



Nokia Service Router Linux

Release 24.3

Routing Protocols Guide

3HE 20248 AAAA TQZZA
Edition: 01
March 2024

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.

Copyright and trademark: Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

© 2024 Nokia.

Table of contents

1	About this guide.....	6
1.1	Precautionary and information messages.....	6
1.2	Conventions.....	6
2	What's new.....	8
3	OSPF.....	9
3.1	OSPF global configuration.....	9
3.1.1	Configuring basic OSPF parameters.....	10
3.1.2	Configuring the router ID.....	11
3.1.3	Configuring an area.....	12
3.1.4	Configuring a stub area.....	12
3.1.5	Configuring a Not-So-Stubby area.....	13
3.1.6	Configuring an interface.....	13
4	BGP.....	15
4.1	BGP global configuration.....	15
4.1.1	Configuring an ASN.....	15
4.1.2	Configuring the router ID.....	16
4.2	Configuring a BGP peer group.....	16
4.3	Configuring BGP neighbors.....	17
4.4	BGP peer import and export policies.....	17
4.5	eBGP multihop.....	19
4.5.1	Configuring eBGP multihop.....	20
4.6	AS path options.....	20
4.6.1	Configuring allow-own-as.....	21
4.6.2	Configuring replace-peer-as.....	21
4.6.3	Configuring remove-private-as.....	22
4.7	BGP MED.....	23
4.7.1	Configuring always-compare-med.....	23
4.8	Route reflection.....	24
4.8.1	Configuring route reflection.....	24
4.9	BGP graceful restart.....	25
4.9.1	Configuring graceful restart.....	25

4.10	BGP unnumbered peering.....	26
4.10.1	Configuring BGP unnumbered peering.....	29
4.11	Prefix-limit for BGP peers.....	29
4.11.1	Configuring the prefix-limit for BGP peers.....	30
4.12	BGP configuration management.....	32
4.12.1	Modifying an ASN.....	32
4.12.2	Deleting a BGP neighbor.....	32
4.12.3	Deleting a BGP peer group.....	32
4.12.4	Resetting BGP peer connections.....	33
4.13	BGP shortcuts.....	33
4.13.1	Configuring BGP shortcuts over segment routing.....	35
4.14	BGP TCP MSS.....	36
4.14.1	Configuring BGP TCP MSS.....	37
4.15	Error handling for BGP update messages.....	38
5	Seamless MPLS with BGP labeled unicast (BGP-LU).....	39
5.1	Configuration.....	39
5.2	Initial configuration for Seamless MPLS.....	40
5.2.1	Configuring interfaces.....	40
5.2.2	Configuring IS-IS.....	41
5.2.3	Configuring MPLS label blocks.....	43
5.2.4	Configuring LDP.....	43
5.3	BGP configuration for Seamless MPLS.....	44
5.3.1	Configuring BGP on ABRs.....	45
5.3.2	Configuring BGP on the core RR.....	46
5.3.3	Configuring BGP on ANs toward ABRs.....	47
5.4	Export policy configuration for Seamless MPLS.....	48
5.4.1	Configuring export policies on ANs and ABRs.....	48
6	IS-IS.....	50
6.1	Basic IS-IS configuration.....	52
6.1.1	Enabling an IS-IS instance.....	52
6.1.2	Configuring the router level.....	52
6.1.3	Configuring the Network Entity Title.....	53
6.1.4	Configuring global parameters.....	54
6.1.5	Configuring interface parameters.....	54

6.1.6	Configuring authentication keys.....	55
6.2	IS-IS graceful restart.....	56
6.2.1	Configuring IS-IS graceful restart.....	57
6.3	Displaying IS-IS information.....	57
6.4	Clearing IS-IS information.....	61
6.5	IS-IS weighted ECMP.....	62
6.5.1	Enabling weighted ECMP.....	62
6.5.2	Configuring weighted load-balancing over interface next-hops.....	64
6.5.3	Normalizing datapath weights.....	65
6.6	Multi-instance IS-IS.....	65
6.6.1	Configuring the IID.....	65
6.6.2	Enabling the IID-TLV.....	66
6.6.3	Displaying MI-ISIS information.....	66
7	Protocol authentication.....	68
7.1	Configuring protocol authentication.....	68
8	Routing policies.....	71
8.1	Creating a routing policy.....	71
8.1.1	Specifying match conditions.....	71
8.1.1.1	Specifying a list as a match condition.....	72
8.1.2	Specifying actions.....	73
8.1.2.1	Policy actions for setting MED in BGP routes.....	74
8.1.3	Specifying a default action.....	75
8.2	Applying a routing policy.....	76
8.2.1	Applying a default policy to eBGP sessions.....	77
8.2.2	Replacing an AS path.....	77
8.2.3	BGP community-sets.....	78
8.2.3.1	Configuring a BGP community-set.....	79

