the route
- and all the multipath-eligible paths have a link bandwidth EC
- then the route is programmed into the FIB with support of weighted ECMP

The number of multipaths is controlled by the **network-instance.protocols.bgp.afi-safi.multipath.maximum-paths** value. See BGP multipath.

The system reduces the bandwidth values to a normalized set of weights that attempts to maintain the ratio of the bandwidth values, but collectively do not exceed the configured maximum for ECMP buckets for each NHG, set with the **bgp.max-ecmp-buckets-per-next-hop-group** parameter.

SR Linux supports using the link bandwidth EC as match criteria and adding, removing, or replacing the link-bandwidth EC as an action in a routing policy. See Routing policies.

Note:

- In a BGP-over-BGP configuration, only the top-level BGP route (NHG) is programmed for wECMP.
- When a tunnel is programmed as an indirect next-hop (such as with GRE tunnels) the weighted ECMP programming of the NHG corresponding to the resolving BGP route is copied over to the NHG created for the tunnel; that is, when a GRE tunnel is resolved by a weighted BGP route, it is load-balanced according to the BGP weights.
- When a BGP route is eligible for both wECMP and resilient hashing, the **resilient-hash-prefix** configuration takes precedence.

  For example, if **max-ecmp-hash-buckets-per-next-hop-group** is 128 and the **resilent-hash-prefix** configuration for the prefix specifies **max-paths = 16** and **hash-buckets-per-path = 16** then SR Linux uses 256 hash buckets for this prefix and caps the number of ECMP members at 16.

4.22.1 BGP link bandwidth extended community

The BGP link bandwidth extended community is a non-transitive 2-octet, AS-specificEC, identified by type 0x40 and sub-type 0x04.

The 2-byte Global Administrator field encodes the AS number of the router that attaches the EC (AS_TRANS is used if the router has a 4-octet ASN).

The 4-byte Local Administrator field encodes a bandwidth in byte/s, expressed in IEEE floating point format (binary32).

Receiving routes with the link bandwidth EC

SR Linux accepts and stores the link bandwidth EC when it is received in any type of BGP route, from any type of peer (eBGP or iBGP), with the following exception: If BGP receives a route with two or more link-bandwidth ECs, then all of them are removed from the route before it is stored in the RIB.

Note: The link-bandwidth EC is not accepted from eBGP peers and is not advertised to eBGP peers if the type is 0x40.

Advertising Routes with Link Bandwidth EC

When a BGP route with a link bandwidth EC is selected for advertisement to a peer (as a best path or add-path) the advertised route includes this link bandwidth EC except as noted below:

- When propagating routes between iBGP peers, if the BGP next-hop of the advertised route is changed by a BGP export policy action, then a link-bandwidth community added by another router is removed from the RIB-OUT (but not the RIB-IN).
- When propagating routes between iBGP peers, if the BGP next-hop of the advertised route is changed by a BGP next-hop-self command then a link-bandwidth community added by another router is removed from the RIB-OUT (but not the RIB-IN).

When a route with a link bandwidth EC is received from an eBGP peer, or selected for advertisement to an eBGP peer, SR Linux considers that it was added locally and the above two exceptions do not apply, meaning that the EC is not stripped from the RIB-OUT.

When a BGP route with a link bandwidth EC is exported from an IP-VRF as a VPN-IP route, the link bandwidth EC is carried with the route so that it is included in the VPN-IP route sent to BGP peers of the default network-instance.

4.22.2 Configuring BGP wECMP routes using link bandwidth EC

Procedure

To enable BGP wECMP using the link bandwidth EC, you set **network-instance.protocols.bgp.afi-safi.multipath.weighted-ecmp.admin-state** to **enable** for eBGP or iBGP.

In the BGP RIB state model, the link bandwidth EC associated with a route is part of its linked **attr-set**, with the link-bandwidth EC appearing as a string in the ext-community list. The string is displayed in the format **link-bandwidth:<as-num>:<bandwidth>[kMGT]**.

Example: Enable BGP wECMP using the link bandwidth EC for eBGP

The following example enables BGP wECMP using the link bandwidth EC for IPv4 unicast routes for which the best path was received from an eBGP peer.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast multipath
 ebgp
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    multipath {
                        ebgp {
                            weighted-ecmp {
                                admin-state enable
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Enable BGP wECMP using the link bandwidth EC for iBGP

The following example enables BGP wECMP using the link bandwidth EC for IPv4 unicast routes for which the best path was received from an iBGP peer.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols bgp afi-safi ipv4-unicast multipath ebgp
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    multipath {
                        ibgp {
                            weighted-ecmp {
                                admin-state enable
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Display the link bandwidth EC path attribute

To display the link-bandwidth EC associated with a route, use the **info from state bgp-rib attr-sets attr-set** command within the network-instance context; for example

```
--{ candidate shared default }--[ network-instance default ]--
# info from state bgp-rib attr-sets attr-set <attr-id> communities ext-community
    communities {
        ext-community {
            link-bandwidth:65000:1T
            }
        }
    }
```

4.22.3 Aggregating link bandwidth values

Procedure

By default, whenever the best RIB-IN for a prefix has a link bandwidth EC, and the router is not required to remove the EC during re-advertisement of the route, BGP simply copies the EC to the RIB-OUT. However, in some cases it may be preferable to encode the total available bandwidth to the destination by summing the link-bandwidth values in each of the used multipaths. To do this, you can enable the **aggregate-used-paths** option in the BGP group or neighbor configuration.

Aggregation is only done when routes are advertised with next-hop-self. If BGP Add-Path is also enabled toward the peer, then all of the Add-Paths advertised to the peer encode the aggregated bandwidth in a link bandwidth EC.

Note: Aggregation of link bandwidth values is supported only for the default network-instance.

Example: Enable aggregation of link bandwidth values

The following example enables aggregation for link bandwidth values for IPv4 unicast routes for a BGP neighbor.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 10.10.10.10
    network-instance default {
        protocols {
            bgp {
                neighbor 10.10.10.10 {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            link-bandwidth {
                                aggregate-used-paths true
                            }
                        }
                    }
                }
            }
        }
    }
```

4.22.4 Automatic generation of link-bandwidth EC based on interface speed

It may be necessary for the system to add a link-bandwidth EC to routes received from specified eBGP peers. You can configure SR Linux to automatically generate the link-bandwidth EC based on the underlying interface speed.

When you enable this feature for a BGP neighbor or group, SR Linux detects the physical bandwidth and attaches a link bandwidth EC to the RIB-IN that encodes the interface speed toward the single-hop eBGP or iBGP peer, as follows:

- If the underlying interface is a single port, the interface speed is the operational speed of the interface.
- If the underlying interface is a LAG, the interface speed is the sum of the interface speeds of the operational LAG member ports. LAG members configured with `forwarding-viable=false` are excluded from the sum.

You can configure the link-bandwidth EC to be transitive or non-transitive. When the link-bandwidth EC is transitive, and the RIB-OUT is destined for an eBGP peer, the EC is propagated across the AS boundary; if the link-bandwidth EC is non-transitive, it is removed from the RIB-OUT.

If the route is the best RIB-IN for the prefix, weighted ECMP is enabled so that it applies to the route, and all the multipath-eligible paths have a link-bandwidth EC (any mix of transitive and non-transitive), then the route is programmed into the FIB with weighted ECMP.

Interaction with link aggregation

If the route is selected for advertisement and `aggregate-used-paths=true` is configured for the route, the RIB-OUT has a regenerated transitive-type link-bandwidth EC that aggregates the bandwidth values in all the used multipaths (regardless of whether they are transitive or non-transitive).

If the route is selected for advertisement and `aggregate-used-paths=false` is configured for the route:

- When the RIB-OUT is destined for an eBGP peer, the link-bandwidth EC is removed from the RIB-OUT (non-transitive) or transmitted as-is (transitive).
- When the RIB-OUT is destined for an iBGP peer:
  - If the route was received from an iBGP peer, and the BGP next-hop of the advertised route is changed by a BGP export policy action or by the **next-hop-self** command, link-bandwidth EC is removed from the RIB-OUT.
  - If the route was received from an eBGP peer, it is transmitted as-is in the RIB-OUT.

Adjusting to interface speed changes

When a link-bandwidth extended community is added to a RIB-IN, BGP continuously monitors the speed of the interface that is the source of the bandwidth value. The speed of the interface may change because of link-speed renegotiation, LAG membership changes, or other reasons.

Whenever the interface speed decreases, BGP updates the RIB-IN immediately so the datapath load balancing adapts as quickly as possible.

Whenever the interface speed increases, BGP delays the update of the RIB-IN by a configurable number of seconds (default 10 seconds), configured with the `hold-down-time` parameter, which is applicable to the peer and AFI-SAFI of the route. The `hold-down-time` delays the reaction of the datapath and the advertisement of the RIB-OUT when the route is re-advertised.

4.22.4.1 Configuring automatic generation of link-bandwidth EC

Procedure

To configure automatic generation of the link-bandwidth EC, you enable it for a BGP peer group or neighbor, and optionally set the hold-down time and whether the link-bandwidth EC is transitive.

Example

The following example enables automatic generation of the link-bandwidth EC for IPv4 unicast routes for a BGP neighbor.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 10.10.10.10
    network-instance default {
        protocols {
            bgp {
                neighbor 10.10.10.10 {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            link-bandwidth {
                                auto-from-interface-speed {
                                    admin-state enable
                                    hold-down-time 15
                                    transitive true
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

In this example, the **hold-down-time** is set to 15 seconds. If the interface speed increases, the system delays the update of the RIB-IN by this amount of time. The link-bandwidth EC is configured to be transitive, allowing it to be propagated across the AS boundary.

4.23 BGP fast reroute

Note: In the current release, BGP FRR is supported on 7730 SXR and 7250 IXR platforms with the following address families:

- IPv4 unicast (7250 IXR only)
- IPv6 unicast (7250 IXR only)
- IPv4 labeled unicast
- IPv6 labeled unicast
- IPv4 VPN (l3vpn-ipv4-unicast) (7250 IXR only)
- IPv6 VPN (l3vpn-ipv6-unicast) (7250 IXR only)

BGP fast reroute (FRR) combines indirection techniques in the forwarding plane and pre-computation of BGP backup paths in the control plane to support fast reroute of BGP traffic around unreachable or failed

BGP next-hops. BGP FRR serves an important role in ensuring high availability of services that rely on BGP for transport.

When BGP FRR is enabled for routes of a specific address family, BGP attempts to program a backup path for every destination that has only a single primary next-hop. Because ECMP routes toward a destination already have path resiliency, no backup path is programmed for those routes. The FRR backup path is the best BGP route that passes through a different next-hop than the primary path. Backup paths are pre-programmed into the FIB to ensure they are ready for switchover immediately after the primary path fails.

The switchover to the backup path can be triggered by multiple events, including:

- IGP topology change that makes the BGP next-hop of the primary path unreachable
- BFD session failure that implies the BGP next-hop of the primary path is unreachable

BGP FRR also uses indirection techniques in datapath programming to allow a single update from the control plane to trigger the failover for multiple prefixes at once.

4.23.1 Configuring BGP FRR

Procedure

To configure BGP FRR, use the **backup-paths** command to install a backup path for every NLRI in the address family when a suitable one exists. You can also optionally configure BGP FRR together with BGP add-path.

Example: Configure BGP FRR

The following example configures BGP FRR for IPv4 unicast addresses.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast ipv4-
unicast backup-paths
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    ipv4-unicast {
                        backup-paths {
                            install true
                        }
                    }
                }
            }
        }
    }
```

4.24 Configuring BGP send-community type

About this task

Note: BGP send-community type is supported on 7730 SXR, 7250 IXR, 7220 IXR, and 7215 IXS systems.

Procedure

You can specify the type of BGP community the router sends to all neighbors for a specified AFI-SAFI family using the **bgp send-community-type** command.

Valid options for the community type include **extended**, **large**, and **standard**. If any community type is not specified, that type is stripped from advertised routes for the specified AFI-SAFI family. In addition, if **none** is included in the **send-community-type** list, all other defined community types are ignored for the specified AFI-SAFI family

By default all three community types are sent to all peers.

Example: Configure BGP send-community type

The following example enables all three community types for the IPv4 unicast family.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast send-
community-type
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    send-community-type [
                        standard
                        extended
                        large
                    ]
                }
            }
        }
    }
```

4.25 BGP Link-State

Note: BGP-LS is supported on 7250 IXR Gen 2, 7250 IXR Gen 2c+, 7250 IXR Gen 3, and 7730 SXR platforms.

BGP Link-State (BGP-LS) is an extension of BGP that is designed to distribute link-state information from IGPs such as IS-IS including node, link, and prefix information.

BGP-LS supports the following capabilities:

- network topology sharing — provides real-time network topology information to external components such as topology servers, Application Layer Traffic Optimization (ALTO) servers, and Path Computation Element (PCE) servers
- traffic engineering — shares detailed link-state and traffic engineering information to optimize routing of network traffic
- path computation — helps PCEs to calculate optimal paths based on the current network state

BGP-LS provides external ALTO and PCE servers with topology information for multi-area or multilevel networks. Through the use of one or more BGP-LS speakers per area or level, the external ALTO or PCE servers can receive full topology information for the entire network. The BGP-LS information can be distributed either through direct BGP peerings or through route reflectors supporting BGP-LS to minimize peering requirements.

BGP-LS NLRIs are subject to route policies that allow control over what information is advertised. Only per-AFI/SAFI policies are supported.

BGP-LS nodes operate in one of the following roles:

- **BGP-LS producer**

  An SR Linux node operating as a BGP-LS producer originates and advertises BGP-LS NLRIs and BGP-LS path attributes. The node's local IGP LSDB and traffic engineering database are exported to BGP. Locally originated BGP-LS NLRIs and path attributes are always selected as best paths. These paths are stored in the BGP Local-RIB and copied to the BGP RIB-Out as required for advertisement to BGP-LS peers.

- **BGP-LS propagator**

  An SR Linux node operating as a BGP-LS propagator acts as a route reflector for the BGP-LS address family. Received BGP-LS NLRI and path attributes are stored in the RIB-IN-pre and RIB-IN-post. Best paths are determined by the BGP best-path selection process. These paths are stored in the BGP Local-RIB and copied to the BGP RIB-Out as required for advertisement to BGP-LS peers.

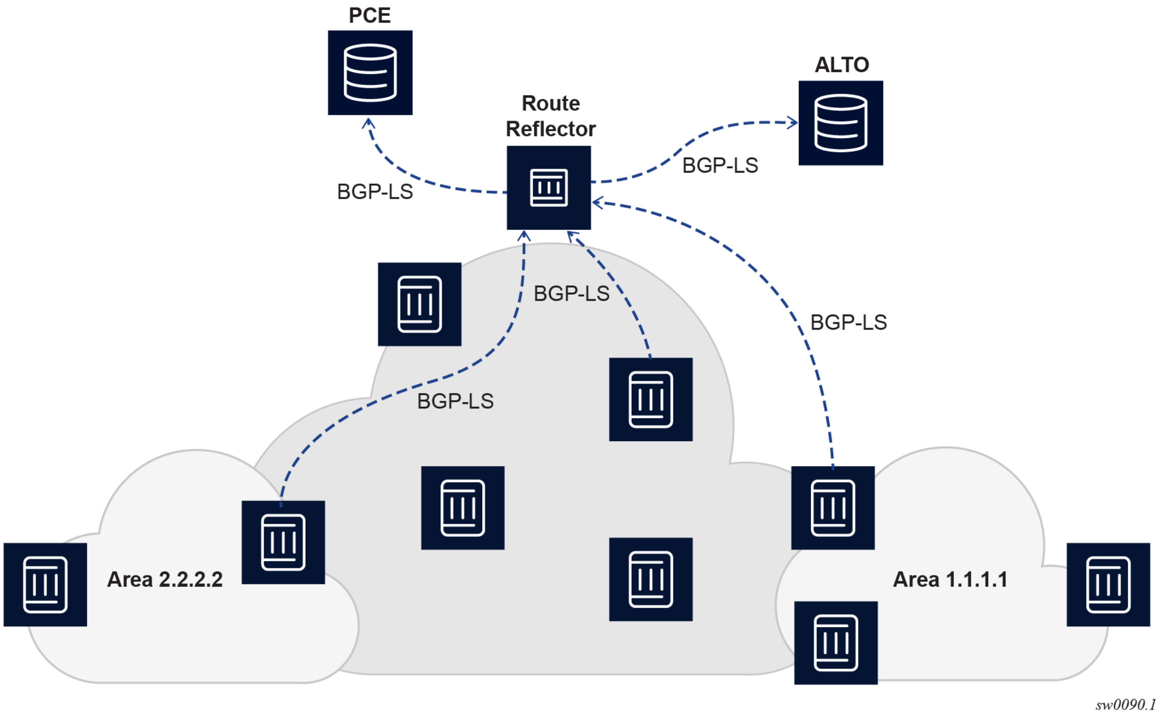
  The roles of BGP-LS producer and propagator are not mutually exclusive.

- **BGP-LS consumer**

  BGP-LS consumers use the BGP-LS data to populate their local TE database for CSPF-based path computations. SR Linux nodes do not operate as BGP-LS consumers. This role is served by the external ALTO and PCE servers.

The following figure shows an example BGP-LS network.

Figure 1: Example BGP-LS network



4.25.1 Supported NLRI and link-state attributes in BGP-LS

The following tables describe BGP-LS Network Layer Reachability Information (NLRI) and link-state attributes supported in SR Linux.

Node NLRI and link-state attributes

Table 6: Node NLRI and link-state attributes support

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protoco ID 1, 2 (IS-IS L1 and L2)	Protocol ID 6 (OSPFv3)
Node NLRI	Node Descriptor	512	Autonomous System	From /network-instance/traffic-engineering/autonomous-system	Not supported
Node NLRI	Node Descriptor	513	BGP-LS Identifier	N/A (this TLV has been deprecated in RFC 9552)	N/A
Node NLRI	Node Descriptor	514	OSPF Area ID	N/A	Not supported
Node NLRI	Node Descriptor	515	IGP Router-ID	6 or 7 byte system-id (extracted from net)	Not supported
Node NLRI	Node Descriptor	516	BGP Router-ID	N/A	N/A
Node NLRI	Node Descriptor	517	BGP Confederation-Member	N/A	N/A
Node NLRI	Node Attribute	263	Multi-Topology Identifier	Not supported	N/A
Node NLRI	Node Attribute	266	Node MSD	Supported Supports advertising SRv6-specific MSD type/values SRH Max Segments Left Type, SRH Max End Pop Type, SRH Max H.encaps Type, SRH Max End D Type.	Not supported
Node NLRI	Node Attribute	1024	Node Flag Bits	Supported Note: bits O, A, and B	Not supported
Node NLRI	Node Attribute	1025	Opaque Node Attribute	Not supported	Not supported
Node NLRI	Node Attribute	1026	Node Name	Supported	Not supported

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protoco ID 1, 2 (IS-IS L1 and L2)	Protocol ID 6 (OSPFv3)
Node NLRI	Node Attribute	1027	IS-IS Area Identifier	Supported (extracted from net)	N/A
Node NLRI	Node Attribute	1028	IPv4 Router-ID of Local Node	Supported	Not supported
Node NLRI	Node Attribute	1029	IPv6 Router-ID of Local Node	Supported	Not supported
Node NLRI	Node Attribute	1032	S-BFD Discriminators	Supported	Not supported
Node NLRI	Node Attribute	1034	SR Capabilities	Supported	Not supported
Node NLRI	Node Attribute	1035	SR Algorithm	Supported	Not supported
Node NLRI	Node Attribute	1036	SR Local Block	Supported	Not supported
Node NLRI	Node Attribute	1037	SRMS Preference	Not supported	Not supported
Node NLRI	Node Attribute	1038	SRv6 Capabilities	Supported	Not supported
Node NLRI	Node Attribute	1039	Flex-Algo Definition	Supported	Not supported
Node NLRI	Node Attribute	1040	Flex-Algo Exclude-any affinity	Supported	Not supported
Node NLRI	Node Attribute	1041	Flex-Algo Include-any affinity	Supported	Not supported
Node NLRI	Node Attribute	1042	Flex-Algo Include-all affinity	Supported	Not supported
Node NLRI	Node Attribute	1043	Flex-Algo Definition Flags	Supported	Not supported
Node NLRI	Node Attribute	1045	Flex-Algo Exclude SRLG	Not supported	Not supported
Node NLRI	Node Attribute	1046	Flex-Algo Unsupported	Not supported	Not supported

Link NLRI and link-state attributes

Table 7: Link NLRI and link-state attributes support

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protocol ID 1, 2 (IS-IS L1 and L2)	Protocol ID 3 (OSPFv2)	Protocol ID 6 (OSPFv3)
Link NLRI	Link Descriptor	258	Link Local/ Remote Identifiers	Supported	Not supported	Not supported
Link NLRI	Link Descriptor	259	IPv4 interface address	Supported	Not supported	N/A
Link NLRI	Link Descriptor	260	IPv4 neighbor address	Supported	Not supported	N/A
Link NLRI	Link Descriptor	261	IPv6 interface address	Supported	N/A	Not supported
Link NLRI	Link Descriptor	262	IPv6 neighbor address	Supported	N/A	Not supported
Link NLRI	Link Descriptor	263	Multi-Topology Identifier	Not supported	N/A	N/A
Link NLRI	Link Attribute	267	Link MSD	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1028	IPv4 Router-ID of Local Node	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1029	IPv6 Router-ID of Local Node	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1030	IPv4 Router-ID of Remote Node	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1031	IPv6 Router-ID of Remote Node	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1088	Administrative group (color)	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1089	Maximum link bandwidth	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1090	Max. reservable link bandwidth	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1091	Unreserved bandwidth	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1092	TE Default Metric	Supported	Not supported	Not supported

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protocol ID 1, 2 (IS-IS L1 and L2)	Protocol ID 3 (OSPFv2)	Protocol ID 6 (OSPFv3)
Link NLRI	Link Attribute	1093	Link Protection Type	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1094	MPLS Protocol Mask	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1095	IGP Metric	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1096	Shared Risk Link Group	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1097	Opaque Link Attribute	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1098	Link Name	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1099	Adjacency Segment Identifier	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1100	LAN Adjacency Segment Identifier	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1101	Peer Node SID	N/A	N/A	N/A
Link NLRI	Link Attribute	1102	PeerAdj SID	N/A	N/A	N/A
Link NLRI	Link Attribute	1103	Peer Set SID	N/A	N/A	N/A
Link NLRI	Link Attribute	1106	SRv6 End.X SID	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1107	SRv6 LAN End.X SID	Supported	N/A	N/A
Link NLRI	Link Attribute	1108	OSPFv3 SRv6 LAN End.X SID	N/A	N/A	Not supported
Link NLRI	Link Attribute	1114	Unidirectional Link Delay	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1115	Min/Max Unidirectional Link Delay	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1116	Unidirectional Delay Variation	Supported, not originated	Not supported	Not supported

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protocol ID 1, 2 (IS-IS L1 and L2)	Protocol ID 3 (OSPFv2)	Protocol ID 6 (OSPFv3)
Link NLRI	Link Attribute	1117	Unidirectional Link Loss	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1118	Unidirectional Residual Bandwidth	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1119	Unidirectional Available Bandwidth	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1120	Unidirectional Utilized Bandwidth	Supported, not originated	Not supported	Not supported
Link NLRI	Link Attribute	1121	Graceful Link Shutdown	N/A	Not supported	Not supported
Link NLRI	Link Attribute	1122	Application Specific Link Attributes	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1172	L2 Bundle Member	Not supported	Not supported	Not supported
Link NLRI	Link Attribute	1173	Extended Admin Group	Supported	Not supported	Not supported
Link NLRI	Link Attribute	1252	SRv6 SID Structure	Supported	Not supported	Not supported

IPv4/IPv6 prefix NLRI and link-state attributes

Table 8: IPv4/IPv6 prefix NLRI and link-state attributes support

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protocol ID 1, 2 (IS-IS L1 and L2)	Protocol ID 3 (OSPFv2)	Protocol ID 6 (OSPFv3)
IPv4/IPv6 Prefix NLRI	Prefix Descriptor	263	Multi-Topology Identifier	Supported	N/A	N/A
IPv4/IPv6 Prefix NLRI	Prefix Descriptor	264	OSPF Route Type	N/A	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Descriptor	265	IP Reachability Information	Supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Descriptor	1044	Flex-Algo Prefix Metric	Supported	Not supported	Not supported

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protocol ID 1, 2 (IS-IS L1 and L2)	Protocol ID 3 (OSPFv2)	Protocol ID 6 (OSPFv3)
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1152	IGP Flags	Supported Note: D flag only	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1153	IGP Route Tag	Not supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1154	IGP Extended Route Tag	Not supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1155	Prefix Metric	Supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1156	OSPF Forwarding Address	N/A	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1157	Opaque Prefix Attribute	Not supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1158	Prefix SID	Supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1159	Range	Not upported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1161	SID/Label	Not supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1162	SRv6 Locator	Supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1170	Prefix Attribute Flags	Supported Note: X, R, and N flags	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1171	Source Router Identifier	Not supported	Not supported	Not supported
IPv4/IPv6 Prefix NLRI	Prefix Attribute	1174	Source OSPF Router-ID	N/A	Not supported	Not supported

SRv6 SID NLRI and link-state attributes

Table 9: SRv6 SID NLRI and link-state attributes support

NLRI type	Sub-TLV type	Sub-TLV code	Sub-TLV name	Protocol ID 1, 2 (IS-IS L1 and L2)	Protocol ID 6 (OSPFv3)
SRv6 SID NLRI	SRv6 SID Descriptor	518	SRv6 SID Information	Supported	Not supported
SRv6 SID NLRI	SRv6 SID Attribute	1250	SRv6 Endpoint Behavior	Supported	Not supported
SRv6 SID NLRI	SRv6 SID Attribute	1251	SRv6 Peer Node SID	Not supported	Not supported
SRv6 SID NLRI	SRv6 SID Attribute	1252	SRv6 SID Structure	Supported	Not supported

4.25.2 Configuring BGP-LS

About this task

As with other BGP address families, SR Linux supports AFI-SAFI commands to enable and disable support for the link-state address family globally, per group, and per neighbor. These commands enable the advertisement of reachable link-state information. BGP-LS AFI-SAFI commands are supported in the default network instance only.

Procedure

To enable BGP-LS, you must enable **afi-safi link-state** for the BGP instance and for the peer or peer group. You can also define the pre-policy prefix limit and post-policy prefix limit for the peers using the **prefix-limit-received** and **prefix-limit-accepted** commands.

Example: Configure a BGP-LS peer

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                admin-state enable
                afi-safi link-state {
                    admin-state enable
                }
                group peer-group-name {
                    admin-state enable
                    afi-safi link-state {
                        admin-state enable
                    }
                    transport {
                        local-address 10.1.1.1
                    }
                }
                neighbor 10.1.1.2 {
                    peer-group peer-group-name
                    afi-safi link-state {
                        link-state {
                            prefix-limit-received {
                                max-received-routes 1000000
                                warning-threshold-pct 90
                            }
                            prefix-limit-accepted {
                                max-received-routes 100000
                                warning-threshold-pct 90
                            }
                        }
                    }
                }
            }
        }
    }
```

Related topics

Prefix limits for BGP peers

4.25.3 Enabling the import of link-state information into the TE database

Procedure

To enable the import of link-state information from an IGP instance into the TE database for export to BGP-LS routes, use the **te-database-install bgp-ls** command for the IS-IS instance. To specify a unique IGP instance identifier to be included in the BGP-LS NLRI, use the **igp-identifier** parameter. This value is carried in the BGP-LS instance ID and is used as a key to identify an IGP domain. The IGP instance identifier is required when a node is performing the role of a BGP-LS producer.

Example: Import IS-IS link-state information into the TE database

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance name te-database-
install bgp-ls
    network-instance default {
        protocols {
            isis {
                instance name {
                    te-database-install {
                        bgp-ls {
                            igp-identifier 100
                        }
                    }
                }
            }
        }
    }
```

4.25.4 Excluding a level from being imported into the TE database

Procedure

To exclude an IS-IS level from being imported into the TE database for BGP-LS, use the **bgp-ls-exclude true** command.

Example: Exclude level from being imported into the TE database

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance name
    network-instance default {
        protocols {
            isis {
                instance name {
                    level 1 {
                        bgp-ls-exclude true
                    }
                }
            }
        }
    }
```

4.25.5 Configuring BGP-LS route export to reachable-only

Procedure

To restrict the export of BGP-LS routes from the TE database to reachable routes only, use the **traffic-engineering bgp-ls export reachable-ls-only** command.

Example: Configure export of BGP-LS reachable routes only from the TE database

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering bgp-ls export
    network-instance default {
        traffic-engineering {
            bgp-ls {
                export {
                    reachable-ls-only true
                }
            }
        }
    }
```

5 Seamless MPLS with BGP labeled unicast (BGP-LU)

Seamless Multi-Protocol Label Switching (MPLS) is a network architecture that extends MPLS networks to integrate access and aggregation networks into a single MPLS domain, to solve the scaling problems in flat MPLS-based deployments. Seamless MPLS transport partitions the core, aggregation, and access networks into isolated IGP/LDP domains (alternatively, Seamless MPLS is also supported over SR-MPLS TE policy). Seamless MPLS does not define any new protocols or technologies and is based on existing and well-known ones. Seamless MPLS provides end-to-end service-independent transport, separating the service and transport plane. Therefore, it removes the need for service-specific configurations in network transport nodes. Service provisioning is restricted only at the points of the network where it is required.

When BGP is used to distribute a route, it can also distribute an MPLS label that is mapped to that route. The label mapping information is appended to the BGP update message that is used to distribute the route. This is described in RFC 3107, *Carrying Label Information in BGP-4*.

AN routers in a regional area learn the reachability of AN routers in other regional areas through BGP labeled routes redistributed by the local ABRs (RFC 3107).

The label stack contains three labels for packets sent in a VPN service between the access nodes:

- The ANs push a service label to the packets sent in the VPN service. The service label remains unchanged end-to-end between ANs. The service label is popped by the remote AN and is the inner label of the label stack.
- The BGP label is the middle label of the label stack and should be regarded as a transport label. The transport label stack is increased to two labels: BGP and LDP transport labels. The BGP label is pushed by the iLER AN and is swapped at the BGP next hop, which can be one of the two local ABRs. Both ABRs are configured with next-hop-self. The BGP label is also swapped by the remote ABR.
- The iLER AN pushes an LDP transport label to the packets sent to the remote AN to reach the BGP next hop. At the local ABR, the LDP transport label is popped and a new LDP transport label is pushed to reach the BGP next hop (remote ABR). The LDP transport label is swapped in every label switching router (LSR) and popped by the ABR nearest to the remote AN. That ABR pops the LDP transport label, swaps the BGP label, and pushes an LDP transport label to reach the remote eLER AN.

Supported platforms

Seamless MPLS with BGP-LU is supported on the following platforms:

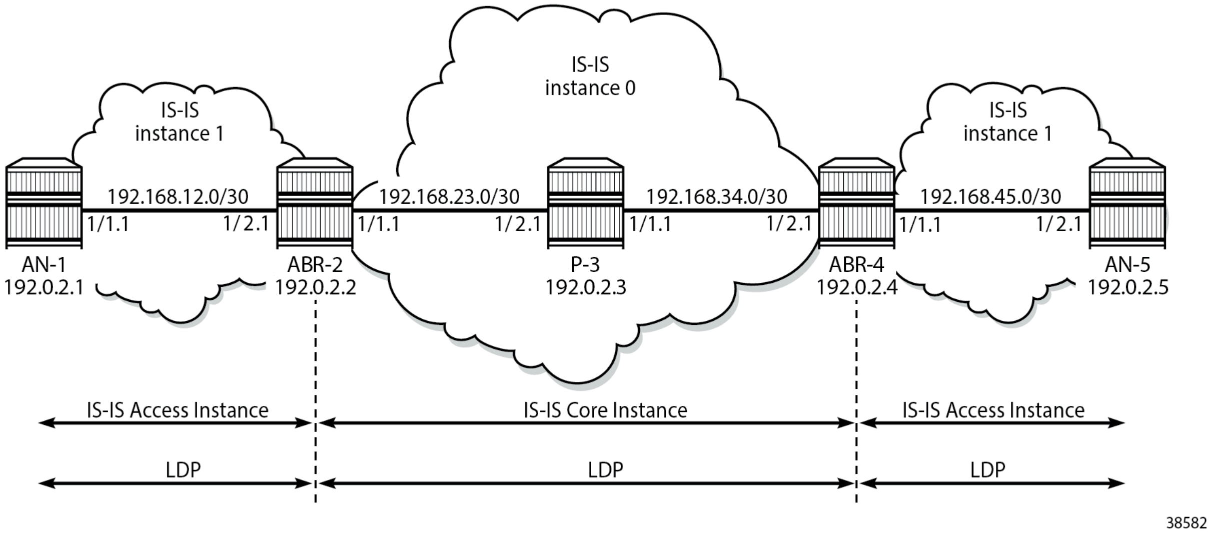
- 7250 IXR
- 7730 SXR

5.1 Seamless MPLS with BGP-LU configuration

The following diagram shows the example topology that is used in this chapter. In the regional areas and in the core area IS-IS L2 capability is used with LDP.

Alternatively, Seamless MPLS with BGP-LU can operate over SR-MPLS TE policy in all areas, or it can also operate over a mix of TE policy and LDP-enabled areas. For example, both regional areas can run LDP, while the core area runs TE policy. However, this example shows LDP configuration only.

Figure 2: Seamless MPLS - IGP/LDP domains



To configure Seamless MPLS, see the following sections:

- Initial configuration for Seamless MPLS
- BGP configuration for Seamless MPLS
- Configuring BGP on ANs toward ABRs

Note: Example configurations are not provided for all nodes in the domain. The provided examples do illustrate the basic configuration required to enable Seamless MPLS, which can be repurposed for other nodes as required.

5.2 Initial configuration for Seamless MPLS

The following sections describe the initial configurations required on all nodes to enable Seamless MPLS.

- Configuring interfaces
- Configuring IS-IS
- Configuring MPLS label blocks
- Configuring LDP

5.2.1 Configuring interfaces

Procedure

Configure the required routing interfaces and add them to the network-instance.

Example: Configure interfaces (ABR-2)

The following example output shows the interface configuration on ABR-2.

```
# on ABR-2:
    interface ethernet-1/2 {
        description ABR2-AN1
        admin-state enable
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.12.2/30 {
                }
            }
        }
    }
    interface ethernet-1/1 {
        description ABR2-P3
        admin-state enable
        subinterface 1 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.168.23.1/30 {
                }
            }
        }
    }
    interface system0 {
        admin-state enable
        subinterface 0 {
            admin-state enable
            ipv4 {
                admin-state enable
                address 192.0.2.2/32 {
                }
            }
        }
    }
```

Example: Add interfaces to the network instance

```
# on ABR-2:
    network-instance default {
        interface ABR2-P3 {
            interface-ref {
                interface ethernet-1/1
                subinterface 1
            }
        }
        interface ABR2-AN1 {
            interface-ref {
                interface ethernet-1/2
                subinterface 1
            }
        }
        interface system0.0 {
        }
```

5.2.2 Configuring IS-IS

Procedure

Configure IS-IS on each of the nodes.

The core area and regional areas run isolated IS-IS instances. ABRs run two IS-IS instances: instance 0 belongs to the core and instance 1 belongs to the access network.

Example: Configure IS-IS on the core instance

On the core instance, all ABRs and Ps require level 2 (L2) capability, as shown in the following example.

```
# on ABR-2:
    network-instance default {
        protocols {
            isis {
                instance ISIS-0 {
                    admin-state enable
                    instance-id 0
                    level-capability L2
                    iid-tlv true
                    net [
                        49.0000.0000.0000.0002.00
                    ]
                    ipv4-unicast {
                        admin-state enable
                    }
                    interface ethernet-1/1.1 {
                        circuit-type point-to-point
                        ipv4-unicast {
                            admin-state enable
                        }
                        level 2 {
                        }
                    }
                    interface system0.0 {
                        admin-state enable
                        passive true
                        ipv4-unicast {
                            admin-state enable
                        }
                        level 2 {
                        }
                    }
                }
```

Example: Configure IS-IS on the access instance

On the access instance, all ABRs and ANs also require L2 capability, as shown in the following example.

```
# on ABR-2:
    network-instance default {
        protocols {
            isis {
                instance ISIS-1 {
                    admin-state enable
                    instance-id 1
                    level-capability L2
                    iid-tlv true
                    net [
                        49.0001.0000.0000.0002.00
                    ]
                    interface ethernet-1/2.1 {
                        circuit-type point-to-point
                        ipv4-unicast {
                            admin-state enable
                        }
                        level 2 {
                        }
                    }
                    interface system0.0 {
                        admin-state enable
                        passive true
                        ipv4-unicast {
                            admin-state enable
                        }
                        level 2 {
                        }
                    }
                }
            }
        }
    }
```

5.2.3 Configuring MPLS label blocks

Procedure

Configure label blocks for LDP and for BGP-LU labels.

Example: Configure label blocks for LDP and BGP-LU

```
--{ + candidate shared default }--[  ]--
# /info with-context system mpls label-ranges
    system {
        mpls {
            label-ranges {
                dynamic D1 {
                    start-label 200
                    end-label 299
                }
                dynamic bgp-lu-block {
                    start-label 12001
                    end-label 13000
                }
            }
        }
    }
```

5.2.4 Configuring LDP

Procedure

Enable Link LDP on all router interfaces on all nodes.

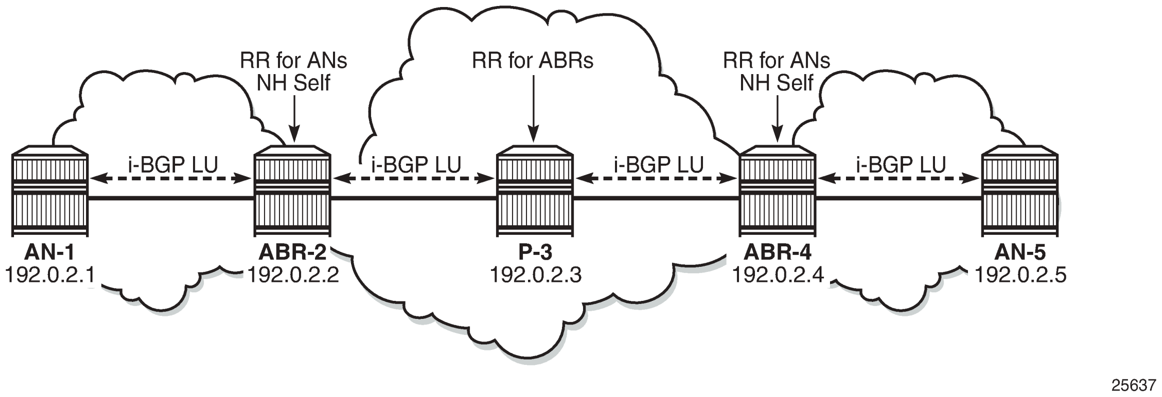
Example: Configure Link LDP (ABR-2)

```
# on ABR-2:
    network-instance default {
        protocols {
            ldp {
                admin-state enable
                dynamic-label-block D1
                discovery {
                    interfaces {
                        interface ethernet-1/1.1 {
                            ipv4 {
                                admin-state enable
                            }
                        }
                        interface ethernet-1/2.1 {
                            ipv4 {
                                admin-state enable
                            }
                        }
                    }
                }
            }
        }
    }
```

5.3 BGP configuration for Seamless MPLS

BGP is configured on all ABRs and all ANs. P-3 acts as a core Route Reflector (RR). To allow for separation of core/access IGP domains, the ABRs become RRs inline and implement next-hop-self on labeled IPv4 BGP prefixes. The following diagram shows the exchange of iBGP Labeled Unicast (LU) routes.

Figure 3: Seamless MPLS – BGP-LU



The following sections describe the BGP configurations required on all nodes to enable Seamless MPLS.

- Configuring BGP on ABRs
- Configuring BGP on the core RR
- Configuring BGP on ANs toward ABRs

5.3.1 Configuring BGP on ABRs

Procedure

Configure two BGP groups on the ABRs: one group toward the core RR and another group toward the AN. Enable **advertise-inactive** on the BGP group toward the core.

The /32 system IP addresses, learned in labeled BGP, are also learned in IS-IS. Because IS-IS has a lower preference compared to iBGP, the IS-IS routes are installed in the routing table. BGP default behavior only advertises those prefixes that were elected by RTM and used.

Note: While the BGP examples show **ipv4-labeled-unicast** configurations, **ipv6-labeled-unicast** is also supported.

Example: Configure BGP core on ABRs

```
# on ABR-2:
    network-instance default
        protocols {
            bgp {
                admin-state enable
                autonomous-system 64496
                router-id 192.0.2.2
                best-path-selection {
                    advertise-inactive true
                }
                bgp-label {
                    labeled-unicast {
                        dynamic-label-block bgp-lu-block
                    }
                }
                afi-safi ipv4-labeled-unicast {
                    admin-state enable
                    ipv4-labeled-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                route-resolution {
                                    ignore-default-routes true
                                }
                                tunnel-resolution {
                                    allowed-tunnel-types [
                                        ldp
                                    ]
                                }
                            }
                        }
                    }
                }
                group AN {
                    admin-state enable
                    peer-as 64496
                    }
                group core {
                    admin-state enable
                    peer-as 64496
                    route-reflector {
                        cluster-id 10.2.2.2
                        client true
                    }
                }
                neighbor 192.0.2.1 {
                    description AN1
                    next-hop-self true
                    peer-group AN
                    }
                neighbor 192.0.2.3 {
                    description coreRR_P3
                    next-hop-self true
                    peer-group core
                }
            }
```

Note:

- To enable ECMP, set **afi-safi [ipv4-labeled-unicast | ipv6-labeled-unicast] multipath maximum-paths** to a value greater than 1. The value of **maximum-paths** sets the maximum number of ECMP forwarding paths, including the best path, for a BGP prefix. The additional non-best-path labeled RIB-INs are added to the ECMP NHG.
- Under **tunnel-resolution allowed-tunnel-types**, you can specify **sr-isis** to enable the use of SR-ISIS tunnels for next-hop resolution of BGP-LU traffic as an alternative to LDP, if SR-ISIS is configured in your domain.

5.3.2 Configuring BGP on the core RR

Example: Configure BGP on the core RR

```
# on P-3:
    network-instance default
        protocols {
            bgp {
                admin-state enable
                autonomous-system 64496
                router-id 192.0.2.3
                best-path-selection {
                    advertise-inactive true
                }
                bgp-label {
                    labeled-unicast {
                        dynamic-label-block bgp-lu-block
                    }
                }
                afi-safi ipv4-labeled-unicast {
                    admin-state enable
                    ipv4-labeled-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                tunnel-resolution {
                                    allowed-tunnel-types [
                                        ldp
                                    ]
                                }
                            }
                        }
                    }
                }
                group core {
                    admin-state enable
                    peer-as 64496
                    afi-safi ipv4-labeled-unicast {
                        admin-state enable
                    }
                    route-reflector {
                        cluster-id 10.3.3.3
                    }
                }
                neighbor 192.0.2.2 {
                    description ABR-2
                    peer-group core
                    route-reflector {
                        cluster-id 10.3.3.3
                        client true
                    }
                }
                neighbor 192.0.2.4 {
                    description ABR-4
                    peer-group core
                    route-reflector {
                        cluster-id 10.3.3.3
                        client true
                    }
                }
            }
```

5.3.3 Configuring BGP on ANs toward ABRs

Example: Configure BGP on AN-1 toward ABR

Configuring **afi-safi ipv4-labeled-unicast** indicates that all advertised IPv4 prefixes are sent to the remote BGP peer as an RFC 3107 formatted label. The **next-hop-self** command only applies to labeled IPv4 prefixes.

```
# on AN-1:
    network-instance default
        protocols {
            bgp {
                admin-state enable
                autonomous-system 64496
                router-id 192.0.2.1
                best-path-selection {
                    advertise-inactive true
                }
                bgp-label {
                    labeled-unicast {
                        dynamic-label-block bgp-lu-block
                    }
                }
                afi-safi ipv4-labeled-unicast {
                    admin-state enable
                    ipv4-labeled-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                route-resolution {
                                    ignore-default-routes true
                                }
                                tunnel-resolution {
                                    allowed-tunnel-types [
                                        ldp
                                    ]
                                }
                            }
                        }
                    }
                }
                group ABRs {
                    admin-state enable
                    peer-as 64496
                    afi-safi ipv4-labeled-unicast {
                        admin-state enable
                    }
                }
                neighbor 192.0.2.2 {
                    description ABR2
                    peer-group ABRs
                }
            }
```

You can show the BGP sessions with the **show network-instance default protocols bgp neighbor** command.

5.4 Export policy configuration for Seamless MPLS

A policy is required on the ANs to advertise the system IP address in labeled BGP toward the ABRs. The same policy is required on the ABRs to advertise their system IP address in labeled BGP toward the core and the AN.

5.4.1 Configuring export policies on ANs and ABRs

Example: Configure a policy on ANs and ABRs

```
# on AN-1 and ABR-2:
    routing-policy {
        prefix-set local-loopback {
            prefix 192.0.2.1/32 mask-length-range exact {
            }
        }
        policy export-system {
            statement 10 {
                match {
                    prefix-set local-loopback
                    protocol local
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

Example: Apply the policy on AN-1

You can apply the export policy to BGP in one of the following **bgp** contexts:

- **protocols bgp**
- **protocols bgp neighbor**
- **protocols bgp group**

Or to apply the policy to BGP-LU only, use one of the following contexts:

- **protocols bgp afi-safi ipv4-labeled-unicast**
- **protocols bgp neighbor afi-safi ipv4-labeled-unicast**
- **protocols bgp group afi-safi ipv4-labeled-unicast**

In this example, the export policy is applied to BGP-LU in the group **ABRs** on AN-1, as follows:

```
# on AN-1:
    network-instance default {
        protocols {
            bgp {
                group ABRs {
                    afi-safi ipv4-labeled-unicast {
                        export-policy export-system
                    }
                }
            }
        }
    }
```

Example: Apply the policy on ABR-2

The same export policy is applied in the group **core** on ABR-2, as follows:

```
# on ABR-2:
    network-instance default {
        protocols {
            bgp {
                group core {
                    afi-safi ipv4-labeled-unicast {
                        export-policy export-system
                    }
                }
            }
        }
    }
```

A similar export policy is required to export prefix 192.0.2.5 from AN-5 to ABR-4 and from ABR-4 to the RR in the core network, P-3.

Use the **show network-instance default route-table** command to display the route table. The prefix of the remote AN should be added to the routing table in AN-1.

5.5 BGP-LU selective install

In BGP-LU networks, the number of BGP-LU routes that are distributed in the control plane can exceed the capacity of the FIB and label forwarding information base (LFIB) of small access routers. One solution to this issue is to apply import policies on the access routers to limit the number of BGP-LU routes accepted in the RIB-IN, but this is labor-intensive and prone to errors. A better solution is to enable selective install of BGP-LU routes in the default network instance, which provides an alternate method of limiting the number of BGP-LU routes in the RIB-IN.

When selective install is configured, BGP-LU routes in the RIB-IN that are received from a BGP peer but not required by any eligible service are handled as follows:

- No BGP-LU route is programmed as a next hop (primary next hop, ECMP next hop, or backup next hop) of any IP route or tunnel.
- No BGP tunnel for the /32 IPv4 or /128 IPv6 prefix is added to the tunnel table.

BGP-LU selective install parameters

The BGP-LU selective install feature supports optional parameters (**program-route** and **program-label-swap**) that allow you to alter the default handling of BGP-LU selective install routes. These parameters enable the installation of ILM entries and FIB entries for the BGP-LU routes, even when those BGP-LU routes are not required by any service and therefore are not installed as tunnels. The parameters are defined as follows:

- **program-label-swap** — When set to **true**, the system programs a label swap entry even when the route is not installed as a tunnel.
- **program-route** — When set to **true**, the system programs an IP FIB entry even when the BGP-LU route is not installed as a tunnel. To set **program-route** to **true**, **program-label-swap** must also be set to **true**.

5.5.1 Configuring BGP-LU selective install

Procedure

To enable BGP-LU selective install, use the **network-instance protocols bgp bgp-label labeled-unicast selective-labeled-unicast-install** command.

Example: Configure BGP-LU selective install

The following example enables BGP-LU selective install. It also enables programming of label swap entries (**program-label-swap true**) and FIB entries (**program-route true**) for the BGP-LU routes, even when those BGP-LU routes are not installed as tunnels.

```
--{  candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp bgp-label labeled-unicast
 selective-labeled-unicast-install
    network-instance default {
        protocols {
            bgp {
                bgp-label {
                    labeled-unicast {
                        selective-labeled-unicast-install {
                            program-label-swap true
                            program-route true
                        }
                    }
                }
            }
        }
    }
```

5.6 Advertisement of BGP-LUv4 routes with IPv6 next hops

In networks where the routers are interconnected by IPv6-only links, SR Linux routers can advertise and receive BGP routes for IPv4 labeled-unicast destinations that are reachable through IPv6 next hops.

Advertising and receiving IPv4 labeled-unicast routes with IPv6 next hops is useful in networks or regions with IPv6-only interfaces.

This feature requires the extended next hop encoding BGP capability which is described in RFC 5549, *Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop*. BGP capabilities are advertised between peers.

By default, IPv4 labeled-unicast routes are advertised with IPv4 next hops. However, on IPv6-only TCP transport sessions, IPv4 labeled-unicast routes can be advertised with IPv6 next hops if the **advertise-ipv6-next-hops** command applies to the session.

To receive IPv4 labeled-unicast routes with IPv6 next-hop addresses, the **receive-ipv6-next-hops** command must be applied to the session. This advertises the RFC 5549 capability to the peer for the IPv4 labeled-unicast address family.

When the BGP session is established, the BGP peers advertise the capability to each other. The Extended Next Hop encoding capability is both a local and a remote capability, as in the following output examples between BGP peers 2001:db8::12:1 and 2001:db8::12:2:

```
# /show  network-instance default protocols bgp neighbor 2001:db8::12:2 detail | grep Cap
Cap Sent:  ROUTE_REFRESH EXT_NH_ENCODING 4-OCTET_ASN MP_BGP GRACEFUL_RESTART
Cap Recv:  ROUTE_REFRESH EXT_NH_ENCODING 4-OCTET_ASN MP_BGP GRACEFUL_RESTART
```

```
# /info with-context from state flat detail network-instance default protocols bgp neighbor
 2001:db8::12:2 | grep EXT
/ network-instance default protocols bgp neighbor 2001:db8::12:2 advertised-capabilities
 [ ROUTE_REFRESH EXT_NH_ENCODING 4-OCTET_ASN MP_BGP GRACEFUL_RESTART ]
/ network-instance default protocols bgp neighbor 2001:db8::12:2 received-capabilities [ ROUTE_
REFRESH EXT_NH_ENCODING 4-OCTET_ASN MP_BGP GRACEFUL_RESTART ]
```

When **next-hop-self** applies to the BGP session and the neighbor address is IPv6, an IPv4 labeled-unicast route that is advertised or re-advertised uses the IPv6 local address used for peering as the next hop.

5.6.1 Enabling IPv6 next-hops on BGP-LUv4 routes

Procedure

To advertise IPv4 labeled-unicast routes with IPv6 next-hops on IPv6-only TCP transport sessions, use the **advertise-ipv6-next-hops** command. To receive IPv4 labeled-unicast routes with IPv6 next-hop addresses, use the **receive-ipv6-next-hops** command. These commands can be applied to the **ipv4-labeled-unicast** family under the **bgp**, **bgp group**, or **bgp neighbor** contexts.