1 About this guide

This document describes routing protocols used with the 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 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

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.
- An open right-angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start > connect to**.
- 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.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 What's new

Topic	Location
Configurable match-set-option for BGP community-sets	BGP community-sets
Seamless MPLS with BGP-LU	Seamless MPLS with BGP labeled unicast (BGP-LU)

3 OSPF

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 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 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 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 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 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 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 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 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 on 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 help, as well as the *SR Linux Advanced Solutions Guide* and the *SR Linux Data Model Reference*.

4.1 BGP global configuration

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

4.1.1 Configuring an ASN

Procedure

You can configure an Autonomous System Number (ASN) for a router. 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 network-instance default
  network-instance default {
    protocols {
      bgp {
```

```

    autonomous-system 65002
  }
}

```

4.1.2 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 network-instance default
  network-instance default {
    protocols {
      bgp {
        router-id 2.2.2.2
      }
    }
  }
}

```

4.2 Configuring a BGP peer group

Procedure

You can 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 network-instance default
  network-instance default {
    protocols {
      bgp {
        group headquarters1 {
          admin-state enable
          traceoptions {
            flag events {
            }
            flag graceful-restart {
            }
          }
        }
      }
    }
  }
}

```



```
}

```

4.3 Configuring BGP neighbors

Procedure

After configuring a BGP peer group and assigning options, you can 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 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
        }
      }
    }
  }
}
```

4.4 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 certain 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.

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

4.5 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.

4.5.1 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 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 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 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
        }
      }
    }
  }
}
```

4.6 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.

4.6.1 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 network-instance default
  network-instance default {
    protocols {
      bgp {
        autonomous-system 65001
        as-path-options {
          allow-own-as 1
        }
      }
    }
  }
}
```

4.6.2 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 network-instance default
```

```

network-instance default {
  protocols {
    bgp {
      as-path-options {
        replace-peer-as true
      }
    }
  }
}

```

4.6.3 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 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 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 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 network-instance default
  network-instance default {
    protocols {
      bgp {
        as-path-options {
          remove-private-as {
            mode replace
            ignore-peer-as true
          }
        }
      }
    }
  }
}
```

4.7 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.

4.7.1 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 network-instance default protocols bgp best-path-selection
  network-instance default {
    protocols {
      bgp {
        best-path-selection {
          always-compare-med true
        }
      }
    }
  }
}
```

4.8 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.

4.8.1 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 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
      }
    }
  }
}
```

4.9 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 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 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.

4.9.1 Configuring graceful restart

Procedure

You can configure graceful restart for the BGP instance. The SR Linux 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 itself is restarting.

Example

When operating as a helper router, the SR Linux 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 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 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 system warm-reboot
  system {
    warm-reboot {
      bgp-max-wait 270
    }
  }
}
```

4.10 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:fe9c: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:fe9c: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:fe9c: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:fe9c: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).

4.10.1 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 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.

4.11 Prefix-limit for BGP peers

SR Linux places a limit 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, then re-establishes the session.

You can configure the following settings for the prefix-limit:

- `max-received-routes`
By default, the prefix-limit that triggers a BGP session teardown is 4294967295 routes, which is the maximum number of routes that can be received from the peer (counting routes accepted and rejected by import policy). You can configure a different prefix-limit by setting a value for `max-received-routes`.
- `prefix-limit-restart-timer`
By default, after a BGP session is torn down because the prefix limit was exceeded, the BGP session is re-established immediately. You can configure the number of seconds the system waits before re-establishing the session by setting a value for `prefix-limit-restart-timer`.
- `prevent-teardown`
You can prevent the BGP session from being torn down when the prefix-limit is exceeded by setting `prevent-teardown` to `true`.
- `warning-threshold-pct`
You can set a warning threshold for the prefix-limit. When the number of routes received from the peer (counting routes accepted and rejected by import policy) reaches a specified percentage of the `max-received-routes` setting, BGP raises a warning log event. The default threshold is 90%.

When upgrading from a release earlier than 23.3.1 to Release 23.3.1 or later, the upgrade script checks the configured `max-received-routes` setting for IPv4 and IPv6 routes. If the configured `max-received-routes` setting is equal to 4295967295 for IPv4 or IPv6 routes, then `prevent-teardown` for IPv4 or IPv6 routes is set to `true`.

4.11.1 Configuring the prefix-limit for BGP peers

Procedure

To configure the prefix-limit, you can set the maximum number of routes from a peer, number of seconds the system waits to re-establish a session following a teardown, and disable the prefix-limit for a peer.

The commands to set maximum number of routes from a peer and disable the prefix-limit can be applied to IPv4 and IPv6 routes. The 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 the peer and peer group, the system default applies.

Example: Configure maximum number of routes from a peer

The following example configures the maximum number of IPv4 routes that can be received from a peer.