Example: Enable IPv6 next-hops on BGP-LUv4 routes

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-labeled-unicast
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-labeled-unicast {
                    ipv4-labeled-unicast {
                        advertise-ipv6-next-hops true
                        receive-ipv6-next-hops true
                    }
                }
            }
        }
    }
```

6 Segment routing with BGP-LU prefix SID

Segment routing (SR) allows a router to source route a packet by prepending an SR header containing an ordered list of SIDs. A SID can have a local impact to one particular node or it can have a global impact within the SR domain. With SR-MPLS, each SID is an MPLS label and the complete SID list is a stack of labels in the MPLS header.

Traditionally, the association of the SID with an IP prefix is propagated by an IGP routing protocol. However, in some cases, the SID must be propagated beyond the IGP protocol boundaries. For these cases, you can enable the prefix SID path attribute for BGP-LU to associate an SR-MPLS SID with an IP prefix in a labeled unicast route.

The prefix SID attribute associates a prefix with the advertised SID, representing network-wide instructions to forward packets along the BGP ECMP-aware best path back to the prefix. To advertise the prefix SID attribute in BGP-LU, SR Linux derives the advertised label from the IGP-signaled label index, which has the effect of stitching the BGP segment routing tunnel to the IGP segment routing tunnel. As a result, the segment routing-enabled routers can establish an LSP with a consistent label value end-to-end across IGP domains (alternatively, segment routing with BGP-LU prefix SID is also supported over SR-MPLS TE uncolored policy).

The prefix SID attribute is an optional transitive BGP path attribute with type code 40. This attribute encodes a 32-bit label index into the SRGB space and can provide details about the SRGB space of the originating router. The encoding of this BGP path attribute and its semantics are further described in *draft-ietf-idr-bgp-prefix-sid*.

SR Linux attaches a meaning to a prefix SID attribute only when it is attached to routes belonging to the labeled unicast IPv4 and labeled unicast IPv6 address families. When attached to routes of unsupported address families, the prefix SID attribute is ignored but still propagated, as with any other optional transitive attribute.

Supported platforms

Segment routing with BGP-LU prefix SID is supported on the following platforms:

- 7250 IXR
- 7730 SXR

Label assignment and conflicts

A unique label index value is assigned to each unique IPv4 or IPv6 prefix that is advertised with a BGP prefix SID. If label index N1 is assigned to a BGP-advertised prefix P1, and N1 plus the SRGB start label creates a label value that conflicts with another SR programmed entry in the label forwarding information base (LFIB), the conflict situation is addressed according to the following rules:

- If the conflict is with another BGP route for prefix P2 that was advertised with a prefix SID attribute, all the conflicting BGP routes for P1 and P2 are advertised with a normal BGP-LU label from the dynamic label range.
- If the conflict is with an IGP route and BGP is attempting to redistribute that IGP route as a label-ipv4 or label-ipv6 route with a route table import policy action that includes the **bgp label-allocation prefix-sid reuse-igp true** option, this is not considered a conflict and BGP uses the IGP-signaled label index

to derive its advertised label. This has the effect of stitching the BGP segment routing tunnel to the IGP segment routing tunnel.

Any /32 label-ipv4 or /128 label-ipv6 BGP routes containing a prefix SID attribute are resolvable and used in the same way as /32 label-ipv4 or /128 label-ipv6 routes without a prefix SID attribute. These routes are installed in the route table and tunnel table. These routes can have ECMP next hops and can be used as BGP-LU transport tunnels.

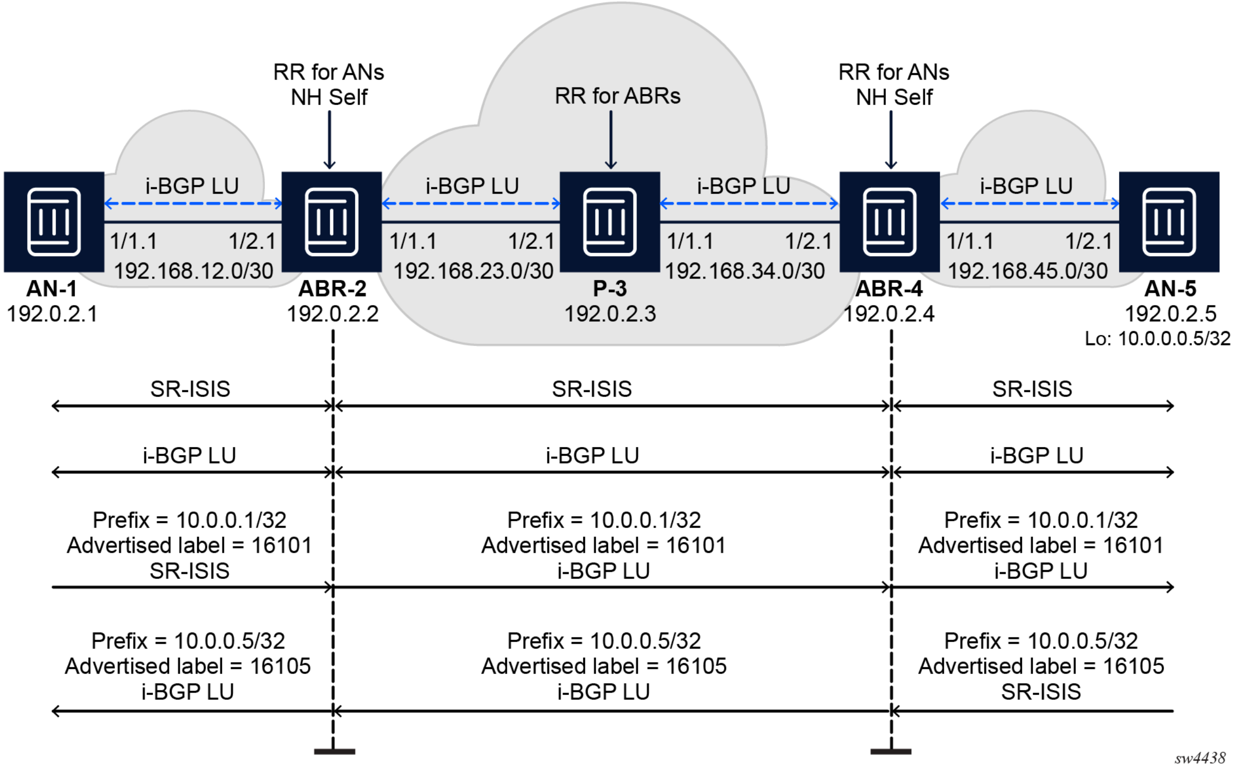
Note: Receiving a /32 label-ipv4 or /128 label-ipv6 route with a prefix SID attribute does not create a tunnel in the segment routing database; it only creates a label swap entry when the route is readvertised with a new next hop. This means that the first SID in any SID list of an SR policy should not be based on a BGP prefix SID because the data path would not be programmed correctly. However, the BGP prefix SID can be used as a non-first SID in any SR policy.

6.1 Segment routing with BGP-LU prefix SID topology example

The following figure shows the example topology referenced in the subsequent procedure, which builds on the BGP-LU configurations shown in Seamless MPLS with BGP-LU configuration. The BGP-LU routers along the shortest path advertise consistent labels for prefix 10.0.0.1/32 on AN-1 and for prefix 10.0.0.5/32 on AN-5, providing consistent label usage end-to-end across the multiple IS-IS domains.

Note: Unlike the seamless MPLS example, the following configurations assume that SR-ISIS is used in the IGP domain and not LDP. Alternatively, segment routing with BGP-LU prefix SID is also supported over SR-MPLS TE uncolored policy.

Figure 4: Segment routing with BGP-LU prefix SID topology - SR-ISIS/iBGP-LU domains



In this example, all nodes at minimum have **protocols bgp segment-routing-mpls** enabled, and are configured with an SR-MPLS SRGB of 16000-16999. The nodes operate as follows:

- AN-1 advertises its loopback address (10.0.0.1/32) to its neighbor ABR-2 using a statically configured prefix SID label (16101) in SR-ISIS.

  Note: On AN-1, BGP-LU is not advertising the AN-1 prefix SID to ABR-2. Instead, this advertisement is handled by SR-ISIS.

- ABR-2 uses a route table import policy to import the AN-1 prefix SID into its BGP RIB, and an export routing policy to export the AN-1 prefix SID to its neighbors. The route table import policy includes the **bgp label-allocation prefix-sid reuse-igp** action. With this option enabled, ABR-2 derives a BGP-LU label for the AN-1 prefix SID by reusing both the SR-MPLS SRGB and the SR-ISIS label index for AN-1. As a result, BGP-LU advertises the same prefix SID label (16101) toward P-3 as it received from AN-1. These policies result in stitching the BGP-LU path (from P-3 to ABR-2) to the SR-ISIS tunnel (ABR-2 to AN-1).
- To propagate prefix label 16101 from ABR-2 to ABR-4, P-3 uses only BGP-LU, advertising routes to its iBGP neighbors. On P-3, SR-ISIS carries no prefix SID for AN-1. As a result, no import or export policy is required on P-3.
- ABR-4 also uses BGP-LU to advertise the AN-1 prefix SID label 16101 to AN-5.
- After the AN-1 prefix SID is propagated end-to-end, when AN-5 wants to reach AN-1, it can simply apply label 16101 to the packets.

- The basic steps outlined above also apply for AN-5 advertising its loopback address (10.0.0.5/32) and prefix SID label (16105) in the opposite direction toward AN-1. The result being that, when AN-1 wants to reach AN-5, it can apply label 16105 to the packets.

6.2 Configuring segment routing with BGP-LU prefix SID

Prerequisites

- Configure SR-MPLS in your network.
- Configure BGP-LU.

About this task

To configure BGP-LU with prefix SID for segment routing, perform the following steps.

Procedure

Step 1. On the access, ABR, and P nodes, enable segment routing for BGP.

Step 2. On the access nodes, associate a static segment routing local prefix SID label with the loopback interface.

Note: No dedicated label block for BGP-LU prefix SID labels is required.

Step 3. On the ABR nodes, configure a route table import policy to import the prefix SIDs with the **reuse-igp true** option and apply the route table import policy to the BGP RIB.

Note: When reusing the label index from the IGP, BGP-LU prefix SID label values are drawn from the SRGB defined as part of the SR-MPLS configuration (and not from the label block defined in the BGP-LU configuration).

Step 4. On the ABR nodes, configure an export routing policy to export prefix SIDs from the BGP RIB, and apply the export policy to the applicable BGP peer groups or neighbors.

Step 5. As required, block propagation of prefix SIDs outside of the segment routing domain where they apply.

Example: Segment routing with BGP-LU prefix SID configurations

The example configurations that follow apply for AN-1 and ABR-2 to advertise a prefix SID for 10.0.0.1/32. Similar configurations (not shown) are required for AN-5 and ABR-4 to advertise a prefix SID for 10.0.0.5/32.

Example: Enable segment routing for BGP on all nodes

The following example enables segment routing using the **bgp segment-routing-mpls** command.

```
--{ candidate shared default }--[  ]--
All-nodes# info with-context network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                segment-routing-mpls {
                    admin-state enable
                }
            }
        }
    }
```

Example: Associate local prefix SID label with the loopback interface on the access node (AN-1 shown)

The following example associates local prefix SID label 101 with the loopback interface lo0.0 on AN-1.

```
--{ candidate shared default }--[  ]--
AN-1# info with-context interface lo0
    interface lo0 {
        admin-state enable
        subinterface 0 {
            ipv4 {
                address 10.0.0.1/32 {
                }
            }
        }
    }
--{ candidate shared default }--[  ]--
AN-1# info with-context network-instance default interface lo0.0
    network-instance default {
        interface lo0.0 {
            interface-ref {
                interface lo0
                subinterface 0
            }
        }
    }
--{  candidate shared default }--[  ]--
AN-1# info with-context network-instance default segment-routing
    network-instance default {
        segment-routing {
            mpls {
                local-prefix-sid 1 {
                    interface lo0.0
                    ipv4-label-index 101
                }
            }
        }
    }
```

Example: Configure the ABR route table import policy (ABR-2 shown)

The following example defines a route table import policy for importing prefix 10.0.0.1/32 into the BGP RIB on ABR-2 using the **routing-policy** command. It also sets the **bgp label-allocation prefix-sid reuse-igp** action, which directs BGP-LU to use the IGP-signaled label-index to derive its advertised label. This has the effect of stitching the BGP segment routing tunnel to the IGP segment routing tunnel.

The example also applies the route table import policy to the BGP RIB on ABR-2 using the **bgp rib-management** command to populate the local BGP-LU table with the segment routing label for prefix 10.0.0.1/32.

```
--{ candidate shared default }--[  ]--
ABR-2# info with-context routing-policy
    routing-policy {
        policy import-prefix-sid-2 {
            statement 10 {
                match {
                    prefix-set 10.0.0.1/32
                }
                action {
                    policy-result accept
                    bgp {
                        label-allocation {
                            prefix-sid {
                                reuse-igp true
                            }
                        }
                    }
                }
            }
        }
    }
--{ candidate shared default }--[  ]--
ABR-2# info with-context network-instance default protocols bgp rib-management
    network-instance default {
        protocols {
            bgp {
                rib-management {
                    table ipv4-labeled-unicast {
                        route-table-import import-prefix-sid-2
                    }
                }
            }
        }
    }
```

Example: Configure the export routing policy for the ABRs (ABR-2 shown)

The following example defines a policy for ABR-2 to export prefix 10.0.0.1/32 to BGP neighbors using the **routing-policy** command. It applies the policy on ABR-2 using the **bgp neighbor** command. It also configures BGP group iBGP with BGP-LU enabled, and applies the group to the peer.

The policy advertises the segment routing label for prefix 10.0.0.1/32 from the ABR-2 BGP-LU table to neighbor P-3 (192.0.2.3).

```
--{ candidate shared default }--[  ]--
ABR-2# info with-context routing-policy
    routing-policy {
        policy export-prefix-sid-1 {
            statement 10 {
                match {
                    prefix-set 10.0.0.1/32
                }
                action {
                    policy-result accept
                }
            }
        }
    }
--{ candidate shared default }--[  ]--
ABR-2# info with-context network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                group iBGP {
                    afi-safi ipv4-labeled-unicast {
                        admin-state enable
                    }
                }
                neighbor 192.0.2.3 {
                    peer-as 64496
                    peer-group ABRs
                    export-policy [
                        export-prefix-sid-1
                    ]
                }
            }
        }
    }
```

Example: Block propagation of prefix SIDs

By default, the prefix SID attribute propagates without restriction. To prevent the prefix SID from propagating outside the segment routing domain where it is applicable, use the **network-instance protocols bgp [group | neighbor] optional-attributes block-prefix-sid** command. The command removes the prefix SID attribute from all routes sent to and received from the iBGP and eBGP peers included in the scope of the command.

The following example blocks the propagation of prefix SIDs to and from BGP neighbor 192.0.2.10 (not shown in the topology diagram).

```
--{ + candidate shared default }--[  ]--
Any-node# info with-context network-instance default protocols bgp neighbor 192.0.2.10
    network-instance default {
        protocols {
            bgp {
                neighbor 192.0.2.10 {
                    optional-attributes {
                        block-prefix-sid true
                    }
                }
            }
        }
    }
```

Example: Display the prefix SID path attribute

To display the path attributes of the advertised BGP-LU routes, including the `prefix-sid` attribute, use the **info from state bgp-rib attr-sets attr-set** command. The following example shows the BGP-LU path attributes advertised from ABR-2 to P-3. In this example, the **info from state** command is entered from the **network-instance** context.

```
--{ candidate shared default }--[ network-instance default ]--
P-3# info with-context from state bgp-rib attr-sets attr-set <attr-id>
    origin igp
    atomic-aggregate false
    next-hop 192.0.2.2
    med 0
    local-pref 100
    prefix-sid {
        tlv label-index {
            label-index {
                label-index 101
            }
        }
        tlv srgb-originator {
            srgb-originator {
                srgb [
                    16000:1000
                ]
            }
        }
    }
```

6.3 LSP statistics for BGP-LU

Note: This feature is currently supported exclusively on the 7730 SXR platform.

When implementing traffic engineering for BGP traffic over MPLS or through a segment routing tunnel, it is essential to have visibility into the traffic volume traversing the BGP labeled unicast tunnel. LSP statistics are often used in SDN applications, such as via Path Computation Engine (PCE) to perform balanced load balancing of traffic across various paths throughout the network.

6.3.1 Enabling statistics collection for BGP-LU routes

Procedure

SR Linux supports the statistics collection of BGP-LU routes and tunnels.

Example: Enabling ingress statistics collection for BGP-LU routes

When a BGP-LU route (**ipv4-labeled-unicast** or **ipv6-labeled-unicast**) is received, it is matched against the BGP peer import policy, which may be applied at the instance, group, or neighbor level.

If the matched policy rule includes the action **routing-policy.policy.statement.action.bgp.statistics.labeled-unicast.ingress** set to `true`, ingress statistics are collected. These statistics are then maintained and displayed for the ILM entry created when the route is re-advertised to other BGP peers using next-hop-self.

Note: Regardless of the policy action, if a BGP-LU route is not re-advertised, no ingress statistics are collected.

In the following configuration example, the policy `srl_ilm_test_policy` includes a statement (`10`) to enable the collection of statistics for incoming MPLS labeled unicast traffic in BGP. Here, the action is applied by the **statement** parameter.

```
{ + candidate shared default }--
# info with-context routing-policy policy srl_ilm_test_policy
    routing-policy {
        policy srl_ilm_test_policy {
            statement 10 {
                action {
                    bgp {
                        statistics {
                            labeled-unicast {
                                ingress true
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Enabling egress statistics collection for BGP-LU routes

When a BGP-LU route (**ipv4-labeled-unicast** or **ipv6-labeled-unicast**) is received, it is matched against the BGP peer import policy, which may be applied at the instance, group, or neighbor level.

If the matched policy rule includes the action **routing-policy.policy.statement.action.bgp.statistics.labeled-unicast.egress** set to `true`, egress statistics are collected. These statistics are then collected and maintained for the NHG entry that results from installing this route as a tunnel in TTM.

Note: Regardless of the policy action, egress statistics are not collected for BGP-LU routes that are not eligible for installation into the TTM, such as non-host prefixes.

In the following configuration example, the policy (`srl_nhlfe_entries`) includes a statement (12) to enable the collection of statistics for outgoing MPLS labeled unicast traffic in BGP. Here, the action is applied by the **statement** parameter.

```
{ + candidate shared default }--
# info with-context routing-policy policy srl_nhlfe_entries
    routing-policy {
        policy srl_nhlfe_entries {
            statement 12 {
                action {
                    bgp {
                        statistics {
                            labeled-unicast {
                                egress true
                            }
                        }
                    }
                }
            }
        }
    }
```

6.4 Forwarding entries

To monitor the traffic volume within a BGP tunnel at each node, the network operators collect statistics for the NHLFE and ILM forwarding entries based on routing policy actions.

Swap ILM Entries (Single IPv4/IPv6 Prefix)

- Programmed when BGP receives a labeled-unicast route and re-advertises it with next-hop-self using LPP (label-per-prefix) allocation.
- When an MPLS packet is received and the top label matches one of these ILM entries, the packet is forwarded according to the next-hop-group (NHG) bound to the swap ILM entry; the NHG has one or more next-hops, one for each ECMP next-hop (NHLFE) of the ILM entry.
- The datapath obtains stats for these entries by counting packets that match each ILM table entry.

BGP-LU Tunnel Table Entries

- Programmed when BGP receives a labeled-unicast route for a host IPv4/IPv6 prefix and adds to TTM, indexed by the NLRI prefix.

- Packets are forwarded according to the NHG bound to the tunnel table entry. An NHG can contain multiple NHLFEs, each representing a specific ECMP path.
- The datapath obtains stats by counting packets matching the NHG used by each tunnel.

7 IS-IS

Note: IS-IS is supported on 7215 IXS, 7220 IXR, 7250 IXR, and 7730 SXR platforms.

Intermediate System to Intermediate System (IS-IS) is a link-state IGP that uses the Shortest Path First (SPF) algorithm to determine routes. Routing decisions are made using the link-state information. IS-IS evaluates topology changes and, if necessary, performs SPF recalculations.

Entities within IS-IS include networks, intermediate systems, and end systems. In IS-IS, a network is an Autonomous System (AS), or routing domain, with end systems and intermediate systems. A router is an intermediate system that sends, receives, and forwards Protocol Data Units (PDUs). End systems are network devices that send and receive PDUs.

End system and intermediate system protocols allow routers and nodes to identify each other. IS-IS sends out link-state updates periodically throughout the network, so each router can maintain current network topology information.

IS-IS supports large ASs by using a two-level hierarchy. A large AS can be administratively divided into smaller, more manageable areas. A system logically belongs to one area. Level 1 routing is performed within an area. Level 2 routing is performed between areas. You can configure routers as Level 1, Level 2, or both Level 1 and 2.

The following summarizes SR Linux support for IS-IS:

- Level 1, Level 2, and Level 1/2 IS types
- Configurable Network Entity Title (NET) per IS-IS instance
- support for IPv4/v6 routing
- ECMP with up to 64 next hops per destination
- IS-IS export policies (redistribution of other types of routes into IS-IS)
- authentication of LSP, CSNP, PSNP, and hello PDUs, using an authentication key or keychain specified as follows:
  - per instance or per level for all PDU types
  - per interface or per interface and level for Hello PDUs
- authentication keychains with a single key per named keychain
- Purge Originator ID TLV (RFC 6232)
- options to ignore and suppress the attached bit
- ability to set the overload bit immediately or after each subsequent restart of the IS-IS manager application and leave it on for a configurable duration each time
- control over the link-state PDU (LSP) MTU size, with range from 490 bytes to 9490 bytes
- configuration control over timers for LSP lifetime, LSP refresh interval, SPF calculation triggers, and LSP generation
- hello padding (strict, loose, and adaptive modes)
- graceful restart, acting in restarting router mode and helper router mode

- Level 1 to Level 2 route summary
- BFD for fast failure detection
- configurable hello timer with multiple per interface and level
- wide metrics (configurable per level)
- configurable route preference for each route type, Level 1-internal, Level 1-external, Level 2-internal and Level 2-external
- detailed statistics for interfaces, adjacencies, and levels
- Multi-instance IS-IS (MI-ISIS), which allows multiple instances of IS-IS to operate on a single circuit
- Multi-Topology MT-ID 0 and MT-ID 2 support, with MT-ID2 reserved for IPv6 network topology
- Multi-Topology Intermediate System to Intermediate System (MT-ISIS) MT0 and MT2 support

The **info detail** command displays default values for an IS-IS instance in SR Linux as shown in the following example:

```
--{ * candidate shared default }--[ network-instance default protocols isis ]--
# info with-context detail
    instance i1 {
        admin-state disable
        level-capability L2
        max-ecmp-paths 1
        poi-tlv false
        attached-bit {
            ignore false
            suppress false
        }
        overload {
            advertise-interlevel false
            advertise-external false
            immediate {
                set-bit false
                max-metric false
            }
            on-boot {
                set-bit false
                max-metric false
            }
        }
        timers {
            lsp-lifetime 1200
            lsp-refresh {
                interval 600
                half-lifetime true
            }
            spf {
                initial-wait 1000
                second-wait 1000
                max-wait 10000
            }
            lsp-generation {
                initial-wait 1000
                second-wait 1000
                max-wait 5000
            }
        }
        transport {
            lsp-mtu-size 1492
        }
        ipv4-unicast {
            admin-state enable
        }
        ipv6-unicast {
            admin-state enable
            multi-topology false
        }
        graceful-restart {
            helper-mode false
        }
        auto-cost {
        }
        authentication {
            csnp-authentication false
            psnp-authentication false
            hello-authentication false
        }
        inter-level-propagation-policies {
            level1-to-level2 {
            }
        }
    }
```

7.1 Basic IS-IS configuration

To configure IS-IS, perform the following tasks:

- Enable an IS-IS instance
- If necessary, modify the level capability on the global IS-IS instance level
- Define area addresses
- Configure IS-IS interfaces

7.1.1 Enabling an IS-IS instance

Procedure

In SR Linux, you can enable an IS-IS instance within a network-instance. The following example enables an IS-IS instance within the default network-instance.

Example

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance default {
        protocols {
            isis {
                instance i1 {
                }
            }
        }
    }
```

7.1.2 Configuring the router level

Procedure

You can configure routers as Level 1, Level 2, or both Level 1 and 2. When IS-IS is enabled, the default level-capability value is Level 2. This means that the router operates with Level 2 routing capabilities. To change the default value in order for the router to operate as a Level 1 router or a Level 1/2 router, you must explicitly modify the level value.

The level-capability value can be configured on the global IS-IS instance level and also on the interface level. The level-capability value determines which level values can be assigned on the router level or on an interface-basis.

In order for the router to operate as a Level 1 only router or as a Level 2 only router, you must explicitly specify the level-number value.

- Specify Level 1 to route only within an area
- Specify Level 2 to route to destinations outside an area, toward other eligible Level 2 routers

Example

The following example configures the level capability for an IS-IS instance to Level 2.

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info with-context isis
 isis {
        instance i1 {
            level-capability L2
        }
    }
```

7.1.3 Configuring the Network Entity Title

Procedure

In SR Linux, you can configure the Network Entity Title (NET) per IS-IS instance. The NET is 8-20 octets long and consists of 3 parts: the area address (1-13 octets), the system ID (6 octets), and the n-selector (1 octet, must be 00)

The area address portion of the NET defines the IS-IS area to which the router belongs. At least one area address should be configured on each router participating in IS-IS.

The area address portion of the NET identifies a point of connection to the network, such as a router interface. The routers in an area manage routing tables about destinations within the area. The NET value is used to identify the IS-IS area to which the router belongs.

The NET value is divided into three parts. Only the Area ID portion is configurable.

1. Area ID — A variable length field between 1 and 13 bytes. This includes the Authority and Format Identifier (AFI) as the most significant byte and the area ID.

2. System ID — A 6-byte system identifier. This value is not configurable. The system ID is derived from the system or router ID.

3. Selector ID — A 1-byte selector identifier that must contain zeros when configuring a NET. This value is not configurable. The selector ID is always 00.

Example

The following example configures a NET for an IS-IS instance. In the example, the Area ID is 49, the System ID is 1921.6800.1002, and the Selector ID is 00.

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info with-context isis
 isis {
        instance i1 {
            net 49.0001.1921.6800.1002.00
            }
        }
    }
```

7.1.4 Configuring global parameters

Procedure

You can configure the commands and parameters on the global IS-IS instance level.

Commands and parameters configured on the global IS-IS instance level are inherited by the interface levels. Parameters specified in the interface and interface-level configurations take precedence over global configurations.

Example

The following example shows the command usage to configure global-level IS-IS. The LSP PDU authentication setting references a keychain defined at the system level (see Protocol authentication).

```
--{ * candidate shared default }--[ network-instance default protocols ]--
# info with-context isis
    isis {
        instance i1 {
            level-capability L2
            overload {
                on-boot {
                    timeout 90
                }
            }
            authentication {
                lsp-authentication {
                    generate true
                    check-received strict
                    keychain isisglobal
                }
            }
        }
    }
```

7.1.5 Configuring interface parameters

Procedure

There are no interfaces associated with IS-IS by default. An interface belongs to all areas configured on a router. Interfaces cannot belong to separate areas. There are no default interfaces applied to the router IS-IS instance. You must configure at least one IS-IS interface in order for IS-IS to work.

You can configure both the Level 1 parameters and the Level 2 parameters on an interface. The level-capability value determines which level values are used.

Example

The following example configures interface parameters for an IS-IS instance:

```
--{ * candidate shared default }--[ network-instance default protocols isis ]--
# info with-context instance i1
    instance i1 {
        interface ethernet-1/2.1 {
            circuit-type point-to-point
            ipv4-unicast {
                admin-state enable
            }
            level 1 {
                authentication {
                    hello-authentication {
                        generate true
                        check-received strict
                        keychain Hello
                    }
                }
            }
        }
        level 1 {
        }
    }
```

7.1.6 Configuring authentication keys

Procedure

IS-IS supports authentication for PDUs using shared keys, which are changed at regular intervals using keys configured in a keychain. This authentication mechanism is described in Protocol authentication.

In addition to using shared keys, authentication for IS-IS Hello, CSNP, PSNP, and LSP PDUs can be done using directly configured keys. You can specify the key used for authenticating IS-IS PDUs associated with a specific IS-IS instance, received or transmitted on a specific interface, and associated with a specific level.

If a Hello PDU is received or transmitted on a specific interface, it is authenticated using the key configured for that interface. If no key exists for the interface, the key configured for the instance is used. For CSNP, PSNP. and LSP PDUs, authentication is performed using the key configured for the level. If no key exists for the level, the key configured for the instance is used.

To configure a key, you specify the secret key (`auth-password`) and cryptographic algorithm to be used for generating the key.

Example

The following example configures keys for an IS-IS instance, interface, and level.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance default {
        protocols {
            isis {
                instance i1 {
                    authentication {
                        key {
                            crypto-algorithm cleartext
                            auth-password $aes$9G3XrtckZzMg=$In9WuOvKPsTw6ehDX5YLgA==
                        }
                    }
                    interface ethernet-1/1.1 {
                        authentication {
                            key {
                                crypto-algorithm hmac-md5
                                auth-password $aes$97mfUA4Swx6I=$PfF02Mtu0gUXH5LwT/ltqQ==
                            }
                        }
                    }
                    level 1 {
                        authentication {
                            key {
                                crypto-algorithm hmac-sha-256
                                auth-password $aes$9YHCLtkxaLGw=$7HXuQHhR4wPXwifkGekFaQ==
                            }
                        }
                    }
                }
            }
        }
    }
```

7.2 IS-IS graceful restart

IS-IS graceful restart is a mechanism to prevent routing protocol re-convergence during a control plane switchover, reset, or upgrade.

Without graceful restart, when an IS-IS router restarts, its IS-IS neighbors detect that the router has gone down and remove routes that pass through that neighbor, resulting in data loss even if the restarting router is able to maintain its forwarding table state.

Configuring graceful restart can prevent this data loss. To do this, the restarting router relies on neighbor routers, which have also been configured for graceful restart, to maintain forwarding state while the router restarts. These neighbor routers are known as helper routers.

For IS-IS graceful restart, SR Linux can operate in the following modes:

- restarting router mode when its own IS-IS manager application (isis_mgr) restarts, or the application is killed
- helper router mode when it detects that an adjacent IS-IS router has restarted

Restarting router mode

When the router is configured to operate in restarting router node, if the SR Linux IS-IS manager application restarts or is killed, routes that are published by the IS-IS manager (associated with any IS-IS instance), and used for programming the FIB, remain programmed for a duration of time sufficient to allow a warm restart of the IS-IS manager application. The system starts sending periodic IS-IS Hello (IIH) PDUs (with the Restart Request (RR) flag set) on all IS-IS interfaces so that neighbors do not time out their adjacencies.

The helper router on each interface responds with an IIH PDU containing a Restart Acknowledgement (RA). The exchange of IIH PDUs on each interface causes adjacencies to reform. After each adjacency comes up, the helper router on the associated interface transmits a complete set of CSNPs to the restarting router.

Helper router mode

When the router is configured to operate in helper router node, SR Linux advertises TLV 211 in IIH PDUs sent on all IS-IS interfaces, which indicates that the IS-IS instance is able to help any adjacent neighbor that signals a restart.

When the router receives an IIH PDU with the RR flag set from an adjacent neighbor, the router maintains its adjacency with the neighbor, responds with an IIH PDU with the RA flag set, and sends a complete set of CSNP PDUs to the restarting router.

7.2.1 Configuring IS-IS graceful restart

Procedure

You can configure SR Linux to operate as a restarting router (informing adjacent routers when the IS-IS manager application restarts or is killed) and as a helper router (indicating to adjacent routers that it can help those signaling a restart). By default, both restarting router mode and helper router mode are disabled.

Example: Configure restarting router mode

To configure the router to operate in restarting router mode for IS-IS graceful restart, enable **non-stop-forwarding** for IS-IS. For example:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance green protocols isis
    network-instance default {
        protocols {
            isis {
                non-stop-forwarding {
                    admin-state enable
                }
            }
        }
    }
```

Example: Configure helper router mode

The following example enables the router to operate in helper router mode:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance green protocols isis instance i1 graceful-restart
    network-instance default {
        protocols {
            isis {
                instance i1 {
                    graceful-restart {
                        helper-mode true
                        acceptable-duration 120
                    }
                }
            }
        }
    }
```

The **acceptable-duration** parameter sets the amount of time in seconds that SR Linux advertises as the Remaining Time in the Restart TLV with the RA flag set when this router starts to help another router that has entered restart mode. By default, this is 60 seconds.

7.3 Displaying IS-IS information

Procedure

Use the commands shown in this section to display the following information for an IS-IS instance running in a specified network instance:

- interface information
- adjacency information
- IS-IS link state database information

Example: Display IS-IS summary information

To display summary information for an IS-IS instance:

```
# show network-instance green_default protocols isis summary
------------------------------------------------------------------------------------------
Network instance "green_default", isis instance "default" is enable and up
Level Capability : L1L2
Export policy    : None
------------------------------------------------------------------------------------------
System-id : 0050.0500.5005
NET       : [ 49.0001.0050.0500.5005.00 ]
Area-id   : [ 49.0001 ]
------------------------------------------------------------------------------------------
IPv4 routing is enable
IPv6 routing is enable using None
Max ECMP path : 1
------------------------------------------------------------------------------------------
Ldp Synchronization is Disabled
------------------------------------------------------------------------------------------
Overload
Current Status : not in overload
------------------------------------------------------------------------------------------
Metric
Reference bandwidth: NA
L1 metric style: wide
L2 metric style: wide
------------------------------------------------------------------------------------------
Graceful Restart
Helper Mode    : disabled
Current Status : not helping any neighbors
```

```
-----------------------------------------------------------------------------------------
Timers
LSP Lifetime                : 1200
LSP Refresh                 : 600
SPF initial wait            : 1000
SPF second wait             : 1000
SPF max wait                : 10000
LSP generation initial wait : 10
LSP generation second wait  : 1000
LSP generation max wait     : 5000
-----------------------------------------------------------------------------------------
Route Preference
L1 internal : 15
L1 external : 160
L2 internal : 18
L2 external : 165
-----------------------------------------------------------------------------------------
L1->L2 Summary Addresses Not configured
-----------------------------------------------------------------------------------------
Instance Statistics
SPF run           : 29
Last SPF          : 2022-03-23T16:16:16.200Z
Partial SPF run   : 16
Last Partial SPF  : 2022-03-23T16:16:17.200Z
-----------------------------------------------------------------------------------------
PDU Statistics
-----------------------------------------------------------------------------------------
| pdu-name |     received      |    processed      |     dropped       |       sent        |
+==========+===================+===================+===================+===================+
| LSP      | 460               | 457               | 3                 | 528               |
| IIH      | 308               | 281               | 27                | 497               |
| CSNP     | 52                | 51                | 1                 | 116               |
| PSNP     | 30                | 30                | 0                 | 3                 |
-----------------------------------------------------------------------------------------
```

Example: Display IS-IS interface information

To display interface information for an IS-IS instance:

```
# show network-instance green_default protocols isis interface
-----------------------------------------------------------------------------------------
Network Instance : green_default
Instance         : default
-----------------------------------------------------------------------------------------
| Interface Name | Oper State | Level | Circuit id | Circuit type   | Ipv4 Metric L1/L2 | Ipv6 Metric
 L1/L2
==============================================================================================
| ethernet-1/1.1 | up         | L1L2  | 2          | point-to-point | 10/10             | 10/10
| ethernet-1/2.1 | up         | L1L2  | 2          | broadcast      | 10/10             | 10/10
| ethernet-1/3.1 | up         | L1L2  | 3          | broadcast      | 10/10             | 10/10
| ethernet-1/16. | up         | L1L2  | 4          | broadcast      | 10/10             | 10/10
| 1              |            |       |            |                |                   |
| lo0.1          | up         | L1L2  | 5          | broadcast      | 0/0               | 0/0
----------------------------------------------------------------------------------------------
```

Example: Display IS-IS interface detail information

To display detail information for a specific IS-IS interface:

```
# show network-instance green_default protocols isis interface ethernet-1/1.1 detail
---------------------------------------------------------------------------------------
Network Instance   : green_default
Instance           : default
```

```
----------------------------------------------------------------------------------------
Interface-Name                      : ethernet-1/1.1
Status                              : IS-IS is admin enabled, oper up
Circuit                             : id 1 is broadcast and not passive
Hello Authentication Generate       : True
Hello Authentication Check Received : Strict
Hello Padding                       : disable
Csnp Interval                       : 10
Lsp Pacing                          : 100
Ldp Sync State                      : disabled
Ldp Sync Duration                   : 3274
---------------------------------------------------------------------------------------
Level                               : 1
Status                              : enabled
Adjacencies                         : 1
Hello Authentication Generate       : True
Hello Authentication Check Received : Strict
Priority                            : 64
Hello Interval                      : 9
Hello Multiplier                    : 3
Ipv4 Metric                         : 10
Ipv6 Metric                         : 10
----------------------------------------------------------------------------------------
Level                               : 2
Status                              : enabled
Adjacencies                         : 1
Hello Authentication Generate       : True
Hello Authentication Check Received : Strict
Priority                            : 64
Hello Interval                      : 9
Hello Multiplier                    : 3
Ipv4 Metric                         : 10
Ipv6 Metric                         : 10
----------------------------------------------------------------------------------------
```

Example: Display IS-IS adjacency information

To display IS-IS adjacency information:

```
# show network-instance default protocols isis adjacency
--------------------------------------------------------------------------------------
Network-instance    : default
IS-IS instance      : global
--------------------------------------------------------------------------------------
System-Id      Adj-Level  Interface        IPv4-Address  State  Uptime       Rem-Hold
<hostname1>    L1         ethernet-1/1.0   10.0.0.1      Up     0d 00:46:43  19s
<hostname1>    L2         ethernet-1/1.0   10.0.0.1      Up     0d 00:46:43  19s
--------------------------------------------------------------------------------------
Adjacencies: 2
--------------------------------------------------------------------------------------
```

Example: Display IS-IS link state database information

To display information for the IS-IS link state database:

```
# show network-instance green_default protocols isis database
--------------------------------------------------------------------------------------
Network-instance    : green_default
IS-IS instance      : default
--------------------------------------------------------------------------------------
| Level Number | Lsp Id               | Sequence | Checksum | Lifetime | Attributes |
+==============+======================+==========+==========+==========+============+
| 1            | 0010.0100.1001.00-00 | 0x33     | 0x1672   | 1167     | L1 L2      |
| 1            | 0020.0200.2002.00-00 | 0x35     | 0xd562   | 1014     | L1 L2      |
| 1            | 0030.0300.3003.00-00 | 0x38     | 0xf447   | 640      | L1 L2      |
| 1            | 0030.0300.3003.01-00 | 0x2f     | 0x4db6   | 1005     | L1 L2      |
| 1            | 0030.0300.3003.02-00 | 0x2e     | 0xd355   | 709      | L1 L2      |
| 1            | 0040.0400.4004.00-00 | 0x39     | 0x6f2a   | 638      | L1 L2      |
| 1            | 0040.0400.4004.01-00 | 0x2f     | 0xf0ef   | 822      | L1 L2      |
| 1            | 0040.0400.4004.02-00 | 0x2f     | 0xa5f8   | 999      | L1 L2      |
| 1            | 0050.0500.5005.00-00 | 0x38     | 0xfbbb   | 905      | L1 L2      |
| 1            | 0050.0500.5005.01-00 | 0x31     | 0x3937   | 745      | L1 L2      |
| 1            | 0050.0500.5005.02-00 | 0x2f     | 0xd19    | 657      | L1 L2      |
| 1            | 0060.0600.6006.00-00 | 0x37     | 0xf287   | 967      | L1 L2      |
| 1            | 0060.0600.6006.01-00 | 0x2f     | 0xadfb   | 753      | L1 L2      |
| 1            | 0060.0600.6006.02-00 | 0x2f     | 0x5f95   | 819      | L1 L2      |
| 1            | 0070.0700.7007.00-00 | 0x33     | 0x48dd   | 1058     | L1 L2      |
| 1            | 0070.0700.7007.01-00 | 0x2f     | 0xadb2   | 1164     | L1 L2      |
| 1            | 0070.0700.7007.02-00 | 0x2e     | 0xdf8e   | 852      | L1 L2      |
| 2            | 0010.0100.1001.00-00 | 0x3e     | 0xb92f   | 1150     | L1 L2      |
| 2            | 0010.0100.1001.00-01 | 0x3c     | 0x1875   | 818      | L1 L2      |
| 2            | 0020.0200.2002.00-00 | 0x41     | 0x540f   | 1177     | L1 L2      |
| 2            | 0020.0200.2002.00-01 | 0x3f     | 0x2db9   | 699      | L1 L2      |
| 2            | 0030.0300.3003.00-00 | 0x3c     | 0x302f   | 1058     | L1 L2      |
| 2            | 0030.0300.3003.00-01 | 0x3f     | 0x5150   | 668      | L1 L2      |
| 2            | 0030.0300.3003.01-00 | 0x30     | 0xb518   | 915      | L1 L2      |
| 2            | 0030.0300.3003.02-00 | 0x2f     | 0xe113   | 1035     | L1 L2      |
| 2            | 0040.0400.4004.00-00 | 0x3e     | 0xa17b   | 657      | L1 L2      |
| 2            | 0040.0400.4004.00-01 | 0x3e     | 0x260b   | 1018     | L1 L2      |
| 2            | 0040.0400.4004.01-00 | 0x30     | 0x1511   | 1066     | L1 L2      |
| 2            | 0040.0400.4004.02-00 | 0x2f     | 0x27a7   | 1035     | L1 L2      |
| 2            | 0050.0500.5005.00-00 | 0x41     | 0x59d6   | 608      | L1 L2      |
| 2            | 0050.0500.5005.00-01 | 0x44     | 0xf165   | 1110     | L1 L2      |
| 2            | 0050.0500.5005.01-00 | 0x33     | 0x7709   | 584      | L1 L2      |
| 2            | 0050.0500.5005.02-00 | 0x31     | 0xa74    | 657      | L1 L2      |
| 2            | 0060.0600.6006.00-00 | 0x3d     | 0xd5ed   | 903      | L1 L2      |
| 2            | 0060.0600.6006.00-01 | 0x44     | 0xdc97   | 666      | L1 L2      |
| 2            | 0060.0600.6006.01-00 | 0x30     | 0x9024   | 1145     | L1 L2      |
| 2            | 0060.0600.6006.02-00 | 0x30     | 0xbc66   | 1020     | L1 L2      |
| 2            | 0070.0700.7007.00-00 | 0x3a     | 0x81fd   | 862      | L1 L2      |
| 2            | 0070.0700.7007.00-01 | 0x3e     | 0xf82a   | 765      | L1 L2      |
| 2            | 0070.0700.7007.01-00 | 0x30     | 0xbad5   | 658      | L1 L2      |
| 2            | 0070.0700.7007.02-00 | 0x2f     | 0x1ecb   | 675      | L1 L2      |
+--------------+----------------------+----------+----------+----------+------------+
LSP Count: 41
-------------------------------------------------------------------------------------
```

To display information for a specific IS-IS link state database:

```
# show network-instance green_default protocols isis database 1
--------------------------------------------------------------------------------------
Network-instance    : green_default
IS-IS instance      : default
--------------------------------------------------------------------------------------
| Level Number | Lsp Id               | Sequence | Checksum | Lifetime | Attributes |
+==============+======================+==========+==========+==========+============+
| 1            | 0010.0100.1001.00-00 | 0x33     | 0x1672   | 1048     | L1 L2      |
| 1            | 0020.0200.2002.00-00 | 0x35     | 0xd562   | 894      | L1 L2      |
| 1            | 0030.0300.3003.00-00 | 0x39     | 0x7762   | 1179     | L1 L2      |
| 1            | 0030.0300.3003.01-00 | 0x2f     | 0x4db6   | 886      | L1 L2      |
| 1            | 0030.0300.3003.02-00 | 0x2f     | 0x4ace   | 1188     | L1 L2      |
| 1            | 0040.0400.4004.00-00 | 0x3a     | 0xcd9f   | 1135     | L1 L2      |
| 1            | 0040.0400.4004.01-00 | 0x2f     | 0xf0ef   | 703      | L1 L2      |
| 1            | 0040.0400.4004.02-00 | 0x2f     | 0xa5f8   | 879      | L1 L2      |
| 1            | 0050.0500.5005.00-00 | 0x38     | 0xfbbb   | 785      | L1 L2      |
| 1            | 0050.0500.5005.01-00 | 0x31     | 0x3937   | 625      | L1 L2      |
| 1            | 0050.0500.5005.02-00 | 0x30     | 0xb86e   | 1148     | L1 L2      |
| 1            | 0060.0600.6006.00-00 | 0x37     | 0xf287   | 847      | L1 L2      |
| 1            | 0060.0600.6006.01-00 | 0x2f     | 0xadfb   | 633      | L1 L2      |
| 1            | 0060.0600.6006.02-00 | 0x2f     | 0x5f95   | 700      | L1 L2      |
| 1            | 0070.0700.7007.00-00 | 0x33     | 0x48dd   | 938      | L1 L2      |
| 1            | 0070.0700.7007.01-00 | 0x2f     | 0xadb2   | 1044     | L1 L2      |
| 1            | 0070.0700.7007.02-00 | 0x2e     | 0xdf8e   | 733      | L1 L2      |
+--------------+----------------------+----------+----------+----------+------------+
LSP Count: 17
--------------------------------------------------------------------------------------
```

7.4 Clearing IS-IS information

Procedure

To clear information for an IS-IS instance, use the **tools** commands below:

Example: Clear statistics

To clear statistics for an IS-IS instance running in a specified network instance:

```
# tools network-instance default protocols isis instance i1 statistics clear
```

Example: Clear link state database information

To clear link state database information for a level:

```
# tools network-instance default protocols isis instance i1 link-state-database clear
```

Example: Clear IS-IS adjacency information

To clear IS-IS adjacency information for an interface:

```
# tools network-instance default protocols isis instance i1 interface ethernet-1/1.1
 adjacencies clear
```

7.5 IS-IS weighted ECMP

Weighted IP ECMP, also known as UCMP (unequal cost multipath), allows the installation of a multipath route in the FIB, where the ECMP flow hashing distributes traffic by directing flows to each next-hop in a ratio proportional to the weight assigned to each next-hop. Multipath routes are the routes that have multiple next-hops.

Weighted ECMP (wECMP) distributes traffic unequally over multiple paths and uses available bandwidth more efficiently for better load balancing. For example, if you have four equal-cost paths to a destination, but one path has a lower bandwidth than the other three, you can use wECMP to assign more traffic to the higher bandwidth path and less traffic to the lower bandwidth paths.

The wECMP feature is available on 7250 IXR platforms. It is supported for both IS-IS IPv4 and IPv6 routes.

The high-level steps for configuring wECMP are:

1. Enabling weighted ECMP

2. Configuring weighted load-balancing over interface next-hops

The system normalizes the weights used in weighted ECMP according to the algorithm described in Normalizing datapath weights.

Note: When a BGP next-hop is resolved by an IS-IS route with weighted ECMP next-hops, any traffic using the BGP route and directed to the BGP next-hop (in case the BGP route has multiple paths) inherits the weighted ECMP load-balancing of the resolving IS-IS route.

7.5.1 Enabling weighted ECMP

Procedure

You can enable the weighted ECMP per IS-IS instance. wECMP can be configured for an IS-IS instance under a default network instance or network instance of type ip-vrf . Enabling weighted ECMP, triggers weighted ECMP programming for all eligible multipath IS-IS routes associated with the instance. To program an IS-IS route as a multipath route, all the multipaths must have an equal total path cost to the destination.

The ECMP weights (**load-balancing-weight**) are normalized based on the number of hash buckets per next-hop group (NHG). The sum of all normalized datapath weights must not exceed the hash bucket limit.

You must configure the maximum number of ECMP hash buckets used for IS-IS weighted ECMP routing on a per IS-IS instance basis. If the hash bucket limit is T and the normalized datapath weight of each next-hop is N_i then the sum of all N_i in each mutipath set (NHG) cannot exceed T. The system normalizes the weights used in weighted ECMP according to the algorithm described in Normalizing datapath weights.

The 7250 IXR platforms have a 256 hash bucket limit. Datapath resources can be conserved by configuring a lower value for the bucket size. However, this reduces the granularity or fine distribution of the traffic. With a smaller bucket size, there are fewer next-hop options available for load balancing, which can limit the flexibility and precision of traffic distribution. You cannot reduce the hash bucket limit below the ECMP limit of the IS-IS instance.

The following example enables weighted ECMP per IS-IS instance:

Example: Enabling weighted ECMP

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance mgmt {
        protocols {
            isis {
                instance test {
                    weighted-ecmp {
                        admin-state enable
                        max-ecmp-hash-buckets-per-next-hop-group 56
                    }
                }
            }
        }
    }
```

Where,

- **admin-state**
  When set to **enable**, triggers weighted ECMP programming for all eligible multipath IS-IS routes associated with the instance. The default is **disable**. This ensures backward compatibility

with previous releases that only supported ECMP wherein the traffic was distributed in equal proportion.

- **max-ecmp-hash-buckets-per-next-hop-group**
  Specify the maximum number of ECMP hash buckets per next-hop-group. Default bucket size is `128`.

  > **Note:** The configured value of **max-ecmp-hash-buckets-per-next-hop-group** must always be greater than or equal to **max-ecmp-paths** (`network-instance.protocols.isis.instance.max-ecmp-paths`).

Example: Disabling weighted ECMP

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance mgmt {
        protocols {
            isis {
                instance test {
                    weighted-ecmp {
                        admin-state disable
                        max-ecmp-hash-buckets-per-next-hop-group 1
                    }
                }
            }
        }
    }
```

You can disable the weighted ECMP per ISIS instance. When you disable the weighted ECMP in an IS-IS instance, all IS-IS routes computed by this instance is programmed in the FIB with classic ECMP programming. This means no weights are assigned to the next-hops, even if some or all of the next-hop interfaces in a multipath set have a non-zero load balancing weight (`load-balancing-weight`) configured.

7.5.2 Configuring weighted load-balancing over interface next-hops

Procedure

When the weighted ECMP feature is enabled, the IS-IS route linked to the routing instance can be programmed into the datapath. This allows for weighted load-balancing across the interface next-hops of the route.

In order for weighted ECMP to be supported across the interface next-hops of an IS-IS route the following conditions must be met:

- All ECMP next-hops must be interface next-hops
- All next-hop interfaces are configured with non-zero load-balancing weights.

To configure load-balancing weight, set the parameter **load-balancing-weight** under `network-instance.protocols.isis.instance.interface.level.weighted-ecmp`. This parameter can be set to a static value, or you can choose to use the options `auto` or `none`. The static value assigned must be between `1` and `4294967295`.

If all the next-hop interfaces in the multipath set have a **load-balancing-weight**, which is either automatically derived from the port/LAG bandwidth using the `auto` option or configured with a static value, then the route is programmed in the FIB with wECMP. If one or more next-hop interfaces have zero weight (`load-balancing-weight = none`), then the wECMP load-balancing falls back to classic ECMP operation and equally distributes the traffic.

In the NHG for the IS-IS route, each next-hop is assigned a non-zero weight. In SR Linux, the wECMP programming capability is available even if some multipath next-hop interfaces are connected to different adjacent routers.

The following example configures weighted load-balancing over interface next-hops:

Example: Configure weighted load-balancing over interface next-hops

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance mgmt {
        protocols {
            isis {
                instance test {
                    interface mgmt0.0 {
                        weighted-ecmp {
                            load-balancing-weight auto
                        }
                    }
                }
            }
        }
    }
```

Here the **load-balancing-weight** value is automatically derived from the port/LAG bandwidth.

7.5.3 Normalizing datapath weights

The ECMP weights (**load-balancing-weight**) are normalized based on the number of hash buckets (**max-ecmp-hash-buckets-per-next-hop-group**) defined per NHG. The sum of all normalized datapath weights must not exceed the hash bucket limit.

The datapath programming of the NHG assigns N_i hash buckets to each next-hop. The value of N_i is determined by normalizing it using the following method:

1. Calculate the Greatest Common Divisor (GCD) of all datapath weights.
2. Divide all datapath weights by the GCD.
3. If the sum of normalized weights is less than the maximum number of buckets, weight determination process is complete.
   Or
4. If the sum of normalized weights exceeds the bucket size,
   a. Assign each nexthop a minimum of one bucket.
   b. Distribute the remaining buckets (max_buckets - num_nexthops) to each nexthop based on the ratio of each nexthop's weight to the total weight (rounded down). There may be cases where no extra buckets are assigned.

7.6 Multi-instance IS-IS

Multi-instance IS-IS (MI-ISIS), defined in RFC 8202, allows multiple instances of IS-IS to operate on a single circuit. To configure MI-ISIS in SR Linux, you set the instance identifier (IID), which identifies an IS-IS instance, and enable sending of the IID-TLV in IS-IS PDUs. The IID-TLV identifies the IS-IS instance and topology.

MI-ISIS uses the following route-selection tie-break mechanism:

- When multiple MI-ISIS instances contain an identical route, the route with the lowest metric is selected as the active route and used for forwarding.
- When multiple MI-ISIS instances have an identical route with identical metric, the route from the instance with the lowest IID is selected as the active route and used for forwarding.

When MI-ISIS is used in combination with ECMP, then only active RTM routes of the same MI-ISIS instance can be load-balanced. ECMP load-balancing between MI-ISIS instances is not supported.

7.6.1 Configuring the IID

Procedure

The IID is a numerical instance identification number (0-127) and is used to identify the local IS-IS instance and to populate the IID-TLV when MI-ISIS is configured. In an MI-ISIS configuration, each IS-IS instance must have a unique name and can only be associated with a single unique IID within a network instance. The default IS-IS instance has an IID of 0.

Example

The following example configures an IID for an IS-IS instance:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance i1 instance-id
    network-instance default {
        protocols {
            isis {
                instance i1 {
                    instance-id 2
                }
            }
        }
    }
```

7.6.2 Enabling the IID-TLV

Procedure

To configure MI-ISIS, you enable SR Linux to include the IID-TLV for the IS-IS instance in IS-IS PDUs. The IID-TLV includes the configured IID and identifies the instance-specific topology or topologies to which the PDU applies.

Example

The following example enables the system to send the IID-TLV for an IS-IS instance :

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance default {
        protocols {
            isis {
                instance i1 {
                    iid-tlv true
                }
            }
        }
    }
```

If `iid-tlv` is set to `false`, the configured IID has only local relevance. Interfaces enabled in such IS-IS instances establish adjacencies unaware of any other instances. The router originates link state packets without the IID-TLV and expects other routers in the area not to advertise any IID-TLV either.

7.6.3 Displaying MI-ISIS information

Procedure

You can display information about the MI-ISIS configuration using `show` reports or `info from state` output.

Example: Display IID-TLV information

The following example displays the configured IID-TLV and any received IID-TLVs for an IS-IS instance:

```
--{ + candidate shared default }--[  ]--
# info with-context from state network-instance default protocols isis instance My
Instance2
    admin-state enable
    instance-id 2
    level-capability L2
    max-ecmp-paths 1
    poi-tlv false
    iid-tlv true
    export-policy MyPolicy
    hello-padding disable
    enable-csnp-on-p2p-links true
    oper-state up
    oper-system-id 0000.0000.8521
    net [
        49.0000.0000.8521.00
    ]
    oper-area-id [
        49
    ]
....
<snip>
....
    level 2 {
        metric-style wide
        <snip>
        link-state-database {
            lsp 0000.0000.8521.00-00 {
                maximum-area-addresses 3
                <snip>
                tlvs {
                    <snip>
                    tlv instance-id {
                        instance-ids {
                            instance-id 2 {
                                topology-id [
                                    0
                                ]
                            }
                        }
                    }
```

7.7 Multi-Topology IS-IS MT0 and MT2

Note: This feature is supported on 7250 IXR and 7730 SXR systems.

SR Linux supports Multi-Topology (MT) Routing for IS-IS, as defined in RFC 5120. Multi-Topology IS-IS (MT-ISIS), allows multiple independent IP routing topologies within a single IS-IS domain.

MT-ISIS support within SR Linux enables the creation of separate IPv4 unicast and IPv6 unicast topologies within the IS-IS domain. These topologies contribute routes to specific route tables, allowing for non-congruent topologies between different routing tables. As a result, networks can control the links or nodes that are used for forwarding different types of traffic.

For example, MT-ISIS can enable all links to carry IPv4 traffic, while only a subset of links can also carry IPv6 traffic.

SR Linux supports the following multi-topologies:

- MT-ID 0 (standard IPv4 and IPv6 topology)
- MT-ID 2 (IPv6-only topology)

IPv4 or IPv6 routing

When an IS-IS instance is created, IPv4 routing is enabled by default and is restricted to Multi-Topology IS-IS (MT-IS-IS) with topology identifier MT0. This configuration ensures that the IS-IS instance is initially set up to handle IPv4 routing within the default topology, MT0, as specified by RFC 5120.

Unlike MT0, which is the default for IPv4, MT2 is not enabled by default and must be configured to manage IPv6 traffic. You must explicitly enable IPv6 routing in an IS-IS instance. IPv6 routing can be configured for either the MT0 or MT2 topology, with MT0 being the default topology for IPv6.

For information about enabling IPv6 routing, see Enabling IPv6 unicast routing for an IS-IS instance , and for MT2 topology configuration details, see Configuring the default MT2 topology.

7.7.1 Enabling IPv6 unicast routing for an IS-IS instance

Procedure

Use the command `.network-instance.[name].protocols.isis.instance.[instance-name].ipv6-unicast.admin-state.enable` to enable IPv6 unicast routing for an IS-IS instance.

Example

The following example enables IPv6 unicast routing for an IS-IS instance.