```
--{ candidate shared default }--[ ]--
# info network-instance default protocols bgp neighbor 192.168.11.1 afi-safi ipv4-unicast
  ipv4-unicast
    network-instance default {
      protocols {
        bgp {
```

```

neighbor 192.168.11.1 {
  afi-safi ipv4-unicast {
    ipv4-unicast {
      prefix-limit {
        max-received-routes 30000000
      }
    }
  }
}

```

If `max-received-routes` is not configured for the peer, the `max-received-routes` setting for the peer group applies. If `max-received-routes` is not configured for the peer group, the system default maximum of 4294967295 routes applies.

Example: Configure prefix-limit restart timer

The following example sets the number of seconds the system waits to re-establish a BGP session with a peer after the session was torn down because the `max-received-routes` value was exceeded.

```

--{ candidate shared default }--[ ]--
# info network-instance default protocols bgp neighbor 192.168.11.1 timers
network-instance default {
  protocols {
    bgp {
      neighbor 192.168.11.1 {
        timers {
          prefix-limit-restart-timer 60
        }
      }
    }
  }
}

```

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, the BGP session with the peer is re-established immediately after teardown (that is, `prefix-limit-restart-timer = 0` seconds).

Example: Disable the prefix-limit

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

```

--{ candidate shared default }--[ ]--
# info network-instance default protocols bgp neighbor 192.168.11.1 afi-safi ipv4-unicast
  ipv4-unicast
network-instance default {
  protocols {
    bgp {
      neighbor 192.168.11.1 {
        afi-safi ipv4-unicast {
          ipv4-unicast {
            prefix-limit {
              prevent-teardown true
            }
          }
        }
      }
    }
  }
}

```

```
}  
  }  
} }
```

4.12 BGP configuration management

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

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

4.12.1 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.

4.12.2 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
```

4.12.3 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
```

4.12.4 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.
```

4.13 BGP shortcuts

With BGP shortcuts, SR Linux can include LDP LSPs or segment routing (SR-ISIS) 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.



Note: BGP shortcuts can only be used for next-hop resolution of IPv4-unicast RIB-Ins with an IPv4 next-hop address.

BGP next-hop resolution describes the procedures that BGP uses to resolve the next-hop address of each BGP RIB-In that forms part of a BGP route. The following table defines BGP RIB-In and BGP route in the context of BGP next-hop resolution.

Table 1: BGP RIB-IN and BGP route

BGP Term	Definition
BGP RIB-In	One of the following: <ul style="list-style-type: none"> a received IPv4-unicast BGP route with an IPv4 next-hop address a received IPv4-unicast BGP route with an IPv6 next-hop address (allowed as a result of sending an extended-nh-encoding capability to the peer) a received IPv6-unicast BGP route with an IPv6 next-hop address
BGP route	A route submitted by BGP to the fib_mgr that resulted from the grouping of one or more BGP RIB-Ins. (Multiple BGP RIB-Ins per route describes a multipath scenario.)

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, as described in the following sections.

Next-Hop Resolution of IPv4-Unicast RIB-Ins with IPv4 next-hop

The following table describes the next-hop resolution steps for IPv4-Unicast RIB-Ins with IPv4 next-hops, depending on the specified tunnel resolution mode.