```
--{ candidate shared default }--[  ]--
info with-context network-instance default protocols isis instance srl_isis_instance
    network-instance default {
        protocols {
            isis {
                instance srl_isis_instance {
                    ipv6-unicast {
                        admin-state enable
                    }
                }
            }
        }
    }
```

7.7.2 Configuring the default MT2 topology

Procedure

To configure the default MT2 topology for IPv6 routing, set the parameter `multi-topology` under the `.network-instance.[name].protocols.isis.instance.[instance-name].ipv6-unicast.` context to `true`.

Example

The following example enables IPv6 routing for an IS-IS instance.

```
--{ candidate shared default }--[  ]--
info with-context network-instance default protocols isis instance srl_isis_instance
    network-instance default {
        protocols {
            isis {
                instance srl_isis_instance {
                    ipv6-unicast {
                        admin-state enable
                        multi-topology true
                    }
                }
            }
        }
    }
```

7.8 IS-IS extensions for Traffic Engineering (TE)

Note: This feature is supported on 7250 IXR Gen 2, 7250 IXR Gen 2c+, and 7730 SXR systems.

IS-IS TE extensions allow network operators to encode and flood traffic engineering information throughout a network using TE-enabled links.

IGPs (IS-IS/OSPF) use only an interface metric to find the shortest path to a destination without considering factors such as bandwidth and link delay. For traffic engineering computations, link attributes such as available bandwidth, maximum reservable bandwidth, TE metric, and link affinity are required. The IS-IS TE extensions contain the TLVs and sub-TLVs that specify these link attributes. IS-IS link-state PDUs include these TLVs. Flooding algorithms used by link-state IGPs ensure that link attributes are distributed evenly across all routers in the routing domain.

7.8.1 Enabling advertisement of IS-IS TE TLVs/sub-TLVs

Procedure

SR Linux supports the transmission and reception of Traffic Engineering (TE)-related TLV and sub-TLV fields in IS-IS Link State Packets (LSPs) to share network resource information. The `network-instance.traffic-engineering` container enables and configures TE functionality for both IPv4 and IPv6. When TE is enabled, the TE-related TLVs and sub-TLVs are incorporated into the LSPs generated by the Interior Gateway Protocol (IGP).

To enable advertising of TLVs and sub-TLVs, set the `advertisement` parameter to `true` within the `network-instance.protocols.isis.instance.traffic-engineering` container. By default, the TE `advertisement` parameter is set to `false`.

Example

In the following example, the IS-IS TE TLVs and sub-TLVs are included in the IS-IS LSP.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance test traffic-
engineering advertisement
    network-instance default {
        protocols {
            isis {
                instance test {
                    traffic-engineering {
                        advertisement true
                    }
                }
            }
        }
    }
```

7.8.2 TE router ID TLV

A TE router ID uniquely identifies a router in an IGP TE domain. The router ID must be operationally active within the IS-IS instance and associated with a system or loopback address.

For the IPv4 and IPv6 address families, IS-IS advertises the IPv4 router ID using TLV 134 as specified in [RFC5305] and the IPv6 router ID using TLV 140 as specified [RFC6119], for use within an IS-IS area.

IPv4 TE router ID TLV

The IPv4 Traffic Engineering (TE) Router ID TLV (Type 134) contains the 4-octet router ID of the router originating the LSP. According to RFC5305, the TE Router ID TLV can be advertised by a router regardless of whether IPv4 TE is enabled.

If IPv4 TE is disabled, the IPv4 TE Router ID TLV is advertised only if the `network-instance.default.router-id` parameter is configured, regardless of whether the router ID is reachable. If the `network-instance.default.router-id` parameter is not configured, then IPv4 TE Router ID TLV is not advertised.

If IPv4 TE is enabled, the value of the advertised IPv4 TE Router ID TLV is selected based on a specific order of preference, which is as follows:

1. `network-instance.default.protocol.isis.instance.traffic-engineering.ipv4-te-router-id`
2. If not available, `network-instance.default.traffic-engineering.ipv4-te-router-id` value is selected.
3. If neither is available, `network-instance.default.router-id` value is selected.

Note: When IPv4 TE is enabled, only the IPv4 TE Router ID that is operationally active within the IS-IS instance and associated with a system or loopback address is eligible to be advertised as per the above priority list.

When IPv4 routing is disabled and IPv6 TE is enabled with router capability advertised, the IPv4 router ID TLV is not advertised. However, another ipv4-router-id TLV set to 0.0.0.0 is included within the router capability top-level TLV, ensuring compliance with the RFC7981 standard.

IPv6 TE router ID TLV

The IPv6 router ID can be advertised in top-level TLV 140, defined in RFC6119, or within the Router Capability sub-TLV12, as defined in RFC5305.

The IPv6 TE router ID TLV is advertised when all of the following conditions are satisfied:

- IPv6 and TE must be enabled
  and
- A configured routable IPv6 TE router interface address must exist
  and
- The advertisement of TE-related TLVs and sub-TLVs must be enabled by setting the `advertisement` parameter to true within the `network-instance.traffic-engineering` container.

If IPv6 or TE is disabled, the IPv6 TE router ID TLV 140 is not advertised.

The value of the advertised IPv6 router ID is selected based on a specific order of preference, as follows:

1. `network-instance.default.protocol.isis.instance.traffic-engineering.ipv6-te-router-id <value>`
2. If not available, `network-instance.default.traffic-engineering.ipv6-te-router-id <value>` is selected.
3. If neither is available, `interface.subinterface.ipv6.address xx:xx::xx/128` is selected.

   Note:
   A configured and enabled IPv6 loopback address is selected for advertisement.

Note: Only the `ipv6-te-router-id` that is operationally active within the IS-IS instance and associated with a system or loopback address is eligible for advertisement in accordance with the above priority list.

RFC 7981 implication to IPv6 Router IDs

RFC 7981 defines IS-IS Router Capability TLV, which allows routers to announce their capabilities within an IS-IS level or across the entire routing domain. The IS-IS router capability TLV consists of multiple sub-TLVs.

Enabling IPv6 TE and advertising router capability results in the advertisement of both the IPv6 Router ID TLV 140 and the IPv6 Router ID TLV 12 within the Router Capability TLV.

The Router Capability sub-TLV12 as defined in RFC 5316, is used when the IPv6 TE Router ID must be distributed to all routers within an entire IS-IS routing domain, instead of being restricted to an area scope. In SR Linux, sub-TLV 12 is advertised only when IPv4 unicast routing is explicitly disabled using the `network-instance.default.protocols.isis.instance.ipv4-unicast.admin-state disable` command.

7.8.2.1 Advertising IPv4 TE router ID TLV

Procedure

A TE router ID must be a unique and stable address within the IS-IS domain. When IPv4 TE is enabled, the advertised IPv4 TE router ID TLV value is selected in a specific order of preference. For preference details, see IPv4 TE router ID TLV.

The following configurations enable the advertisement of the IPv4 router ID and TE link parameters using the legacy TE encoding defined in RFC 5305.

Example: Advertise IPv4 router ID TLV using IS-IS TE router ID

In the following configuration example, IPv4 router ID (`192.168.0.0`) configured within the IS-IS TE instance is selected for advertising IPv4 TE router ID TLV.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance test traffic-
engineering ipv4-te-router-id
    network-instance default {
        protocols {
            isis {
                instance test {
                    traffic-engineering {
                        ipv4-te-router-id 192.168.0.0
                    }
                }
            }
        }
    }
```

Example: Advertise IPv4 router ID TLV using network instance TE router ID

In the following configuration example, IPv4 router ID (`192.168.11.1`) configured within a default network instance with TE capabilities is selected for advertising IPv4 TE router ID TLV.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering ipv4-te-router-id
    network-instance default {
        traffic-engineering {
            ipv4-te-router-id 192.168.11.1
        }
    }
```

Example: Advertise IPv4 router ID TLV using default network instance router ID

In the following configuration example, IPv4 router ID (`192.0.5.1`) configured within a default network instance is selected for advertising IPv4 TE router ID TLV.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        router-id 192.0.5.1
    }
```

7.8.2.2 Advertising IPv6 TE router ID TLV

Procedure

A TE router ID must be a unique and stable address within the IS-IS domain. When IPv6 TE is enabled, the advertised IPv6 TE router ID TLV value is selected in a specific order of preference. For preference details, see IPv6 TE router ID TLV.

The following configurations enable the traffic engineering behavior with IPv6 TE links. The IS-IS instance automatically advertises the IPv6 TE TLVs, and sub-TLVs defined in RFC6119. For information about IPv6 TE router ID TLV, see IPv6 TE router ID TLV

Example: Advertise IPv6 router ID TLV using an IS-IS TE router ID

In the following configuration example, IPv6 router ID (`2001:db8::1`) configured within the IS-IS TE instance is selected for advertising IPv4 TE router ID TLV.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance test traffic-
engineering ipv6-te-router-id
    network-instance default {
        protocols {
            isis {
                instance test {
                    traffic-engineering {
                        ipv6-te-router-id 2001:db8::1
                    }
                }
            }
        }
    }
```

Example: Advertise IPv6 router ID TLV using default network instance TE router ID

In the following configuration example, IPv6 router ID (`2001:db8::8`) configured within a default network instance with TE capabilities is selected for advertising IPv4 TE router ID TLV.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering ipv6-te-router-id
    network-instance default {
        traffic-engineering {
            ipv6-te-router-id 2001:db8::8
        }
    }
```

Example: Advertise IPv6 router ID TLV using subinterface IPv6 address

In the following configuration example, the value of IPv6 address (`2002::1234:abcd:ffff:c0a8:101/64`) configured within a subinterface is selected for advertising IPv6 TE router ID TLV.

```
--{ * candidate shared default }--[  ]--
# info with-context interface ethernet-1/1 subinterface 0 ipv6
    interface ethernet-1/1 {
        subinterface 0 {
            ipv6 {
                address 2002::1234:abcd:ffff:c0a8:101/64 {
                }
            }
        }
    }
```

7.8.3 Advertising TE attributes using legacy and ASLA TLVs

When `network-instance.default.protocols.isis instance.default traffic-engineering advertisement` is enabled, the router advertises TE TLVs.

IS-IS allows advertising the protocol enabled on a TE-link by using the Application Specific Link Attributes (ASLA) sub-TLV as per draft-ietf-isis-te-app (RFC 8919). The router receiving the link TE attributes can identify the enabled application on the advertising router. For backward compatibility, the router continues to support the legacy mode of advertising link TE attributes, as recommended in RFC 5305. For information about the legacy mode of advertising link TE attributes, see Enabling advertisement of TE attributes in legacy mode.

SR Linux supports decoding of received TLVs and maintains the details in the LSDB YANG state model.

Table 10: TE attributes received in legacy TLV and sub-TLVs

Name	TLV	Sub TLV	Decoding in LSDB YANG
Admin group	22	3	Yes
Link local/remote identifiers	22	4	Yes
IPv4 interface address	22	6	Yes
IPv4 neighbor address	22	8	Yes
Maximum link bandwidth	22	9	Yes
Maximum reservable link bandwidth	22	10	Yes
Unreserved bandwidth	22	11	Yes
Extended administrative group	22	14	Yes
Link protection type	22	20	Yes
Link delay variation	22	35	Yes

Name	TLV	Sub TLV	Decoding in LSDB YANG
Link loss	22	36	Yes
Residual bandwidth	22	37	Yes
Available bandwidth	22	38	Yes
Utilized bandwidth	22	39	Yes
IPv6 interface address	22	12	Yes
IPv6 neighbor address	22	13	Yes
TE metric	22	18	Yes
Unidirectional link delay	22	33	Yes
Min/max unidirectional link delay	22	34	Yes
TE IPv4 router ID	134	NA	Yes
IPv4 SRLG	138	NA	Yes
IPv6 SRLG	139	NA	Yes
TE IPv6 router ID	140	NA	Yes

The table below lists the TE attributes received in ASLA TE TLVs. SR Linux supports decoding of received ASLA TLVs and maintains the details in the LSDB telemetry database.

Table 11: TE attributes received in ASLA TLV and sub-TLVs

Name	TLV	Sub TLV	Sub-Sub TLV	Decoding in LSDB YANG
Admin group	22	16	3	Yes
Maximum link bandwidth	22	16	9	Yes
TE metric	22	16	18	Yes
Min/max unidirectional link delay	22	16	34	Yes
Extended admin group	22	16	14	Yes

7.8.3.1 Enabling advertisement of TE attributes in legacy mode

Procedure

SR Linux supports advertising of TE attributes using either the legacy-link-attribute-advertisement or ASLA (application-specific link attributes) encodings, or both.

Example

The following example configuration enables advertisement of TE attributes in legacy mode. For disabling the advertisement of TE attributes, set `legacy-link-attribute-advertisement` to `false`.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance test traffic-
engineering legacy-link-attribute-advertisement
    network-instance default {
        protocols {
            isis {
                instance test {
                    traffic-engineering {
                        legacy-link-attribute-advertisement true
                    }
                }
            }
        }
    }
```

7.9 Non-Stop Routing (NSR) IS-IS

Note: This feature is currently supported on 7250 IXR Gen 2, 7250 IXR 6e/10e, and 7250 IXR Gen 3 platforms.

IS-IS Non-Stop Routing (NSR) is a high-availability feature that enables a router to continue forwarding traffic without disruption during a control plane switchover, such as when the primary routing processor (CPM) fails or is manually restarted.

With Non-Stop Routing or NSR enabled on the SR Linux routers, routing peers remain unaware of any faults in the routing process. If a fault occurs, a reliable and deterministic switchover to the inactive control complex occurs, ensuring that routing topology (IS-IS) and reachability remain unaffected, even in the presence of routing updates.

NSR achieves high availability through parallelization by always maintaining up-to-date routing state information on standby routing instances.

IS-IS NSR is always enabled by default and does not require configuration. Non-stop forwarding or NSF, is only necessary when graceful restart is used. For more information about NSF, see the "Non-stop forwarding" section of the *SR Linux Configuration Basics Guide*. NSR functions independently of the NSF configuration.

7.10 Multi-part TLVs in IS-IS

The IS-IS protocol uses Type-Length-Value (TLV) encodings with an 8-bit `Length` field, limiting the maximum size of any TLV to 255 octets. When more information needs to be advertised (for example, extended reachability, neighbor or capability data), a router may break the information into multiple TLVs of the same type and key. These are called multi-part TLVs (MP-TLVs). MP-TLVs allow routers to carry large sets of data that exceed individual TLV size limits.

7.10.1 Enabling MP-TLVs

About this task

SR Linux supports per-level and per-type control of MP-TLV origination for IS-IS and control over capability advertisement. For interoperability, the default receiving and concatenation behavior is set to always on.

Caution: Deployments with older or non-compliant routers may suffer interoperability issues if MP-TLVs are enabled. Nokia recommends having MP-TLVs disabled if routers in a domain do not support MP-TLVs. Only enable this feature if all routers in the domain support MP-TLVs.

Procedure

MP-TLVs can be enabled using the following parameters under the **network-instance protocols isis instance multi-part-tlv** command:

- **neighbor** — when set to **true**, enables the origination of multiple instances of neighbor TLVs (TLV 22, TLV 222)
- **srlg** — when set to **true**, enables the origination of multiple instances of SRLG TLVs (TLV 138, TLV 238)

Note: The IPv6-SRLG TLV cannot be an MP-TLV, as outlined in Section 4.4 of RFC 6119.

The following shows an example configuration of the **network-instance protocols isis instance multi-part-tlv** command.

Example: MP-TLV configuration

```
--{ * candidate shared default }--[  ]--
A:root@srl1# info with-context network-instance default
    network-instance default {
        protocols {
            isis {
                instance 1 {
                    multi-part-tlv {
                        neighbor true
                        srlg true
                    }
                }
            }
        }
    }
```

8 IP ECMP Load Balancing

Equal-Cost Multipath Protocol (ECMP) refers to the distribution of packets over two or more outgoing links that share the same routing cost. Static, IS-IS, OSPF, and BGP routes to IPv4 and IPv6 destinations can be programmed into the datapath by their respective applications, with multiple IP ECMP next hops.

The SR Linux device load-balances traffic over multiple equal-cost links with a hashing algorithm that uses header fields from incoming packets to calculate which link to use. When an IPv4 or IPv6 packet is received on a subinterface, and it matches a route with a number of IP ECMP next hops, the next hop that forwards the packet is selected based on a computation using this hashing algorithm. The goal of the hash computation is to keep packets in the same flow on the same network path, while distributing traffic proportionally across the ECMP next hops, so that each of the *N* ECMP next hops carries approximately 1/*N*th of the load.

The hash computation takes various key and packet header field values as inputs and returns a value that indicates the next hop. The key and field values that can be used by the hash computation depend on the platform, packet type, and configuration options, as follows:

On 7250 IXR platforms, the following can be used in the hash computation:

- Hash-seed (0 to 65535)
  On 7250 IXR-6, 7250 IXR-10, 7250 IXR-X1b, and 7250 IXR-X3b devices, the hash-seed can be system-generated (the default) or user-specified. If the hash-seed is system-generated, SR Linux generates a hash-seed using the least-significant 16 bits of the base chassis MAC address.

  On 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e devices with 36 x 800 IMM, the system randomly generates a per-interface hash-seed using the chassis base MAC address and the 64-bit port ID as inputs; this ensures that the hash-seed is the same after every restart of the port, LAG, or IMM.
- On Gen 3 linecards on 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e devices, a hash-seed can be specified within a hash-profile. When the hash-profile is applied to an interface, the specified hash-seed takes effect for the interface.
- For IPv4 TCP/UDP non-fragmented packets, the hash algorithm can use source IPv4 address, destination IPv4 address, IP protocol, Layer 4 source port, Layer 4 destination port, and some of the terminated Ethernet header fields. The algorithm is asymmetric; that is, inverting source and destination pairs does not produce the same result.
- For IPv6 TCP/UDP non-fragmented packets, the hash algorithm can use source IPv6 address, destination IPv6 address, IPv6 flow label (even if it is 0), IP protocol (IPv6 next-header value in the last extension header), Layer 4 source port, Layer 4 destination port, and some of the terminated Ethernet header fields. The algorithm is symmetric; that is, inverting source and destination pairs produces the same result.

On 7250 IXR, 7220 IXR-H4, 7220 IXR-H5, 7250 IXR-X1b, and 7250 IXR-X3b devices, if an IP packet being forwarded has a UDP destination port of 4791, indicating it is carrying an RDMA over Converged Ethernet version 2 (ROCEv2) payload, the 24-bit Dest Queue-pair value in the ROCEv2 header (BTH+) is added to the hash algorithm. In this case, hashing is based on the existing 5-tuple flow and the new Dest Queue-pair value. The system determines an IP packet's Dest Queue-pair value based on the format of the BTH+ header by looking for the 24-bit value that is 5-bytes offset from the end of the UDP header. On 7220 IXR-H4 and 7220 IXR-H5 devices, the UDF mechanism is used to match on qualifying packets and extract the Dest Queue-pair value from the specified offset.

On the 7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D3, 7220 IXR-H2, and 7220 IXR-H3 the following can be used in the hash computation:

- Hash-seed (0 to 65535), which can be system-generated (the default) or user-specified. If the hash-seed is system-generated SR Linux generates a hash-seed using the least-significant 16 bits of the base chassis MAC address.
- For IPv4 TCP/UDP non-fragmented packets, the hash algorithm can use VLAN ID, source IPv4 address, destination IPv4 address, IP protocol, Layer 4 source port, and Layer 4 destination port. The algorithm is asymmetric.
- For IPv6 TCP/UDP non-fragmented packets, the hash algorithm can use VLAN ID, source IPv6 address, destination IPv6 address, IPv6 flow label (even if it is 0), IP protocol (IPv6 next-header value in the last extension header), Layer 4 source port, and Layer 4 destination port.

On 7215 IXS platforms, the following can be used in the hash computation:

- Source IP address
- Destination IP address
- Layer 4 source port
- Layer 4 destination port
- Hash seed
- IPv6 flow label
- Received MPLS labels (terminated and non-terminated)
- IP protocol number

8.1 Avoiding hash polarization

Hash polarization is when the hash algorithm selects ECMP next-hops inefficiently; for example, when the system always chooses the same next-hop for specific packet flows. Hash polarization can occur when adjacent routers use the same hash-seed.

To avoid hash polarization effects, ensure that directly connected nodes have unique hash-seeds. You can do this by explicitly configuring the hash-seeds, or by verifying that the state value of system-generated hash-seeds is different on adjacent routers.

To check the state value of system-generated hash-seeds, use the **info from state** command.

The following example displays the system-wide hash-seed (either user-configured or system-generated) on 7220 IXR and 7250 IXR platforms:

```
--{ + running }--[  ]--
# info with-context from state system load-balancing hash-options
    system {
        load-balancing {
            hash-options {
                hash-seed 2203
            }
        }
    }
```

The following example displays the system-generated, interface-specific hash-seed on 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e platforms:

```
--{ + running }--[  ]--
# info with-context from state interface ethernet-1/1 load-balancing hash-seed
    interface ethernet-1/1 {
        load-balancing {
            hash-seed 41521
        }
    }
```

The following enhancements can result in better load balancing results when hash polarization is a factor:

- **Per-complex seed randomization**

  The **system load-balancing hash-options per-complex-randomization** configuration accepts boolean values of **true** and **false**, with the default value of **true**. When set to **true**, the system adds some non-zero random bits to each hash calculation. The random bits are generated independently for each forwarding complex, which improves hashing performance by reducing polarization across adjacent systems.

  When set to **false**, there is no randomization per forwarding complex. This enhancement applies to all IP and MPLS packets that arrive on a routed subinterface that are forwarded at Layer 3.

- **Ethernet header fields for IP/MPLS hashing**

  The `system load-balancing hash-options terminated-eth-fields` configuration accepts boolean values of **true** and **false**, with the default value of **false**. When set to **true**, the system includes fields from the terminated Ethernet header fields in the hash calculation for IP and MPLS packets arriving on a routed subinterface that are forwarded at Layer 3. This includes the following fields:

  - MAC SA
  - MAC DA
  - Ethertype and all VLAN TPIDs
  - first and second VLAN ID (if present)

Checking hash polynomials (7250 IXR platforms)

On 7250 IXR platforms, the system computes a set of load-balancing keys for each received packet. Packets that belong to the same 5-tuple flow have the same load-balancing keys, ensuring they follow the same path through the network and do not get misordered.

For each received packet, the system computes load-balancing keys for the following clients or "hash-users":

- key 1 is used to select an ECMP level 1 FEC member
- key 2 is used to select an ECMP level 2 FEC member
- key 3 is used to select an ECMP level 3 FEC member
- key 4 is used to select a LAG member
- key 5 is used to generate a value to be stamped into a network header at egress (for example, IPv6 flow label or VXLAN UDP source port)

To create the load-balancing keys, the system sends a primary key (computed from the combined CRC values of each of the packet header layers), along with the user-configured or system-generated hash-seed, to one of eight polynomial functions available on the device. Each hash-user is assigned one of the eight polynomial functions. The system uses the primary key and the hash-seed as input to the polynomial function, which returns the load-balancing key for the hash-user as output.

There is a greater risk of hash polarization if two adjacent routers use the same polynomial function for the same hash-user, instead of two different polynomial functions. You can use an **info from state** command to display the polynomial function assigned to the each hash-user; for example:

```
--{ + running }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 1 load-balancing hash-
user * hash-polynomial
    platform {
        linecard 1 {
            forwarding-complex 1 {
                load-balancing {
                    hash-user level-1-fec {
                        hash-polynomial 1
                    }
                    hash-user level-2-fec {
                        hash-polynomial 2
                    }
                    hash-user level-3-fec {
                        hash-polynomial 3
                    }
                    hash-user lag {
                        hash-polynomial 4
                    }
                    hash-user network-header {
                        hash-polynomial 5
                    }
                }
            }
        }
    }
```

If adjacent routers use the same hash-polynomial function for the same hash-user, you can avoid potential hash polarization by changing the hash-seed on one of the routers.

8.2 Configuring IP ECMP load balancing

Procedure

To configure IP ECMP load balancing, you specify hash-options that are used as input fields for the hash calculation, which determines the next hop for packets matching routes with multiple ECMP hops.

Example: Configure hash options for IP ECMP load balancing

The following example configures hash options for IP ECMP load balancing, including a hash-seed and packet header field values to be used in the hash computation.

```
--{ * candidate shared default }--[  ]--
# info with-context system load-balancing
    system {
        load-balancing {
            hash-options {
                hash-seed 128
                ipv6-flow-label false
            }
        }
    }
```

On 7250 IXR-6, 7250 IXR-10, 7250 IXR-X1b, and 7250 IXR-X3b devices, if no value is configured for the **hash-seed**, the default is for the system to generate a hash-seed using the least-significant 16 bits of the base chassis MAC address. If a hash-option is not specifically configured either **true** or **false**, the default for the hash option is **true**. The user-configured or system-generated hash-seed applies system-wide.

On 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e devices, SR Linux randomly generates a per-interface hash-seed using the chassis base MAC address and the 64-bit port ID as inputs.

On 7250 IXR devices, if **source-address** is configured as a hash option, the **destination-address** must also be configured as a hash option. Similarly, if **source-port** is configured as a hash option, the **destination-port** must also be configured as a hash option.

Example: Configure IP ECMP load balancing based only on IPv4 source and destination address (7250 IXR-X1b only)

The following example configures the hash options so that the SR Linux device load-balances IPv4 traffic using only the IPv4 source and destination address. In this example, fields such as Layer 4 protocol and Layer 4 source/destination ports are not used in the load-balancing calculation. The result of this configuration is that all IPv4 traffic with same source address and destination address pair is forwarded to the same next-hop and on the same port always.

```
--{ * candidate shared default }--[  ]--
# info with-context system load-balancing hash-options
    system {
        load-balancing {
            hash-options {
                destination-address true
                destination-port false
                ipv6-flow-label false
                protocol false
                source-address true
                source-port false
                mpls-label-stack false
                per-complex-randomization false
                terminated-eth-fields false
            }
        }
    }
```

Example: Configure a hash-profile and apply it to an interface

On 7250 IXR-6e, 7250 IXR-10e, and 7250 IXR-18e devices (Gen 3 linecards only), you can specify a hash-seed in a hash-profile and apply the hash-profile to an interface. The interface uses the hash-seed configured in the hash profile.

The following example configures a hash-seed in a hash-profile, then applies the hash-profile to an interface.

```
--{ * candidate shared default }--[  ]--
# info with-context system load-balancing hash-profile p1
    system {
        load-balancing {
            hash-profile p1 {
                hash-seed 123
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info with-context interface ethernet-1/1 load-balancing hash-profile
    interface ethernet-1/1 {
        load-balancing {
            hash-profile p1
        }
    }
```

Example: Configure a non-zero parser seed

On 7250 IXR Gen 2c+ and 7250 IXR Gen 3 platforms, you can add some randomness to each forwarding complex, as shown in the following example.

```
--{ * candidate shared default }--[  ]--
# info with-context system load-balancing
    system {
        load-balancing {
            hash-options {
                per-complex-randomization true
            }
        }
    }
```

Example: Configure Ethernet header fields for IP/MPLS hashing

On 7250 IXR Gen 2c+ and 7250 IXR Gen 3 platforms, you can configure hashing to use Ethernet header fields in the hash calculation for IP and MPLS packets.

```
--{ * candidate shared default }--[  ]--
# info with-context system load-balancing
    system {
        load-balancing {
            hash-options {
                terminated-eth-fields true
            }
        }
    }
```

8.3 Resilient ECMP hashing

Note: Resilient ECMP hashing is supported on the following devices: 7250 IXR-6, 7250 IXR-10, 7250 IXR-6e, 7250 IXR-10e, 7250 IXR-18e, 7250 IXR-X1b, 7250 IXR-X3b, 7220 IXR-D1, 7220 IXR-D2, and 7220 IXR-D3.

For some IP prefixes with IP ECMP next-hops, it may be advantageous to move as few flows as possible when removing or adding members to the ECMP set; for example, when the ECMP next-hops of an IP route correspond to network appliances or host servers that maintain state for the flows they service, and moving flows requires state to be rebuilt. To do this, SR Linux supports resilient hashing, which can minimize hash bucket reassignment during changes to the ECMP set.

Because resilient hashing consumes extra datapath resources, no IP ECMP routes are programmed for resilient hashing by default. To enable resilient hashing for an IP ECMP route, the prefix of the route must be matched by an entry configured in a resilient-hash-prefix list. A resilient-hash-prefix list entry specifies two parameters: the number of hash buckets per path (**hash-buckets-per-path** value) and the maximum number of paths (**max-paths** value). Together, these parameters determine a hash-bucket fill pattern for the matching prefix. The **hash-buckets-per-path** is the number of times each next-hop is repeated in the fill pattern if there are **max-paths** ECMP next-hops

For example, if the **hash-buckets-per-path** value is 4 and the **max-paths** value is 6, then the fill pattern has 24 hash buckets, offering space for 6 next-hops, each repeated 4 times. A route matching this resilient-hash-prefix list entry is initially programmed with 4 next-hops. The initial programming of its next-hop-group looks like this:

[nh1, nh2, nh3, nh4, (nh1), (nh1)],

[nh1, nh2, nh3, nh4, (nh2), (nh2)],

[nh1, nh2, nh3, nh4, (nh3), (nh3)],

[nh1, nh2, nh3, nh4, (nh4), (nh4)]

Next-hops in parentheses represent repeated next-hops that are leaving space for future next-hop additions.

If a new ECMP next-hop nh5 is added to the route, then the fill pattern becomes the following:

[nh1, nh2, nh3, nh4, nh5, (nh1)],

[nh1, nh2, nh3, nh4, nh5, (nh2)],

[nh1, nh2, nh3, nh4, nh5, (nh3)],

[nh1, nh2, nh3, nh4, nh5, (nh4)]

If instead of adding an ECMP next-hop, the existing ECMP next-hop nh4 is deleted, then the fill pattern becomes the following:

[nh1, nh2, nh3, (nh1), (nh1), (nh1)],

[nh1, nh2, nh3, (nh2), (nh2), (nh2)],

[nh1, nh2, nh3, (nh3), (nh3), (nh3)],

[nh1, nh2, nh3, (nh1), (nh2), (nh3)]

The resilient hashing algorithm does not guarantee that the traffic share of each ECMP member is equally affected by a next-hop addition or removal. The example above shows that when the route goes from 3 to 4 ECMP next-hops, the existing nh1 next-hop loses two hash buckets (out of 24), while the existing nh2 and nh3 next-hops lose only one hash bucket. The resilient hashing algorithm is focused on minimizing hash bucket reassignment throughout the changes.

On both 7250 IXR and 7220 IXR systems, the product of the **hash-buckets-per-path** value and **max-paths** value must be less-than or equal-to a platform-defined limit of 128.

Note: If a route is covered by a resilient-hash-prefix list entry, but it has only a single next-hop (non ECMP), the `resilient` flag displays as `true` in the route-table state, even though multiple hash buckets are not allocated.

Resilient hashing interaction with weighted ECMP

When a BGP, IS-IS, or gRIBI route is eligible for both weighted ECMP and resilient hashing, the resilient hashing configuration overrides the weighted ECMP configuration. For example, if **max-ecmp-hash-buckets-per-next-hop-group** is 256 and the **resilent-hash-prefix** configuration for the prefix specifies a `max-paths` value of 4 and **hash-buckets-per-path** of 4, then 16 hash buckets are used for this prefix and the number of ECMP members is capped at 4.

8.3.1 Configuring resilient hashing

Procedure

To configure an IP ECMP route for resilient hashing, you configure the route prefix in a resilient-hash-prefix list and specify the number of hash buckets per path and the maximum number of ECMP next-hop paths per route.

Example

The following example configures an entry in a resilient-hash-prefix list. Active routes in the FIB that exactly match this prefix or are longer matches of this prefix are provided with resilient-hash programming. The **hash-buckets-per-path** value is 4 and the **max-paths** value is 6, so the fill pattern has 24 hash buckets, offering space for 6 next-hops, each repeated 4 times.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default ip-load-balancing resilient-hash-prefix *
    network-instance default {
        ip-load-balancing {
            resilient-hash-prefix 10.10.10.0/24 {
                hash-buckets-per-path 4
                max-paths 6
            }
        }
    }
```

8.4 Dynamic Load Balancing

7220 IXR-H*x* platforms support dynamic load balancing for ECMP distribution of packets over outgoing links. Dynamic load balancing improves on hash-based load balancing by considering the state of aggregate ECMP group members when assigning flows to groups.

At packet ingress, a flow is identified, and the state of the flow is evaluated. Based on this evaluation, the flow is assigned to an aggregate ECMP group. The dynamic load balancing algorithm analyzes the aggregate ECMP groups and detects ECMP load imbalances among the egress paths based on three load-balancing factors: egress port utilization, port queue fill size, and Ingress Traffic Manager (ITM) port queue size. When the algorithm detects an ECMP load imbalance, it can reassign flows to different aggregate ECMP groups so that balance is restored.

Dynamic load balancing is enabled for specific prefixes in a network-instance. The following options are configurable at the `system` level, which applies them to all dynamic load balancing ECMP groups.

- flowset-size: the number of flow entries reserved for each aggregate ECMP group.
- inactivity-timer: the amount of time a flow must be idle before it is eligible for reassignment to a different aggregate ECMP group member.
- mode: whether the system can reassign a flow to a different aggregate ECMP group (flow-dynamic) after the initial assignment when the flow is inactive, whether the flows do not move after the initial assignment (flow-fixed), or whether the load balancing is done on a per-packet basis, distributing them across all active next-hops.
- link-quality-sampling-interval: how often the system samples the link quality.

Dynamic load balancing is supported for physical interfaces only; it is not supported for LAG interfaces. Dynamic load balancing is limited to unicast traffic only.

8.4.1 Configuring dynamic load balancing

Procedure

You can configure options to adjust how the dynamic load balancing algorithm balances traffic, the network prefixes for which traffic is load balanced, and thresholds for monitoring resources used by dynamic load balancing.

Example: Configure system options for dynamic load balancing

The following example configures system-level parameters to adjust the dynamic load balancing algorithm.

```
--{ +* candidate shared default }--[  ]--
# info with-context system load-balancing dynamic
    system {
        load-balancing {
            dynamic {
                flowset-size 512
                inactivity-timer 100
                link-quality-sampling-interval 7
                weighting-factor {
                    port-utilization 70
                    queue-utilization 20
                    itm-utilization 10
                }
            }
        }
    }
```

In this example, a flow must be inactive for a minimum of 100 microseconds to be moved to a better quality interface. The quality of the link is evaluated at an interval of 7 microseconds. The dynamic load balancing algorithm is configured so that port-utilization is weighted heaviest.

Example: Configure the mode for dynamic load balancing

By default, dynamic load balancing assigns a flow to a different aggregate ECMP group if the flow is inactive following the initial assignment. You can optionally configure the mode for dynamic load balancing to either **per-packet** or **flow-dynamic**. In either mode, the system considers the port load quality and chooses the best quality path. In **per-packet** mode each packet is assigned to a best quality path. In **flow-dynamic** mode, the entire flow is assigned to a best quality path.

The following example sets the dynamic load balancing mode to `per-packet`.

```
--{ * candidate shared default }--[  ]--
# info with-context system load-balancing dynamic
    system {
        load-balancing {
            dynamic {
                mode per-packet
            }
        }
    }
```

Example: Configure network prefixes for dynamic load balancing

Dynamic load balancing is enabled on an IP prefix-by-prefix basis within a network-instance. Routes matching the specified prefix have dynamic load balancing enabled on their associated ECMP next-hop group.

The following example enables dynamic load balancing for a prefix within the default network-instance.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default ip-load-balancing dynamic-load-balancing
    network-instance default {
        ip-load-balancing {
            dynamic-load-balancing {
                prefix 10.101.12.0/24 {
                }
            }
        }
    }
```

Example: Configure resource monitoring thresholds for dynamic load balancing

The following example configures the system to generate a warning message when the usage level for dynamic load balancing ECMP groups exceeds a threshold, and a notice message when the usage level for dynamic load balancing ECMP groups drops below a threshold.

```
--{ +* candidate shared default }--[  ]--
# info with-context platform resource-monitoring datapath asic resource dynamic-load-
balancing-ecmp-groups
    platform {
        resource-monitoring {
            datapath {
                asic {
                    resource dynamic-load-balancing-ecmp-groups {
                        upper-threshold-set 75
                        upper-threshold-clear 50
                    }
                }
            }
        }
    }
```

In this example, a warning message is generated and `used-upper-threshold-exceeded` of the datapath resource is set to `true` whenever utilization of the resource in any line card, forwarding complex, or pipeline reaches 75% in a rising direction. A notice message is generated and `used-upper-threshold-exceeded` is set to `false` whenever utilization of the datapath resource in any line card, forwarding complex, or pipeline reaches 50% in a falling direction.

Example: Display resource usage for dynamic load balancing ECMP groups

The following example shows the current resource usage for dynamic load balancing ECMP groups in the system:

```
--{ running }--[  ]--
# info with-context from state platform linecard 1 forwarding-complex 0 datapath asic
 resource dynamic-load-balancing-ecmp-groups
    platform {
        linecard 1 {
            forwarding-complex 0 {
                datapath {
                    asic {
                        resource dynamic-load-balancing-ecmp-groups {
                            used-percent 100
                            used-entries 128
                            free-entries 0
                        }
                    }
                }
            }
        }
    }
```

Example: Display the port quality for the ECMP path

The following example displays the current port quality metrics of the ECMP path. In this example, two metrics are displayed for each ITM. Single ITM platforms display a metric value of 0.

```
--{ running }--[  ]--
# info with-context from state interface ethernet-1/1 load-balancing
    interface ethernet-1/1 {
        load-balancing {
            last-dynamic-load-balancing-quality-metrics [
                7
                7
            ]
        }
    }
```

Example: Verify that dynamic load balancing is enabled for a prefix

It is possible that dynamic load balancing may become disabled for a prefix because of exhaustion of resources such as flowset size or dynamic load balancing ECMP group limit. The following example displays whether dynamic load balancing is enabled for a prefix. In the example, `dynamic-load-balancing true` indicates the prefix is enabled for dynamic load balancing.

```
--{ running }--[  ]--
# info from state network-instance default route-table ipv4-unicast route 192.2.1.0/24 id
 0 route-type static route-owner static_route_mgr origin-network-instance default
    leakable false
    leaked false
    metric 1
    preference 5
    active true
    last-app-update "2025-10-22T19:23:39.036Z (25 minutes ago)"
    next-hop-group 6181644
    next-hop-group-network-instance default
    resilient-hash false
    dynamic-load-balancing true
    fib-programming {
        suppressed false
        last-successful-operation-type add
        last-successful-operation-timestamp "2025-10-22T19:23:39.065Z (25 minutes ago)"
        pending-operation-type none
        last-failed-operation-type none
    }
```

In the following example, `requested true` indicates dynamic load balancing was configured for the prefix in the network-instance, and `enabled true` indicates that the prefix is programmed for dynamic load balancing. The `flows-rebalanced` statistic indicates the number of flows that were rebalanced from the original ECMP path assignment.

```
--{ + running }--[  ]--
# info from state network-instance default route-table next-hop-group 6181644
    backup-next-hop-group 0
    backup-active false
    dynamic-load-balancing {
        requested true
        enabled true
        flows-rebalanced 1290184
```

9 Protocol authentication

In SR Linux, authentication of routing control messages for BGP, as well as other protocols such as LDP and IS-IS, is done using shared keys.

Message authentication between two routers involves sharing knowledge of a secret key and a cryptographic algorithm, such as MD5. This secret key, together with the message data, are used to generate a message digest. The message digest is added to each message transmitted by the sender and validated by the receiver, with the expectation that only a sender in possession of the secret key and algorithm details could generate the same message digest computed by the receiver of the message.

To limit exposure in the event a key is compromised, the secret key is changed at regular intervals using keys configured in a keychain. A keychain defines a list of one or more keys; each key is associated with a secret string, an algorithm identifier, and a start time.

When a protocol references a keychain for securing its messages with a set of peers, it uses the active key in the keychain with the most recent start time to generate the message digest in its sent messages, and it drops every received message that does not have an acceptable message digest.

9.1 Configuring protocol authentication

Procedure

To configure protocol authentication, you configure an authentication keychain at the system level and configure a protocol to use the keychain. All protocol authentication is done using keychains. If a protocol requires authentication with a single neighbor using a single key, the key is configured within a keychain, and the protocol references the keychain.

Example: Configure a keychain

The following example configures a keychain consisting of two keys.

```
--{ candidate shared default }--[  ]--
# info with-context system authentication
    system {
        authentication {
            keychain k1 {
                key 1 {
                    admin-state enable
                    algorithm md5
                    authentication-key ZcvSElJzJx/wBZ9biCt
                }
                key 2 {
                    admin-state enable
                    algorithm md5
                    authentication-key e7xdKlYO2DOm7v3IJv
                }
            }
        }
```

Example: Configure BGP to use the keychain for protocol authentication

The following example configures BGP to use the keys in the keychain above for protocol authentication:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp authentication
    network-instance default {
        protocols {
            bgp {
                authentication {
                    keychain k1
                    }
                }
            }
        }
    }
```

Example: Configure BGP to use a password without a keychain for protocol authentication

SR Linux supports configuring passwords without a keychain for authentication between BGP peers. You must configure the authentication with the same password on both BGP peers. Otherwise, the connection between them cannot be established .

The following example configures BGP to use a password without a keychain for protocol authentication:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp authentication
    network-instance default {
        protocols {
            bgp {
                authentication {
                     password $aes$0K0SDPvDqXCU=$Cny0JELb0jmt8crznXsYzQ==
                    }
                }
            }
        }
    }
```

The following example illustrates the authentication password configuration at group level:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp group Dut-C--Dut-D
 authentication
    network-instance default {
        protocols {
            bgp {
                group Dut-C--Dut-D {
                    authentication {
                        password $aes$9GP/wMALbt+c=$C59vOvqPux/Ue5bTRvVzfQ==
                    }
                }
            }
        }
    }
```

The following example illustrates the authentication password configuration at neighbor level:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 3.4.0.1 authentication

    network-instance default {
        protocols {
            bgp {
                neighbor 3.4.0.1 {
                    authentication {
                        password $aes$cJHWoYWwJVVU=$MOmvTOs2av5CGxBPdjVXuQ==
                    }
                }
            }
        }
    }
```

10 Routing policies

Routing policies allow detailed control of IP routes learned and advertised by routing protocols such as BGP, IS-IS, or OSPF.

Each routing policy has a sequence of rules (called entries or statements) and a default action. Each policy statement has zero or more match conditions, such as whether a route is owned by a specific protocol, and a main action, such as to accept or reject the routes; the statement may also have route-modifying actions. A route matches a statement if it meets all of the specified match conditions.

Each statement has a specified position in the policy. If a policy has multiple statements, they are processed in the order they are put in the policy. In addition, you can specify a default action that applies to routes that do not match any statement in the policy.

The first statement that matches the route determines the actions that are applied to the route. If the route is not matched by any statements, the default action of the policy is applied. If there is no default action, then a protocol- and context-specific default action is applied.

All routes match a statement with no match conditions. When a route fulfills the match conditions of a statement, the base action of the statement is applied, along with all of its route-modifying actions.

10.1 Routing policy match conditions

The following table summarizes the match conditions supported in SR Linux routing policies.

Table 12: Match conditions for routing policies

Match condition	Description
`protocol`	Test if the route is owned by a specific protocol. The following options are supported: • aggregate • arp-nd • bgp • bgp-evpn • bgp-evpn-ifl-host • bgp-ipvpn • bgp-label • dhcp • host • isis • local • ospfv2

Match condition	Description
	- ospfv3 - static
`family`	Test if the prefix is associated with any of the AFI/SAFI (address families) listed. The following options are supported: - ipv4-unicast - ipv6-unicast - l3vpn-ipv4-unicast - l3vpn-ipv6-unicast - ipv4-labeled-unicast - ipv6-labeled-unicast - evpn - route-target - srte-policy-ipv4 - srte-policy-ipv6 - link-state
`call-policy`	Call another routing policy as a subroutine. See Routing policy subroutines.
`prefix.prefix-set`	Test if the route matches any entry in a configured prefix-set. See Prefix sets.
`internal-tags.tag-set`	Test if the route contains members of a configured tag-set. See "Configuring route internal tags" in the *SR Linux VPN Services Guide*.
`bgp.as-path.as-path-set`	Test if the route has an AS path that matches the regular expression in the configured AS-path set. See AS path sets.
`bgp.as-path-length`	Test if the AS-path length of the route is a specific value or in a range.
`bgp.community-set`	Test if a BGP community attached to the route matches the list of member elements and match-set-option in the BGP community-set. See BGP community-sets.
`bgp.evpn.route-type`	Test if the EVPN route type is a specific type (1-8).
`bgp.extended-community.extended-community-set`	Test if a BGP extended community attached to the route matches the list of member elements and match-set-option in the BGP extended community-set. See BGP community-sets.

Match condition	Description
`bgp.standard-community.standard-community-set`	Test if a BGP standard community attached to the route matches the list of member elements and match-set-option in the BGP standard community-set. See BGP community-sets.
`isis.level`	Test if the IS-IS route is associated with Level 1 or Level 2.
`isis.route-type`	Test if the IS-IS route is internal or external (redistributed from another protocol). An IPv4 prefix is external if it is signaled in TLV 130 or TLV135 with RFC 7794 X flag=1. An IPv6 prefix is external if TLV 236/TLV 237 external bit = 1.
`ospf.area-id`	Test if the OSPFv2 or OSPFv3 route is associated with the specified area ID.
`ospf.route-type`	Test if the OSPFv2 or OSPFV3 route is a specific route type: any of intra-area, inter-area, type-1-ext, type-2-ext, type-1-nssa, type-2-nssa, summary-aggregate, or nssa-aggregate.
`ospf.instance-id`	Test if the OSPFv2 or OSPFv3 route is associated with the specified instance ID.
`multicast.group-address.prefix-set`	Test if the multicast group address matches any entry in a configured prefix-set. See Prefix sets.
`multicast.source-address.prefix-set`	Test if the multicast source address matches any entry in a configured prefix-set. See Prefix sets.
`origin-network-instance`	The source network-instance where the route was originally learned or configured. If this match condition is omitted from a policy statement, the implied match is any network-instance.
`network-instance-leaked-route`	When set to `true`, this condition matches all leaked routes (that is, routes with an origin network-instance different from the local-context network-instance). When set to `false`, this condition matches all non-leaked routes (that is, with an origin network-instance the same as the local-context network-instance).

10.1.1 Specifying match conditions in a routing policy

Procedure

You can specify the match conditions listed in Table 12: Match conditions for routing policies in a policy statement. If a statement has no match conditions defined, all routes evaluated by the policy are considered to be matches.

Example: Test if the route is owned by BGP

The following example specifies BGP protocol as a match condition in a policy statement:

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    protocol bgp
                }
            }
        }
    }
```

Example: Test if the route prefix is associated with an address family

The following example matches routes with a prefix belonging to the IPv4 or IPv6 labeled unicast address family:

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy policy policy02
    routing-policy {
        policy policy02 {
            statement 100 {
                match {
                    family [
                        ipv4-labeled-unicast
                        ipv6-labeled-unicast
                    ]
                }
            }
        }
    }
```

Example: Test if an OSPF route is a specific route type

The following example matches OSPF type 2 external routes:

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy policy policy03
    routing-policy {
        policy policy03 {
            statement 100 {
                match {
                    protocol ospfv2
                    ospf {
                        route-type type-2-ext
                    }
                }
            }
        }
    }
```

10.1.2 Routing policy subroutines

A match condition in a routing policy can call another routing policy as a subroutine.

SR Linux supports up to three levels of policy subroutines. For example, a main policy P_0 can call a P_1 policy. The P_1 policy can call a P_2 policy, and the P_2 policy can call a P_3 policy.

A route that is accepted by a P_n subroutine policy matches the P_{n-1} policy statement that called policy P_n.

The subroutine policies (P_1-P_3) can modify the attributes of matched routes. These modifications can happen in statements with action `accept` or action `next-statement` or where the default-action is `accept`.

The subroutine policy match can be combined with other match conditions in the same statement of a policy. Logically, the subroutine is evaluated after all other match conditions in the statement.

If the subroutine policy has a statement with action `next-policy` or default-action `next-policy`, it is considered the same as `reject`.

If a route that does not match any terminating entry of a subroutine (that is, the route does not match any statement with an action of `accept` or `reject` or `next-policy`), it is considered to be accepted by the subroutine if the `default-action` of the subroutine policy is `accept`. Otherwise, the route is considered to be rejected by the subroutine.

The following example shows a routing policy P1 that uses policy P2 as a subroutine.

```
--{ +* candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        policy P1 {
            statement 10 {
                match {
                    call-policy [
                        P2
                    ]
                    bgp {
                        as-path {
                            as-path-set downstream
                        }
                    }
                }
                action {
                    policy-result accept
                }
            }
        }
        policy p2 {
            statement 10 {
                match {
                    prefix {
                        prefix-set selected
                    }
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

10.2 Routing policy actions

If a route matches a policy statement, the actions configured in the policy are applied to the route, including route property modifications. Depending on the configured action, the route may continue to be analyzed by further routing policy statements (if any), or the final disposition (either accept or reject) of the route is determined.

Default routing policy action

Each routing policy can optionally be configured with a default action, which is applied to routes that do not match any policy statement. The default action specifies a main action and optionally one or more route property modification actions. The default action is applied to all routes evaluated by the policy that do not match any statement or are only matched by statements that are non-terminating, such as `next-statement`.

If the default action is explicitly configured as `accept` or `reject`, evaluation ends at the current policy even if there are further policies in the chain. If a policy does not have a configured default action, or the default action for the policy is `next-policy` or `next-statement`, then routes that do not match any statement in the policy are automatically evaluated by the next policy in the chained list. See Specifying a default action for a configuration example.

Main routing policy actions

The following table lists the main routing policy actions. These actions can be configured for the `policy-result` of the default action or at the statement action level in a routing policy.

Table 13: Main routing policy actions

Action	Description
`accept`	The route is accepted, route property modifications are applied, and evaluation stops immediately; further statements and policies are not considered.
`reject`	The route is dropped and evaluation stops immediately; further statements and policies are not considered.
`next-statement`	No immediate decision is made about whether to accept or reject the route. Route policy modifications are applied, and evaluation continues automatically to the next statement. If there are no further statements in this policy, the default action of the current policy applies.
`next-policy`	No immediate decision is made about whether to accept or reject the route. Route policy modifications are applied, and evaluation skips the remainder of the statements in the current policy continuing automatically to the first statement in the next policy in the chained list. If there are no further policies, then the default action of the last policy in the chain applies.

If no value is configured for `policy-result`, the implicit default is `next-statement` (if not configured at the statement action level) or `next-policy` (if not configured at the default-action level).

Route property modification actions

The following table lists the route property modification actions that can be specified in a routing policy.

Table 14: Route property modification actions

Action	Description
`bgp.as-path.prepend`	Prepend a specific AS number one or more times to the AS path.
`bgp.as-path.remove`	Clear the AS path to make it empty.
`bgp.as-path.replace`	Clear the existing AS path and replace it with a new AS_SEQUENCE containing the listed AS numbers.
`bgp.communities.add`	Add all of the non-regex community members in the referenced community-set. **Note:** New communities are prepended before existing communities, as shown in BGP RIB state information and in the advertised UPDATE.
`bgp.communities.remove`	Remove all communities that match any regex or non-regex member in the referenced community-set.
`bgp.communities.replace`	Delete all existing communities and add all non-regex community members in the referenced community-set.
`bgp.disable-ip-route-install`	Accept the route, allowing its re-advertisement, but do not install the route to the IP FIB. See Policy action to disable IP FIB installation.
`bgp.extended-community.method`	Indicate the method used to specify the extended communities for the action. The only supported value is `reference`.
`bgp.extended-community.operation`	Add, remove, or replace extended communities in matched routes. The `add` operation adds every non-regex member of every referenced set to the EXT_COMMUNITIES attribute of the matched routes (after de-duplication). The `remove` operation removes every extended community that matches any regex or non-regex member in any referenced set. The `replace` operation removes every existing extended community, then adds every non-regex

Action	Description
	member of every referenced set to the EXT_COMMUNITIES attribute of the matched route (after de-duplication). Note that new communities are prepended before existing communities, as shown in BGP RIB state information and in the advertised UPDATE.
`bgp.extended-community.referenced-sets`	Indicate the extended community sets to perform the add, remove, or replace operation.
`bgp.standard-community.method`	Indicate the method used to specify the standard communities for the action. The only supported value is `reference`.
`bgp.standard-community.operation`	Add, remove, or replace standard communities in matched routes. The `add` operation adds every non-regex member of every referenced set to the COMMUNITIES attribute of the matched routes (after de-duplication). The `remove` operation removes every standard community that matches any regex or non-regex member in any referenced set. The `replace` operation removes every existing standard community, then adds every non-regex member of every referenced set to the COMMUNITIES attribute of the matched route (after de-duplication). **Note:** New communities are prepended before existing communities, as shown in BGP RIB state information and in the advertised UPDATE.
`bgp.standard-community.referenced-sets`	Indicate the standard community sets to perform the add, remove, or replace operation.
`bgp.label-allocation.prefix-sid.use-igp`	Use the programmed SR-IGP label index for the matching prefix, resulting in a stitch to the IGP segment routing tunnel.
`bgp.local-preference.operation`	Specify the operation to use when applying the configured value to the current LOCAL_PREF; the only valid operation is `set`.
`bgp.local-preference.value`	Set the value of the LOCAL_PREF attribute in matching BGP routes.
`bgp.med.operation`	Set or modify the MED according to the `bgp.med.value` action in matching BGP routes.