Table 2: Next-hop resolution for IPv4 Unicast RIB-Ins with IPv4 next-hop address

Tunnel Resolution Mode	Next-hop resolution steps in BGP
prefer	<ol style="list-style-type: none"> Start with TTM lookup: <ol style="list-style-type: none"> Find all the tunnels in TTM with an endpoint that matches the BGP next-hop address and that have one of the types listed in the allow list. If there is a single tunnel, select that tunnel. The RIB-IN is resolved; exit. If there are multiple tunnels, select the tunnel with the numerically lowest TTM preference, and if a further tie-break is needed, select the tunnel with the lowest TTM metric. The RIB-IN is resolved; exit. If there are no tunnels, fallback to FIB lookup: <ol style="list-style-type: none"> Find the longest match active route in the FIB that matches the BGP next-hop address. There are presently no restrictions on this route; it can be an IGP route, a static blackhole route, a default route, or another BGP route. If there is a longest match route and it eventually resolves to a blackhole next-hop, interface or tunnel then the RIB-IN is resolved; exit. If there is no matching route the RIB-IN is unresolved.
require	Perform TTM lookup only, as described in 1 above.

Tunnel Resolution Mode	Next-hop resolution steps in BGP
	If there is no matching tunnel, the RIB-IN is unresolved.
disabled	Perform FIB lookup only, as described in 2 above.

Next-Hop Resolution of IPv4-Unicast and IPv6-Unicast RIB-Ins with IPv6 next-hop

If the next-hop address for the IPv4-Unicast RIB-In is an IPv6 address, the next-hop is resolved by the longest prefix match IPv6 route in the FIB. This is the only option because there are no IPv6 tunnels in the TTM. The same logic applies to BGP RIB-Ins with IPv6-unicast NLRl address family as they can only have an IPv6 next-hop address. The next-hop resolution logic is the same as the FIB lookup described in the preceding table.

4.13.1 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 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 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
                  ]
                }
              }
            }
          }
        }
      }
    }
  }
}

```

4.14 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 network-instance default

```

```

network-instance default {
  mtu {
    path-mtu-discovery true
  }
}

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

```

4.14.1 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: Configuring BGP instance `tcp-mss`

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

```

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

```

Example: Configuring BGP peer group `tcp-mss`

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

```

info from state network-instance default protocols bgp group trans
port tcp-mss
network-instance default {
  protocols {
    bgp {
      group test {

```

```

        transport {
            tcp-mss 1024
        }
    }
}

```

Example: Configuring BGP neighbor tcp-mss

The following example configures the BGP neighbor tcp-mss .

```

info 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.

4.15 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.

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. 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.



Note: Service configuration is not yet supported with BGP-LU in the current release.

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.



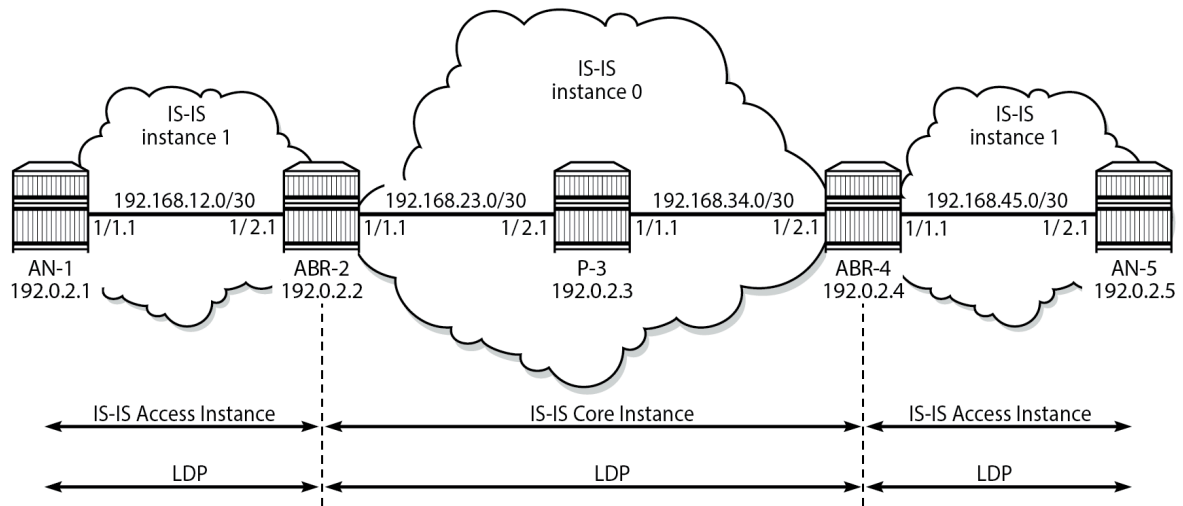
Note: Service configuration is not yet supported with BGP-LU in the current release.

- 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.

5.1 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.

Figure 1: Seamless MPLS - IGP/LDP domains



38582

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
      }
    }
  }
}
```

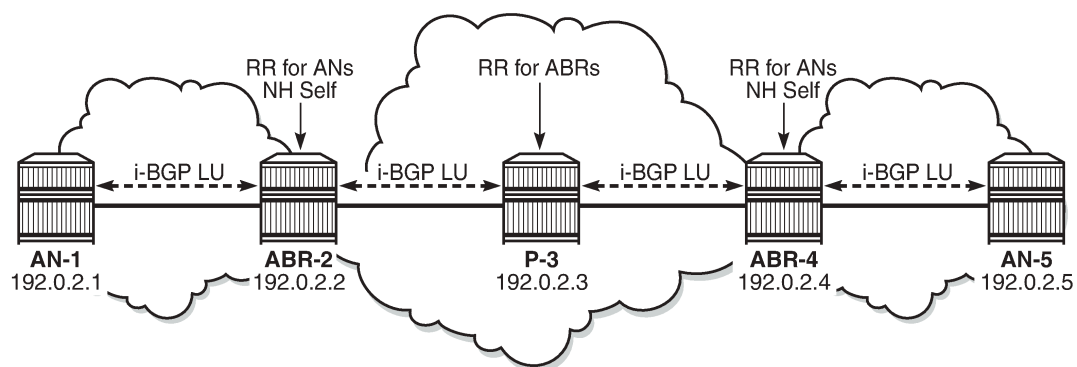

Example: Configure Link LDP (ABR-2)

```
# 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 2: Seamless MPLS – BGP-LU



25637

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 max-path-level-1** to a value greater than 1. The value of **max-path-level-1** 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.

6 IS-IS

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

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

```
--{ * candidate shared default }--[ network-instance default protocols isis ]--
# info detail
instance il {
  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 {
        }
    }
}

```

6.1 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

6.1.1 Enabling an IS-IS instance

Procedure

On 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 network-instance default protocols isis
network-instance default {
    protocols {
        isis {
            instance i1 {
            }
        }
    }
}

```

6.1.2 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 isis
isis {
    instance i1 {
        level-capability L2
    }
}
```

6.1.3 Configuring the Network Entity Title

Procedure

On 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:

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

```

        net 49.0001.1921.6800.1002.00
    }
}

```

6.1.4 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 isis
  isis {
    instance i1 {
      level-capability L2
      overload {
        on-boot {
          timeout 90
        }
      }
      authentication {
        lsp-authentication {
          generate true
          check-received strict
          keychain isisglobal
        }
      }
    }
  }
}

```

6.1.5 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 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 {
  }
}

```

6.1.6 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 network-instance default protocols isis
network-instance default {
  protocols {
    isis {
      instance i1 {
        authentication {
          key {
            crypto-algorithm cleartext
            auth-password $aes$9G3XrtckZzMg=$In9Wu0vKPsTw6ehDX5YLgA==
          }
        }
      }
      interface ethernet-1/1.1 {
        authentication {
          key {

```


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.

6.2.1 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 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 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.

6.3 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: 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: 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-1/16.1 | up       | L1L2 | 4          | broadcast      | 10/10              | 10/10              |
| 1              |          |      |           |               |                   |                   |
| lo0.1          | up        | L1L2 | 5          | broadcast      | 0/0                | 0/0                |
-----
    
```

Example: 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: 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: 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

6.4 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
```

6.5 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-6/10/6e/10e platforms and 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.

6.5.1 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-6/10/6e/10e platforms have a 128 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 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 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.

6.5.2 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: Configuring weighted load-balancing over interface next-hops

```

--{ * candidate shared default }--[ ]--
# info 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.

6.5.3 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 ($\text{max_buckets} - \text{num_nexthops}$) 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.

6.6 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.

6.6.1 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 network-instance default protocols isis instance i1 instance-id
  network-instance default {
    protocols {
      isis {
        instance i1 {
          instance-id 2
        }
      }
    }
  }
}
```

6.6.2 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 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.

6.6.3 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 from state network-instance default protocols isis instance MyInstance2
  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 Protocol authentication

On the 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.

7.1 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 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 e7xdKlY02D0m7v3IJv
      }
    }
  }
}
```

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 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 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 network-instance default protocols bgp group Dut-C--Dut-D authentication
  network-instance default {
    protocols {
      bgp {
        group Dut-C--Dut-D {
          authentication {
            password $aes$9GP/wMALbt+c=$C59v0vqPux/Ue5bTRvVzfQ==
          }
        }
      }
    }
  }
}
```

The following example illustrates the authentication password configuration at neighbor level:

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

```
# info 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=$M0mvT0s2av5CGxBPdjVXuQ==
        }
      }
    }
  }
}
```

8 Routing policies

The SR Linux supports policy-based routing. Policy-based routing controls the size and content of the routing tables, the routes that are advertised, and the best route to take to reach a destination.

Each routing policy has a sequence of rules (called entries or statements) and a default action. Each statement has a numerical sequence identifier that determines its order relative to other statements in that policy. The statement supports both alphanumerical and numerical sequence identifiers. 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 keywords **first/last/before/after** in the **insert** command. If the **insert** command is not used, the statement is added to the end of the policy.

Each policy statement has zero or more match conditions and a base action (either accept or reject); the statement may also have route-modifying actions. A route matches a statement if it meets all of the specified match conditions.

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.

8.1 Creating a routing policy

A routing policy consists of one or more statements that contain the following:

- Match conditions, such as protocol type, AS path length, and address family, against which routes are evaluated
- Actions, such as accept or reject, for routes that match the conditions in the statement

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.

8.1.1 Specifying match conditions

Procedure

You can specify the following match conditions in a policy statement:

- Match routes by their protocol type – BGP, static, direct, host, IS-IS, OSPF, and so on.
- Match routes of any protocol by their address family – IPv4-unicast, IPv6-unicast, EVPN.
- Match routes of any protocol by their IPv4/IPv6 prefix.
- Match aggregate and BGP IPv4/IPv6 routes by their standard and large communities.
- Match BGP routes by their AS path length.
- Match BGP routes by their AS path encoding (using a regular expression).

- Match BGP routes by whether the EVPN route is a specified route type (ethernet-ad, mac-ip, ethernet-segment, imet, or ip-prefix).
- Match IS-IS routes by their association with either Level1 (L1) or Level2 (L2).
- Match IS-IS routes by their route type (internal or external). An IPv4 route is internal if the prefix was signaled in TLV 128. An IPv4 route is external if the prefix was signaled in TLV 130. The encoding of TLV 236 indicates whether an IPv6 route is internal or external.
- Match OSPF routes by their area ID.
- Match OSPF routes by their route type (intra-area, inter-area, type-1-ext, type-2-ext, type-1-nssa, type-2-nssa, summary-aggregate, or nssa-aggregate).
- Match OSPF routes by instance ID.
- Match EVPN IP prefix routes based on protocol bgp-evpn.
- Match routes based on IP addresses and prefixes via prefix-sets for route types 2 and 5.

If a statement has no match conditions defined, all routes evaluated by the policy are considered to be matches.

Example

The following example specifies BGP protocol as a match condition in a policy statement:

```
--{ candidate shared default }--[ ]--
# info routing-policy policy policy01
  routing-policy {
    policy policy01 {
      statement 100 {
        match {
          protocol bgp
        }
      }
    }
  }
}
```

8.1.1.1 Specifying a list as a match condition

Procedure

You can specify a list of items, such as address prefixes, autonomous systems, BGP communities, and Ethernet tags as match criteria. To do this, you configure the list and include the list in a policy statement.

Example

The following example specifies a list of prefixes that can be used as a match condition in a policy statement:

```
--{ candidate shared default }--[ ]--
# info routing-policy
  routing-policy {
    prefix-set western {
      prefix 10.10.0.1/32 mask-length-range exact {
      }
      prefix 10.10.0.2/32 mask-length-range exact {
      }
      prefix 10.