Action	Description
	See Route property operation action for setting MED.
`bgp.med.value`	Set or modify the MED to the specified value in matching BGP routes. See Policy actions for setting MED in BGP routes.
`bgp.next-hop-resolution.set-tag-set`	Reference a tag-set to be used for controlling next-hop resolution.
`bgp.next-hop.set`	Set or modify the BGP next-hop address to a specified IPv4 or IPv6 address or to the keyword `self`.
`bgp.origin.set`	Set or modify the BGP ORIGIN attribute value to the specified value.
`isis.level`	Add the redistributed route to the specified level database. Supported values are 1 or 2.
`isis.metric.set-style`	Explicitly encode the metric of the matched IS-IS route as a wide metric in the appropriate IS-IS TLV that supports wide metrics.
`isis.metric.set-value`	Set the IS-IS metric value of the redistributed route to a number between 1 and 16777215.
`internal-tags.tag-set`	Add the tags in the tag-set as internal tags.
`ospf.metric.set-value`	Set the OSPF metric value of the redistributed route to a uint16 number.
`ospf.metric.set-external-type`	Set the OSPF external metric type of the redistributed route as type-1 or type-2.
`route-preference.set`	Overwrite the route preference with the specified value.

10.2.1 Specifying actions in a routing policy

Procedure

You can specify the actions listed in Routing policy actions to routes that match a policy statement.

Example: Accept static routes

The following example configures a routing policy statement to accept static routes.

```
--{ +* candidate shared default }--[  ]--
# info with-context routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement st1 {
                match {
                    protocol static
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

Example: Set origin attribute for matching BGP routes

The following example specifies a policy with two statements. If a BGP route matches the first statement, the action is to proceed to the next statement. If the route matches the second statement, the action is to specify a new value for the origin attribute of the BGP route.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        policy policy02 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    policy-result next-statement
                }
            }
            statement 101 {
                match {
                    prefix-set western
                }
                action {
                    bgp {
                        origin {
                            set egp
                        }
                    }
                }
            }
        }
    }
```

Example: Add communities from BGP standard-community set to matched routes

The following example matches BGP routes and adds every non-regex member of standard-community set st1 to the COMMUNITIES attribute of the matched routes.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        standard-community-set st1 {
            member [
                668$
                ^65100
            ]
        }
        policy p1 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    bgp {
                        standard-community {
                            operation add
                            referenced-sets [
                                st1
                            ]
                        }
                    }
                }
            }
        }
    }
```

Example: Replace communities in matched routes with communities in a BGP extended-community set

The following example removes every existing extended community from matching BGP routes, then adds every non-regex member of extended-community set ex1 to the EXT_COMMUNITIES attribute of the matched routes.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        extended-community-set ex1 {
            member [
                target:10.1.1.1:.*
            ]
        }
        policy p2 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    bgp {
                        extended-community {
                            operation replace
                            referenced-sets [
                                ex1
                            ]
                        }
                    }
                }
            }
        }
    }
```

10.2.2 Specifying a default action

Procedure

You can optionally specify the policy action for routes that do not match the conditions defined in the policy statements. The default action can be set to all available action states including accept, reject, next-entry, and next-policy.

- If a default action is defined, and no matches occur with the statements in the policy, the default action is used.
- If no default action is specified, the default behavior of the protocol controls whether the routes match.

For BGP, the default import action is to accept all routes from BGP. For internal routes, the default export action is to advertise them to BGP peers. For external routes, the default export action is not to advertise them to BGP peers.

Example

The following example defines a policy where the default action for non-matching routes is to reject them:

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        policy policy01 {
            default-action {
                policy-result reject
            }
        }
    }
```

10.2.3 Policy actions for setting MED in BGP routes

The following describes the supported policy actions for received BGP routes based on the contents of the MED attribute in the route (if any).

When a BGP export policy matches a non-BGP route for advertisement to neighbors:

- If the route is matched by an export policy statement that applies the action **bgp med set route-table-cost**, then a MED attribute is added to the BGP route and its value is the route-table metric for the non-BGP route.
- If the route is matched by an export policy statement that applies the action **bgp med set <number>**, then a MED attribute is added to the BGP route and its value is **<number>**.
- If neither case applies, no MED is added to the route.

When a BGP route is received with a MED attribute:

- If the route is matched by an import policy statement that applies the action **bgp med set route-table-cost**, then the MED attribute is replaced and the new value is the route-table metric of the route that resolves the BGP next-hop.
- If the route is matched by an import policy statement that applies the action **bgp med set <number>**, then the MED attribute is replaced and the new value is **<number>**.
- If neither case applies, the RIB-IN MED attribute is unmodified.

When a BGP route is received without a MED attribute:

- If the route is matched by an import policy statement that applies the action **bgp med set route-table-cost**, then a MED attribute is added to the BGP route and the new value is the route-table metric of the route that resolves the BGP next-hop.
- If the route is matched by an import policy statement that applies the action **bgp med set <number>**, then a MED attribute is added to the BGP route and its value is **<number>**.
- If neither case applies, no MED attribute is added to the route.

When a BGP route is received with a MED attribute from one eBGP peer, and the route must be advertised to other eBGP peers:

- If the route is matched by an export policy statement that applies the action **bgp med set route-table-cost**, then the MED attribute is replaced and the new value is the route-table metric of the route that resolves the BGP next-hop (0 when receiving a route from a single hop eBGP peer)
- If the route is matched by an export policy statement that applies the action **bgp med set <number>**, then the MED attribute is replaced and the new value is **<number>**.
- If neither case applies, the MED is stripped.

When a BGP route is received without a MED attribute from one eBGP peer, and the route must be advertised to other eBGP peers:

- If the route is matched by an export policy statement that applies the action **bgp med set route-table-cost**, then a MED attribute is added to the BGP route and its value is the route-table metric of the route that resolves the BGP next-hop (0 when receiving a route from a single hop eBGP peer).
- If the route is matched by an export policy statement that applies the action **bgp med set <number>**, then a MED attribute is added to the BGP route and its value is **<number>**.
- If neither case applies, no MED is added to the route.

When a BGP route is received with a MED attribute from one eBGP peer, and the route must be advertised to iBGP peers, or else the route was received with a MED attribute from an iBGP peer for propagation to other iBGP peers:

- If the route is matched by an export policy statement that applies the action **bgp med set route-table-cost**, then the MED attribute is replaced and the new value is the route-table metric of the route that resolves the BGP next-hop.
- If the route is matched by an export policy statement that applies the action **bgp med set <number>**, then the MED attribute is replaced and the new value is **<number>**.
- If neither case applies, the MED is unmodified.

When a BGP route is received with a MED attribute from one eBGP/iBGP peer, and the route must be advertised to other eBGP peers, the MED is stripped by default (if the route is not matched by a policy).

When a BGP route is received without a MED attribute from one eBGP peer, and the route must be advertised to iBGP peers, or else the route was received without a MED attribute from an iBGP peer for propagation to other eBGP or iBGP peers:

- If the route is matched by an export policy statement that applies the action **bgp med set route-table-cost**, then a MED attribute is added to the BGP route and its value is the route-table metric of the route that resolves the BGP next-hop.
- If the route is matched by an export policy statement that applies the action **bgp med set <number>**, then a MED attribute is added to the BGP route and its value is **<number>**.
- If neither case applies, no MED is added to the route.

Route property operation action for setting MED

The `bgp.med.operation` action can set or modify the MED according to the `bgp.med.value` action in matching BGP routes, as described in the following table:

Table 15: Operations for setting BGP MED in routing policies

Operation	Value	Result
`set`	Unsigned 32-bit integer	Overwrite the current MED with the specified value.

Operation	Value	Result
	`route-table-cost`	Overwrite the current MED with the metric of the resolving route.
`add`	Unsigned 32-bit integer	Increment the current MED by the specified value.
	`route-table-cost`	Increment the current MED by the metric of the resolving route.
`subtract`	Unsigned 32-bit integer	Decrement the current MED by the specified value.
	`route-table-cost`	Decrement the current MED by the metric of the resolving route.

10.2.4 Policy action to disable IP FIB installation

Some data center switches have limited IP FIB capacity. It is possible these switches could receive more BGP routes than they have room for in their IP FIB tables. In such cases, you may want to suppress the IP FIB installation of some or all of the BGP routes the switch does not require for forwarding (for example, because the switch is not in the forwarding path, as in the case of a non-next-hop-self route reflector, or because the switch can fall back to less specific routes, such as a default route).

Matching and rejecting a BGP route in a BGP import policy prevents the route from being installed in the IP FIB, but it also prevents the route from being re-advertised to other BGP peers, which could be unacceptable. For these cases, you can configure the `disable-ip-route-install` policy action. This policy action works as follows:

For unlabeled IPv4 and IPv6 routes matched and accepted by a BGP import policy with a `disable-ip-route-install` action:

- Routes are re-advertisable to other peers per normal BGP rules, even if `advertise-inactive` is not configured or `next-hop-self` is enabled.
- Routes are less preferred by the BGP decision process than any installed route for the same prefix. For these routes, `fib-install-disabled` is displayed as the `tie-break-reason` in state information.
- Routes do not contribute toward and activate a covering aggregate route.

For unlabeled IPv4 and IPv6 routes matched and accepted by a BGP import policy with a disable-ip-route-install action:

- Host routes do not create/program tunnels.
- Re-advertisement with next-hop self does create/program ILM entries.
- Routes are re-advertisable to other peers per normal BGP rules, even if `advertise-inactive` is not configured or `next-hop-self` is enabled.
- Routes are less preferred by the BGP decision process than any installed route for the same prefix. For these routes, `fib-install-disabled` is displayed as the `tie-break-reason` in state information.
- Routes do not contribute toward and activate a covering aggregate route.

BGP does not publish routes to fib_mgr that have been matched and accepted by a BGP import policy with a `disable-ip-route-install` action. This means they are not be re-advertisable by other protocols.

In BGP RIB state information, FIB-suppressed routes display `used-route false` and `fib-disabled true`.

10.3 Applying a routing policy

Procedure

Routing policies can be applied to routes received from other routers (imported routes), as well as routes advertised to other routers (exported routes). Routing policies can be applied at the network-instance level, peer-group level, and neighbor level.

Example: Apply a routing policy to imported routes

The following example specifies that BGP in the default network-instance applies `policy01` to imported routes:

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp import-policy
    network-instance default {
        protocols {
            bgp {
                import-policy [
                    policy01
                ]
            }
        }
    }
```

Example: Apply a routing policy to BGP routes exported from a peer-group

The following example applies `policy02` to BGP routes exported from the peers in peer-group `headquarters1`:

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols bgp group headquarters1
    network-instance default {
        protocols {
            bgp {
                group headquarters1 {
                    export-policy [
                        policy02
                    ]
                }
            }
        }
    }
```

Example: Apply a routing policy to BGP routes exported from a BGP neighbor

The following example applies `policy02` to BGP routes exported from a specific BGP neighbor:

```
--{ * candidate shared default }--[  ]--
# info  network-instance default protocols bgp neighbor 192.168.11.1
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    export-policy [
                        policy02
                    ]
                }
            }
        }
    }
```

10.3.1 Applying a default policy to eBGP sessions

Procedure

You can specify the action to take for routes exported to or imported from eBGP peers to which no configured policy applies. This is set with the **ebgp-default-policy** command and the **export-reject-all** and **import-reject-all** parameters.

- The **export-reject-all** parameter, when set to **true**, causes all outbound routes that do not match a configured export policy to be rejected as if they had been rejected by a configured export policy. The default is **true**.
- The **import-reject-all** parameter, when set to **true**, causes all inbound routes that do not match a configured import policy to be rejected as if they had been rejected by a configured import policy. The default is **true**.

Example

The following example allows a BGP neighbor to export a default route even though the route is not subject to any configured policy:

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        protocols {
            bgp {
                ebgp-default-policy {
                    export-reject-all false
                }
                neighbor 2001:db8::c11 {
                    send-default-route {
                        ipv6-unicast true
                    }
                }
            }
        }
    }
```

10.3.2 Replacing an AS path

Procedure

You can configure a routing policy where the AS path in matching routes is replaced by a list of AS numbers specified in the policy. For routes that match the policy, the current AS path is deleted and replaced with an AS_SEQ element containing the AS numbers listed in the policy in their configured sequence.

If you configure an empty AS list in the policy, then the current AS path in a matching route is deleted, and it would then have a null AS_PATH attribute.

Example

The following is an example of a routing policy that matches BGP routes. The action for the policy is `next-policy`, so routes that match the first policy are evaluated by the second policy, whose action is to replace the AS path in matching routes.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    policy-result next-policy
                }
            }
        }
        policy policy02 {
            statement 100 {
                match {
                    prefix-set western
                }
                action {
                    bgp {
                        as-path {
                            replace [
                                12
                                13
                                14
                            ]
                        }
                    }
                }
            }
        }
    }
```

10.4 Resequencing statements in a routing policy

Procedure

Each routing policy statement has a numerical sequence identifier that determines its order relative to other statements in that policy. When a route is analyzed by a policy, it is evaluated by each statement in sequential order.

The definition of the sequence of statements is fully flexible and can be defined as required using the **insert** command. If the **insert** command is not used, new statements are added to the end of the policy.

In the following examples, routing policy policy01 consists of three statements: red, green, and blue.

```
--{ +* candidate shared default }--[  ]--
# info with-context routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement red {
                match {
                    protocol bgp
                }
                action {
                    policy-result accept
                }
            }
            statement green {
                match {
                    prefix-set pset1
                }
                action {
                    policy-result reject
                }
            }
            statement blue {
                match {
                    family [
                        ipv4-unicast
                        ipv6-unicast
                    ]
                }
                action {
                    policy-result next-policy
                }
            }
        }
    }
```

Example: Move a statement to the end of a routing policy

The following example moves statement red to the end of the routing policy:

```
--{ +* candidate shared default }--[  ]--
# routing-policy policy policy01
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# insert statement red last
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# info with-context
    statement green {
        match {
            prefix-set pset1
        }
        action {
            policy-result reject
        }
    }
    statement blue {
        match {
            family [
                ipv4-unicast
                ipv6-unicast
            ]
        }
        action {
            policy-result next-policy
        }
    }
    statement red {
        match {
            protocol bgp
        }
        action {
            policy-result accept
        }
    }
```

Example: Move a statement to the beginning of a routing policy

The following example moves statement blue to the beginning of the routing policy:

```
--{ +* candidate shared default }--[  ]--
# routing-policy policy policy01
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# insert statement blue first
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# info with-context
    statement blue {
        match {
            family [
                ipv4-unicast
                ipv6-unicast
            ]
        }
        action {
            policy-result next-policy
        }
    }
    statement red {
        match {
            protocol bgp
        }
        action {
            policy-result accept
        }
    }
    statement green {
        match {
            prefix-set pset1
        }
        action {
            policy-result reject
        }
    }
```

Example: Move a statement before an existing statement

The following example moves statement blue before statement green:

```
--{ +* candidate shared default }--[  ]--
# routing-policy policy policy01
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# insert statement blue before green
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# info with-context
    statement red {
        match {
            protocol bgp
        }
        action {
            policy-result accept
        }
    }
    statement blue {
        match {
            family [
                ipv4-unicast
                ipv6-unicast
            ]
        }
        action {
            policy-result next-policy
        }
    }
    statement green {
        match {
            prefix-set pset1
        }
        action {
            policy-result reject
        }
    }
```

Example: Move a statement after an existing statement

The following example moves statement red after statement green:

```
--{ +* candidate shared default }--[  ]--
# routing-policy policy policy01
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# insert statement red after green
--{ +* candidate shared default }--[ routing-policy policy policy01 ]--
# info with-context
    statement green {
        match {
            prefix-set pset1
        }
        action {
            policy-result reject
        }
    }
    statement red {
        match {
            protocol bgp
        }
        action {
            policy-result accept
        }
    }
    statement blue {
        match {
            family [
                ipv4-unicast
                ipv6-unicast
            ]
        }
        action {
            policy-result next-policy
        }
    }
```

10.5 AS path sets

In a routing policy, an AS path set groups one or more AS paths using regular expressions. An AS path regular expression is a string consisting of AS numbers and (usually) one or more regular expression operators to be searched within the complete AS_PATH attribute. AS path sets can be referenced in match conditions and actions in routing policy statements.

An AS path set can have up to 32 member elements, each up to 255 characters in length.

Match-set-options for AS path sets

When used as a match condition in a policy statement, an AS path set containing multiple elements matches a route based on its configured `match-set-option`, which can be one of the following:

- **any**: a route is considered a match if its AS_PATH is matched by any of the member strings in the AS path set; if you do not specify a match-set-option, this is the default.
- **all**: a route is considered a match if its AS_PATH is matched by all of the member strings in the AS path set.
- **invert**:a route is considered a match if its AS_PATH is matched by none of the member strings in the AS path set.

10.5.1 Regex modes for AS path sets

A regular expression is a sequence of symbols and characters expressing a pattern to be searched within the complete AS_PATH attribute. The symbols are operators that imply a matching logic with respect to terms that precede or follow them. The smallest term that can be matched is the elementary term. Elementary terms can be combined to create more complex terms, and these can be collated to create even more complex terms.

In SR Linux, the regex mode configured for an AS path set determines the elementary term and the set of operator symbols that are supported. SR Linux supports the following regex modes:

- ASN mode

  The AS path is converted to a string and the string is matched one complete AS number at a time. AS-path sets are in ASN mode by default.
- Character mode

  The AS path is converted to a string and the string is matched one character at a time.

ASN mode

When an AS path set is configured in ASN mode, the string is matched by treating each AS number as an elementary term. When a route is compared to an AS path regular expression, each 4-byte AS number is extracted, one-by-one from the AS path, starting with the most recent AS and proceeding to the oldest or originating AS, and checked for a match against the regular expression. If there is any AS number that does not match the pattern, then the route is considered not a match for the AS path regular expression, and therefore not a match for the policy statement where this match condition is used.

SR Linux supports the following regex operations for strings in an AS path set configured in ASN mode:

Table 16: Supported regex operations for AS path sets in ASN mode

Operation	Meaning
null	Keyword equivalent to "^$"
|	Matches the term on alternate sides of the pipe.
*	Matches multiple occurrences of the term.
?	Matches 0 or 1 occurrence of the term.
+	Matches 1 or more occurrence of the term.
( )	Used to parenthesize so a regular expression is considered as one term.
[ ]	Used to demarcate a set of elementary or range terms.
[^]	Used to demarcate a set of elementary or range terms that are explicitly non-matching.
-	Used between the start and end of a range.
{m,n}	Matches at least m and at most n repetitions of the term.
{m}	Matches exactly m repetitions of the term.
{m,}	Matches m or more repetitions of the term.
^	Asserts that the following term appears at the beginning of the AS path.
$	Asserts that the preceding term appears at the end of the AS path.
_	Matches one or more non-word characters, beginning of string, or end of string.
< >	Matches any AS path numbers containing a confederation SET or SEQ.

The following table shows examples of valid regex strings for AS path sets in ASN mode.

Table 17: AS path set regex examples (ASN mode)

Regex string	Meaning	AS_PATH examples that match	AS_PATH examples that do not match
"null"	Match empty AS path.	empty	"100"
"< 100 >"	Match AS path that has only ASN 100 in a confed-sequence or confed-set.	"(100)"	"100""(100 200)"
"100 | 200"	Match AS path has only ASN 100 or ASN 200.	"100" "200"	empty "100 200" "(100)"

Regex string	Meaning	AS_PATH examples that match	AS_PATH examples that do not match
"(50 50){1,2}"	Match any AS path with 2 or 4 repetitions of 50.	"50 50" "50 50 50 50"	"50 50 50" "50 50 50 50 50"
"^[100-109]"	Match any AS path with single AS in the range 100-109 inclusive.	"105" "109"	"100 200" "110"
"[^100] 200$"	Match any AS path with two ASNs – the first must not be 100 and the second must be 200.	"105 200"	empty "100 200" "50 200 300"
"100 200"	Match any AS path with ASNs 100 and 200 occurring exactly in this sequence somewhere in the AS path.	"100 200" "3300 3501 100 200 2900"	empty "200 100" "100 101 200" "3300 100 3501 200 2900"

Character mode

When an AS path set is configured in character mode, the string is matched by treating each character in the string as an elementary term.

SR Linux supports the following regex operations for strings in an AS path set configured in character mode:

Table 18: Supported regex operations for AS path sets in character mode

Operation	Meaning
|	Matches the term on alternate sides of the pipe.
*	Matches 0, 1, or more occurrences of the term.
?	Matches 0 or 1 occurrence of the term.
+	Matches 1 or more occurrence of the term.
( )	Used to parenthesize so a regular expression is considered as one term.
[ ]	Used to demarcate a set of elementary terms, range terms, or character class.
[^]	Used to demarcate a set of elementary or range terms that are explicitly not matching.
-	Used between the start and end of a range.
{m,n}	Matches at least m and at most n repetitions of the term.
{m}	Matches exactly m repetitions of the term.

Operation	Meaning
{m,}	Matches m or more repetitions of the term.
^	Matches the beginning of the string.
$	Matches the end of the string.
_	Match one or more non-word characters, beginning of string, or end of string.
\[	Match left bracket (start of set).
\]	Match right brace (end of set).
\(	Match left parentheses (start of confed).
\)	Match right parentheses (end of confed).

The following table shows examples of valid regex strings for AS path sets in character mode.

Table 19: AS path set regex examples (character mode)

Regex string	Meaning	AS_PATH examples that match	AS_PATH examples that do not match
"^$"	Match empty AS path.	empty	"100"
"\(.*_100_.*\)"	Match any AS path with an AS confed sequence that includes 100.	"50 (100 200)"	"100"
"(_100_)|(_200_)"	Match any AS path that includes 100 or 200 at any location.	"100" "(100)" "100 200" "50 200 600"	empty "300"
"^100_"	Match any AS path that starts with 100 (most recent).	"100 500 600"	"50 100"
"_100$"	Match any AS path that ends with 100 (origin).	"100" "50 80 100"	empty "300"
"(_50_50_){1,2}"	Match any AS path with 2 or 4 repetitions of 50 at any location.	"30 50 50 100" "50 50 50 50"	"50" "100"
"^10[0-9]_"	Match any AS path with a leading AS in the range 100-109 inclusive.	"105 200 400" "109"	"200 100"

10.5.2 Configuring an AS path set

Procedure

To use an AS path set in a routing policy, you configure one or more regular expressions for the AS path set, optionally set the regex mode and, or match-set-option, then specify the AS path set as a match condition in the routing policy.

Example: Configure an AS path set in ASN mode

The following example configures an AS path set that consists of three regular expressions. The AS path set is configured in ASN mode, so when the AS_PATH attribute is converted to a string, the string is matched one complete AS number at a time. The match-set-option **any** is configured in the policy statement, so a route is considered a match if its AS_PATH is matched by any of the member strings in the AS path set.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        as-path-set asps1 {
            regex-mode asn
            as-path-set-member [
                "(50 50){1,2}"
                "100 | 200"
                "[^100] 200$"
            ]
        }
        policy p1 {
            statement 100 {
                match {
                    bgp {
                        as-path {
                            as-path-set asps1
                            match-set-options any
                        }
                    }
                }
            }
        }
    }
```

Example: Configure an AS path set in character mode

The following example configures an AS path set in character mode. When the AS_PATH attribute is converted to a string, the string is matched one character at a time. The match-set-option **all** is configured in the policy statement, so a route is considered a match only if its AS_PATH is matched by all of the member strings in the AS path set.

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        as-path-set asps2 {
            regex-mode character
            as-path-set-member [
                "(_100_)|(_200_)"
                "^10[0-9]_"
            ]
        }
        policy p2 {
            statement 100 {
                match {
                    bgp {
                        as-path {
                            as-path-set asps2
                            match-set-options all
                        }
                    }
                }
            }
        }
    }
```

10.6 BGP community-sets

A BGP community-set is a policy construct that groups one or more BGP communities into a list of member elements, each representing either a specific community value or a regular expression that is parsed one character at a time. Community-sets can be referenced in match conditions and actions in routing policy statements.

SR Linux supports the following types of community sets:

- Hybrid

  A hybrid community-set can contain any mix of the following:

  – standard 4-byte BGP community values or regular expressions (as defined in RFC 1997)

  – extended 8-byte BGP community values or regular expressions (as defined in RFC 4360)

  – large 12-byte BGP community values or regular expressions (as defined in RFC 8092)

  Hybrid community-sets are configured using `routing-policy.community-set`.

- Standard (maps to OpenConfig)

  A standard community-set can only contain standard 4-byte BGP community values or regular expressions (as defined in RFC 1997). Standard community-sets are configured using `routing-policy.standard-community-set`.

- Extended (maps to OpenConfig)

  An extended community-set can only contain extended 8-byte BGP community values or regular expressions (as defined in RFC 4360). Extended community-sets are configured using `routing-policy.extended-community-set`.

Match-set-options for community-sets

When used as a match condition in a policy statement, a community set containing multiple elements matches a route based on its configured match-set-option, which can be one of the following:

- **all**: a route matches the community-set if every element in the community-set is matched by a community attached to the route. If you do not specify a match-set-option, this is the default.
- **any**: a route matches the community-set if there is at least one element in the community-set that is matched by a community attached to the route.
- **invert**: a route matches the community-set if none of the elements in the community-set are matched by a community attached to the route.

For hybrid community sets, the match-set-option is configured as part of the community set. All policy statements that reference a hybrid community-set use the match-set-option configured for that community-set.

For standard and extended community-sets, the match-set-option is configured within the policy statement itself.

Using hybrid and standard/extended community-sets in one policy statement

The match container of a single policy statement can reference a hybrid community-set, a standard-community-set and, or an extended community-set. The policy statement does not apply to a route unless all the community-set matches return a TRUE result.

The `action.bgp` container of a single policy statement can reference a hybrid community-set, a standard-community-set and, or an extended-community-set. In this case, the hybrid community actions are applied first, before any of the processing (match or action) of the standard community-set or extended community-set is applied. This means that a community added by a hybrid community action can be matched by standard community-set or extended community-set.

10.6.1 Hybrid community-sets

In a routing policy, a policy statement can refer to a hybrid community set by making it the target of the leaf `routing-policy.policy.statement.match.bgp.community-set`. If the hybrid community-set contains multiple elements, then the matching behavior is controlled by `routing-policy.community-set.match-set-options`. (See Match-set-options for community-sets.)

The following table lists the valid formats for elements in a hybrid community set.

Table 20: Supported formats for elements in hybrid community-sets

Format	Example	Meaning
no-export		Well-defined standard community NO-EXPORT.
no-advertise		Well-defined standard community NO-ADVERTISE.
no-export-subconfed		Well-defined standard community NO-EXPORT-SUBCONFED.
<0-65535>:<0-65535>		Standard community matched exactly.
<regex> string must include one `:` and it is part of the match	`"65535:[0-9]"`	Standard communities matched by one regular expression See Table 21: Supported regex symbols for hybrid community-sets.
<0-4294967295>:<0-4294967295>:<0-4294967295>		Large community matched exactly.
<regex>:<regex>:<regex>	`"(100)|(101):65000:4294967295"`	Large communities matched by three LEGACY style regular

Format	Example	Meaning
string must include two `:` and they are not part of the match		expressions. See Table 21: Supported regex symbols for hybrid community-sets.
target:<0-65535>:<0-4294967295>	`"target:1:100"`	type0 route-target extended community matched exactly.
target:<ipv4-address>:<0-65535>	`"target:1.1.1.1:100"`	type1 route-target extended community matched exactly.
target:<0-4294967295>:<0-65535>		type2 route-target extended community matched exactly.
target:<regex>:<regex> string must start with `target` and include two `:` and they are not part of the match	`"target:(100)|(101)"`	route-target extended communities matched by two regular expressions. See Table 21: Supported regex symbols for hybrid community-sets.
origin:<0-65535>:<0-4294967295>		type0 route-origin extended community matched exactly.
origin:<ipv4-address>:<0-65535>		type1 route-origin extended community matched exactly.
origin:<0-4294967295>:<0-65535>		type2 route-origin extended community matched exactly.
origin:<regex>:<regex> string must start with `origin` and include two `:` and they are not part of the match		route-origin extended community matched by two regular expressions. See Table 21: Supported regex symbols for hybrid community-sets.
color:NN:<0-4294967295> NN=00,01,10,11	`"color:01:100"`	color extended community matched exactly.
bgp-tunnel-encap:(MPLS|VXLAN)	`"bgp-tunnel-encap:MPLS"`	BGP tunnel encapsulation extended community.

The following table lists the supported regex symbols for hybrid community-sets.

Table 21: Supported regex symbols for hybrid community-sets

Operation	Meaning
|	Matches the term on alternate sides of the pipe.
*	Matches multiple occurrences of the term.
?	Matches 0 or 1 occurrence of the term.
+	Matches 1 or more occurrence of the term.

Operation	Meaning
( )	Used to parenthesize so that a regular expression is considered as one term.
[ ]	Used to demarcate a set of elementary or range terms.
[^]	Used to demarcate a set of elementary or range terms that are explicitly non-matching.
-	Used between the start and end of a range.
{m,n}	Matches least m and at most n repetitions of the term.
{m}	Matches exactly m repetitions of the term.
{m,}	Matches m or more repetitions of the term.
^	Matches the beginning of the string.
$	Matches the end of the string.
\	An escape character to indicate that the following character is a match criteria and not a grouping delimiter.

10.6.1.1 Configuring a hybrid community-set

Procedure

To configure a hybrid community-set for a routing policy, you specify the member elements and match-set-option, then specify the community set as a match condition in the routing policy.

Example

The following example configures a hybrid community-set that consists of two member elements. The match-set-option is **invert**, so a route matches this community-set if neither of the elements in the community-set are matched by a community attached to the route. The community-set is specified as a match condition in a routing policy.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        community-set cs1 {
            match-set-options invert
            member [
                bgp-tunnel-encap:MPLS
                target:1.1.1.1:100
            ]
        }
        policy p1 {
            statement 100 {
                match {
                    bgp {
                        community-set cs1
                    }
                }
            }
        }
    }
```

10.6.2 Standard community-sets

In a routing policy, a policy statement can refer to a standard community-set by making it the target of the leaf `routing-policy.policy.statement.match.bgp.standard-community.standard-community-set`. If the standard community-set contains multiple elements, the matching behavior is controlled by `routing-policy.policy.statement.match.bgp.standard-community.match-set-options`. (See Match-set-options for community-sets.)

Each standard community-set consists of a leaf-list of member elements. Each member represents either a specific community value or a regex string to be evaluated. SR Linux decides whether to treat a community-set member as an exact match or regex string as follows.

Table 22: Determining exact match or regex string for standard community-set member elements

Member format[3]	Interpreted as
<d>:<d>	Exact match of a standard community (not passed to regex engine).
<d> <r>	String to be passed to the regex engine.

SR Linux supports the following operations for strings passed to the regex engine:

Table 23: Supported regex operations for standard and extended community-sets

Operation	Meaning
|	Matches the term on alternate sides of the pipe.
*	Matches 0, 1, or more occurrences of the term.
?	Matches 0 or 1 occurrence of the term.
+	Matches 1 or more occurrence of the term.
`( )`	Used to parenthesize so a regular expression is considered as one term.
`[ ]`	Used to demarcate a set of elementary terms, range terms, or character class.
`[^ ]`	Used to demarcate a set of elementary or range terms that are explicitly non-matching.
-	Used between the start and end of a range.
`{m,n}`	Matches least `m` and at most `n` repetitions of the term.
`{m}`	Matches exactly `m` repetitions of the term.
`{m,}`	Matches `m` or more repetitions of the term.
^	Matches the beginning of the string.

[3] where <d> is a string with digits only and no regex operation symbols, and <r> is a string with at least one regex operation symbol

Operation	Meaning
$	Matches the end of the string.
\	An escape character to indicate that the following character is match criteria and not a grouping delimiter.

The following table shows examples of valid regex strings in standard community-sets.

Table 24: Standard community regex examples

Regex string	Meaning	Communities that match	Communities that do not match
".*"	Match all standard communities.	65200:100	
"^651[0-9][0-9]:.*$" "^651[0-9][0-9]"	Match any standard community with 65100-65199 as the AS.	65150:200	65200:200
"^65100"	Match any standard community associated with ASN 65100.	65100:100 65100:200	65101:900
"666$"	Match any standard community that ends in the digits 666.	65100:666 65100:6666	666:1
":666$""_666$"	Match any standard community for which the administered value is exactly 666.	65100:666	65100:6666

10.6.2.1 Configuring a standard community-set

Procedure

To configure a standard community-set for a routing policy, you specify the member elements, then specify the community-set as a match condition in the routing policy.

Example

The following example configures a community-set consisting of four member elements: one specific community and three regex strings. The community-set is specified as a match condition in a routing policy. The match-set-option **any** is configured in the policy statement, so a route is considered a match if its AS_PATH is matched by any of the member elements in the AS path set

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        standard-community-set s1 {
            member [
                65200:200
                10*:34684
                ^65100
                "^651[0-9][0-9]:.*$"
            ]
        }
        policy p2 {
            statement 100 {
                match {
                    bgp {
                        standard-community {
                            standard-community-set s1
                            match-set-options any
                        }
                    }
                }
            }
        }
    }
```

10.6.3 Extended community-sets

In a routing policy, a policy statement can refer to an extended community-set by making it the target of the leaf `routing-policy.policy.statement.match.bgp.extended-community.extended-community-set`. If the extended community-set contains multiple elements, the matching behavior is controlled by `routing-policy.policy.statement.match.bgp.extended-community.match-set-options`. (See Match-set-options for community-sets.)

Each extended community-set consists of a leaf-list of member elements. Each member represents either a specific community value, or a regex string to be evaluated. SR Linux decides whether to treat a community-set member as an exact match or regex string as follows:

Table 25: Determining exact match or regex string for extended community-set member elements

Member format[4]	EC type/subtype	Interpreted as

ext:<4h>:<1-12h>	type = *
	subtype = *
		Generic extended community exact match (not passed to regex engine)

ext
ext:
ext:<h>
ext:<r>
	type = *
	subtype = *
		Generic extended community regex

gbp-tag:<d>:<d>	type = 0x03
	subtype = 0x17
		Group Policy ID extended community exact match (not passed to regex engine)

gbp-tag
gbp-tag:
gbp-tag:<d>
	type = 0x03
	subtype = 0x17
		Group Policy ID extended community regex

[4] where <d> is a string with digits only and no regex operation symbols, <r> is a string with at least one regex operation symbol, and <a> is a string of alphabetic characters with no regex operation symbols

Member format[4]	EC type/subtype	Interpreted as
gbp-tag<r>		

target:<d>:<d>	type = 0x00, 0x01, 0x02
	subtype = 0x02
		Route target extended community exact match (not passed to regex engine)

target:<r>:<d>
target:<d>:<r>
target:<r>:<r>
target
target:
target:<d>
target<r>
	type = 0x00, 0x01, 0x02
	subtype = 0x02
		Route target extended community regex

origin:<d>:<d>	type = 0x00, 0x01, 0x02
	subtype = 0x03
		Route origin extended community exact match (not passed to regex engine)

origin:<r>:<d>
origin:<d>:<r>
origin:<r>:<r>
origin
origin:
origin:<d>
origin<r>
	type = 0x00, 0x01, 0x02
	subtype = 0x03
		Route origin extended community regex

link-bandwidth:<d>:<d>[kMGT]	type = 0x00, 0x40
	subtype = 0x04
		Link-bandwidth extended community exact match (not passed to regex engine)

link-bandwidth
link-bandwidth:
link-bandwidth:<d>
link-bandwidth<r>[kMGT]
	type = 0x00, 0x40
	subtype = 0x04
		Link-bandwidth extended community regex

color:<bits>:<d>	type = 0x03
	subtype = 0x0B
		Color extended community exact match (not passed to regex engine)

color
color:
	type = 0x00, 0x40
	subtype = 0x04
		Color extended community regex

[4] where <d> is a string with digits only and no regex operation symbols, <r> is a string with at least one regex operation symbol, and <a> is a string of alphabetic characters with no regex operation symbols

Member format[4]	EC type/subtype	Interpreted as
color:<bits> color<r>		

bgp-tunnel-encap:<a>	type = 0x03
	subtype = 0x0CS
		BGP tunnel encap extended community exact match (not passed to regex engine)

See Table 23: Supported regex operations for standard and extended community-sets for valid operations for strings passed to the regex engine.

The following table shows examples of valid regex strings for route target and route origin elements in extended community-sets.

Table 26: Extended community regex examples for route target and route origin

Regex string	Meaning	Communities that match	Communities that do not match
"t.*"	Not valid, rejected.		
"o.*"	Not valid, rejected.		
"target"	Match all RT ext communities.	target:65000:100	link-bandwidth:65000:100M
"origin"	Match all RO ext communities.	origin:65000:100	target:65000:100
"target:.*10.*"	Match any RT ext community (any type) with two consecutive digits 10.	target:65100:99 target:10.5.6.7:99 target:4294967295:1105	target:999:999 target:1.0.1.0:999
"target:10.1.1.1:.*"	Match type-1 RT ext community with 10.1.1.1 address as the administrator.	target:10.1.1.1:100	
"target:10\..*"	Match type-1 RT ext community with any 10/8 address as the administrator.	target:10.1.1.1:100	target:105:100
"target:10.*"	Match any RT ext community with leading digits 10 in the administrator part.	target:105:999 target:100000:999 target:102.3.4.1:999	target:1110:999
"target.*_666$"	Match any RT ext community where the	target:65000:666 target:10.5.6.7:666	target:65

[4] where <d> is a string with digits only and no regex operation symbols, <r> is a string with at least one regex operation symbol, and <a> is a string of alphabetic characters with no regex operation symbols

Regex string	Meaning	Communities that match	Communities that do not match
	administered value is exactly 666.	target:4294967295:666	

The following table shows examples of valid regex strings for link-bandwidth elements in extended community-sets.

Table 27: Extended community regex examples for link-bandwidth

Regex string	Meaning	Communities that match	Communities that do not match
"l.*"	Not valid, rejected.		
"lb.*"	Not valid, rejected.		
"link-bandwidth" "link-bandwidth:.*"	Match all LB ext communities.	link-bandwidth:65000:40k	route-target:65000:100
"link-bandwidth:10.*	Match any LB ext community with leading digits 10 in the administrator part.	link-bandwidth:10:1T link-bandwidth:105:1200	link-bandwidth:11:10M
"link-bandwidth.*_100G$"	Match any LB ext community that encodes the bandwidth value 100Gbps.	link-bandwidth:65000:100G link-bandwidth:3333:100G	link-bandwidth:100:100M link-bandwidth:100:1100G

The following table shows examples of valid regex strings for elements in generic extended community-sets.

Table 28: Extended community regex examples for generic extended community-sets

Regex string	Meaning	Communities that match	Communities that do not match
"e*"	Not valid, rejected.		
"ext" "ext:.*"	Match all ext communities.	ext:0600:000000000001	None
"ext:0002.*"	Match all 2-octet AS-specific route-target extended communities	route-target:65000:100	ext:0600:000000000001
"^ext:0600_.*1$"	Match all EVPN MAC Mobility extended	ext:0600:000000000001	ext:0601:000000000001

Regex string	Meaning	Communities that match	Communities that do not match
	communities with lowest bit = 1.		

The following table shows examples of valid regex strings for elements in Group Policy ID extended community-sets.

Table 29: Extended community regex examples for generic Group Policy ID community-sets

Regex string	Meaning	Communities that match	Communities that do not match
`"gbp*"`	Not valid, rejected.		
`"gbp-tag"` `"gbp-tag:.*"`	Match all Group Policy ID extended communities.	gbp-tag:0:53	route-target:65000:100
`"gbp-tag:0:.*`	Match all Group Policy ID extended communities with scope = 0	gbp-tag:0:53	gbp-tag:1:53

10.6.3.1 Configuring an extended community-set

Procedure

To configure an extended community-set for a routing policy, you specify the member elements, then specify the community-set as a match condition in the routing policy.

Example: Configure an extended community-set with route target and route origin member elements

The following example configures a community-set consisting of four member elements: one specific community and three regex strings. The community-set is specified as a match condition in a routing policy. The match-set-option **any** is configured in the policy statement.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        extended-community-set e1 {
            member [
                origin.*_666$
                origin:.*10.*
                target:1.1.1.1:100
                target:1:100
            ]
        }
        policy p3 {
            statement 100 {
                match {
                    bgp {
                        extended-community {
                            extended-community-set e1
                            match-set-options any
                        }
                    }
                }
            }
        }
    }
```

Example: Configure an extended community-set with link-bandwidth member elements

The following example configures a community-set consisting of elements that match link-bandwidth extended communities. The community-set contains one specific link-bandwidth community and two regex strings.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        extended-community-set e2 {
            member [
                link-bandwidth.*_100G$
                link-bandwidth:10.*
                link-bandwidth:65000:1T
            ]
        }
        policy p3 {
            statement 100 {
                match {
                    bgp {
                        extended-community {
                            extended-community-set e2
                            match-set-options any
                        }
                    }
                }
            }
        }
    }
```

Example: Configure a generic extended community-set

The following example configures a generic community-set. The community-set contains two regex strings.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        extended-community-set e3 {
            member [
                ext:.*
                ext:0002.*
            ]
        }
        policy p3 {
            statement 100 {
                match {
                    bgp {
                        extended-community {
                            extended-community-set e3
                            match-set-options any
                        }
                    }
                }
            }
        }
    }
```

10.7 Prefix sets

In a routing policy, a prefix set groups one or more IPv4 and, or IPv6 prefix matching entries. A prefix matching entry has two components:

- an IPv4 or IPv6 prefix, specifying an IPv4 or IPv6 address with zeroed host bits and a prefix length; for example, 192.168.1.0/21
- a `mask-length-range`, specifying either `exact` or a range in the format <a>..<b>, where <a> is the starting prefix length (0-32 for IPv4, 0-128 for IPv6), and <b> is the ending prefix length (0-32 for IPv4, 0-128 for IPv6)

A prefix set containing multiple elements matches a route based on its configured `match-set-option`. For prefix sets, the default and only supported `match-set-option` is **any**, which means that the route is considered a match if its destination is matched by any of the prefix matching entries.

10.7.1 Configuring a prefix set

Procedure

To configure a prefix set for a routing policy, you specify the member elements, then specify the prefix set as a match condition in the routing policy.

Example

The following example configures a prefix set that consists of two member elements. The prefix set is specified as a match condition in a routing policy.

```
--{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        prefix-set pset1 {
            prefix 10.10.1.0/24 mask-length-range 24..32 {
            }
            prefix 10.10.20.0/24 mask-length-range exact {
            }
        }
        policy p1 {
            statement 100 {
                match {
                    prefix {
                        prefix-set pset1
                        match-set-options any
                    }
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

11 Network instance route limits

In certain situations, you may want to limit the maximum number of IPv4 or IPv6 routes per network instance installed in the FIB. By default, routes that exceed the configured maximum are not installed. However, you can configure the system to allow the excess routes to be programmed and to record the route limit condition as a warning log event. In this case, the exceeding route remains active in state.

The route limit is determined based on the number of unique prefixes installed.

11.1 Protocol behavior when route limit is reached

When a protocol publishes an IPv4 or IPv6 route to the fib_mgr that exceeds the configured maximum route limit, fib_mgr suppresses the installation of that route. The system records the rejected routes and the exceeded limit in state at the following paths:

network-instance route-table [ipv4-unicast | ipv6-unicast] route id route-type route-owner origin-network-instance active = **false**

network-instance route-table [ipv4-unicast | ipv6-unicast] route id route-type route-owner origin-network-instance route-limit-exceeded = **true**

network-instance route-limit [ipv4 | ipv6] maximum-exceeded = **true**

Regardless of the source of the route, the system does not remove the failed FIB route from the route table. The route continues to appear as an entry in state under **network-instance route-table [ipv4-unicast | ipv6-unicast] route**. The fib_mgr does not retry the installation of suppressed routes. Each protocol must resubmit routes instead.

The behavior of the route source protocol is protocol-dependent and similar to a FIB programming failure, as described in the following sections.

BGP

When the route limit is reached, the behavior of BGP routes (including **bgp**, **bgp-evpn**, **bgp-evpn-iff**, **bgp-evpn-ifl-host**, **bgp-ipvpn**, **bgp-label**) is as follows:

- In the BGP RIB-IN, the route is identified as used (because it is published to fib_mgr) but invalid. Under **network-instance bgp-rib afi-safi** <*afi-safi*> **rib-in-out rib-in-post**, the following state is displayed:
  - **route used-route** = **true**
  - **route valid-route** = **false**
  - **route invalid-reason fib-programming-failed** = **true**
- If the route is already advertised in the RIB-OUT, the route is withdrawn. If BGP is waiting for FIB confirmation, it cancels the advertisement of the route in the RIB-OUT.
- If the route is received again in the BGP RIB-IN (with or without modified path attributes), the system attempts to reprogram the route.

IGPs (including IS-IS, OSPFv2, and OSPFv3)

When an IS-IS or OSPF instance determines that one of its routes is rejected due to reaching the route-limit, the IGP instance immediately goes into overload state.

To clear the overload state, use the **tools network-instance protocols [isis | ospf] instance overload clear** command. This command triggers the IGP to perform an SPF recalculation and to resubmit all failed routes to the fib_mgr.

Static and aggregate routes

Static routes and aggregate routes that are rejected when the route limit is reached appear with the following state:

network-instance static-routes route installed = **false**

network-instance aggregate-routes route installed = **false**

To retry a static or aggregate route that has failed programming, toggle the **route admin-state** from **disable** to **enable**.

ARP-ND

For an IPv4 ARP or IPv6 ND host route that has failed FIB programming, the programming is retried whenever the IPv4 ARP or IPV6 neighbor entry is refreshed.

gRIBI

For gRIBI routes (except for gRIBI decapsulation pseudo-routes), the system reports the FIB programming status as FIB_FAILED.

Host, local, Linux

For host, local, and Linux routes (and gRIBI decapsulation pseudor-routes), programming never fails as a result of the route limit configuration.

11.2 Configuring network instance route limits

Procedure

To configure network instance route limits for IPv4 and IPv6 families, use the **route-limit** command. The default value for each family is 0, which indicates no limit.

Example: Configure network instance route limit for IPv4

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default route-limit *
    network-instance default {
        route-limit ipv4 {
            maximum 100000
            warning-only true
            warning-threshold-percent 90
        }
        route-limit ipv6 {
            maximum 300000
            warning-only false
            warning-threshold-percent 85
        }
    }
```

12 Table connections for route redistribution

Table-connections are a data model concept, standardized by OpenConfig, for managing redistribution of routes between protocols. In a network-instance, a table-connection exists for every inter-protocol route redistribution vector. For example, both of the following redistribution vectors are represented by a table-connection:

- redistribution of IPv4 static routes into the IPv4 BGP RIB
- redistribution of IPv6 BGP routes into the IPv6 IS-IS route table

Table-connections are structured as follows:

```
+--network-instances
     +--network-instance* [name]
        +--table-connections
           +--table-connection [source-protocol destination-protocol address-family]
              +--disable-metric-propagation?
              +--import-policy
              +--default-import-policy
```

In this model, each `table-connection` can have an `import-policy`. The `import-policy` controls the specific set of routes belonging to `address-family` to be redistributed from the `source-protocol` to the `destination-protocol`. The `import-policy` is evaluated against all routes of the `source-protocol` (across all instances of this protocol) with four possible outcomes:

- Matched routes with an `accept` action are redistributed to the `destination-protocol`.
- Matched routes with a `reject` action are not redistributed to the `destination-protocol`.
- Unmatched routes are redistributed to the `destination-protocol` if the `default-import-policy` is `accept`.
- Unmatched routes are not redistributed to the `destination-protocol` if the `default-import-policy` is `reject`.

The default value of `default-import-policy` is `reject`, which means a table-connection that does not have a configured `import-policy` does not redistribute any routes.

Note:

- Destination protocol B cannot advertise a route of source protocol A unless there is an A → B table connection that causes the route to be accepted.
- When a route of source protocol A is redistributed into destination protocol B, it is added to the RIB of protocol B, and therefore is advertisable to peers of protocol B without any export policy.
- Table connection policies can change the properties of redistributed routes.

12.1 Supported table-connections in SR Linux

SR Linux supports the following table-connections for redistributing routes between protocols (both IPv4 and IPv6):

- BGP → IS-IS
- Static → BGP
- Static → IS-IS
- Local → IS-IS
- Local → BGP

Each of these table-connections supports its own configurable import policy (or chain of import policies). The import policy (or policy chain) evaluates the best routes of each source protocol. (In this context, *best route* means the most-preferred route for the prefix considering all protocols; this is the route that is installed as the active route.)

The following table summarizes the routing policy match conditions and actions supported by each table connection:

Table 30: Supported routing policy match conditions and actions for table-connections

Table-connection	Supported match conditions	Supported actions	Effect of `disable-metric-propagation` setting
Static → BGP	• `call-policy` • `prefix-set` • `tag-set` • `family` (mismatch with `address-family` of the table-connection causes no routes to be matched) • `protocol` (only `static` causes routes to be matched)	• `bgp.as-path` • `bgp.med` • `bgp.local-preference` • `bgp.standard-community` • `bgp.next-hop` • `bgp.communities` • `bgp.extended-community`	• If `disable-metric-propagation = true`, the route is originated without a MED attribute • If `disable-metric-propagation = false`, the route is originated with a MED attribute and the value comes from the metric of the static route
BGP → IS-IS	• `call-policy` • `prefix-set` • `bgp.standard-community` • `tag-set`	• `isis.level` • `isis.metric.set-style` • `isis.metric.set-value`	• if `disable-metric-propagation = true`, the route is originated with an external metric of 0

Table-connection	Supported match conditions	Supported actions	Effect of `disable-metric-propagation` setting
	- `family` (mismatch with `address-family` of the table-connection causes no routes to be matched) - `protocol` (only `bgp` causes routes to be matched) - `bgp.as-path` - `bgp.community-set` - `bgp.extended-community`		- if `disable-metric-propagation = false`, the route is originated with an external metric copied from the MED value (but capped at 2^24-1).
Static → IS-IS	- `call-policy` - `prefix-set` - `tag-set` - `family` (mismatch with `address-family` of the table-connection causes no routes to be matched) - `protocol` (only `static` causes routes to be matched)	- `isis.level` - `isis.metric.set-style` - `isis.metric.set-value`	- if `disable-metric-propagation = true`, the route is originated with an external metric of 0. - if `disable-metric-propagation = false`, the route is originated with an external metric copied from the metric of the static route (but capped at 2^24-1).
Local → IS-IS	- `call-policy` - `prefix-set` - `family` (mismatch with `address-family` of the table-connection causes no routes to be matched) - `protocol` (only `local` causes routes to be matched)	- `isis.level` - `isis.metric.set-style` - `isis.metric.set-value`	- If `disable-metric-propagation = true`, or `disable-metric-propagation = false`, the route is originated with an external metric of 0.

Table-connection	Supported match conditions	Supported actions	Effect of disable-metric-propagation setting
Local → BGP	• call-policy • prefix-set • family (mismatch with address-family of the table-connection causes no routes to be matched) • protocol (only local causes routes to be matched)	• bgp.as-path • bgp.med • bgp.local-preference • bgp.standard-community • bgp.next-hop • bgp.communities • bgp.extended-community	• If disable-metric-propagation = true, the route is originated without a MED attribute • if disable-metric-propagation = false, the route is originated with a MED attribute with a value of zero.

12.2 Enabling table-connections

Table-connections are disabled by default in SR Linux. To enable table-connections for a network-instance you set its `table-connections.admin-state` to `enable`.

When `table-connections.admin-state` is set to `disable`, configuration of table-connection list entries is not possible for the network-instance. In this case, for protocol B to advertise active routes of protocol A, protocol B must be configured with an export-policy that accepts routes of protocol A.

When `table-connections.admin-state` is set to `enable`:

- Protocol B cannot advertise an active route of protocol A unless an A → B table connection is configured that causes the route to be accepted.
- When a route of protocol A is redistributed to protocol B, it is added to the RIB of protocol B and therefore is advertisable to peers of protocol B without an export policy. However, if protocol B does have an export policy, it has final control over the advertisement of the redistributed route. In this context, table connection policies can change the properties of redistributed routes, and the export policies of the destination protocol can change the same route properties again.

  For example, a static → BGP table connection could add a MED value of 10, and a BGP export policy applied to a particular neighbor or peer-group could further increment this MED by 10.

Note: Even though the redistributed route is added to the RIB of the destination protocol B, the protocol of the redistributed route from the point of view of export policies of protocol B remains the original protocol (A). An implication of this is that a route redistributed from protocol A to protocol B cannot be further redistributed to protocol C.

12.3 Configuring table-connections

Procedure

To configure a table-connection to redistribute routes from one protocol to another, you enable table connections for the network-instance, specify the source protocol, address family, destination protocol, and import-policy. You can optionally specify the default-import-policy (either accept or reject) for routes that do not match the import-policy, as well as whether the route metric is carried over from the source to the destination protocol.

Example

The following example configures a table-connection to redistribute routes from BGP to IS-IS. Table-connections are enabled for the default network-instance, BGP is specified as the source protocol, and IS-IS is specified as the destination protocol. The import-policy is a chain of two routing policies. The default-import-policy is accept, so routes not matching the import-policy chain are accepted.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default
    network-instance default {
        table-connections {
            admin-state enable
            table-connection bgp address-family ipv4 destination-protocol isis {
                default-import-policy accept
                import-policy [
                    p1
                    p2
                ]
            }
        }
    }
```

Customer document and product support

Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback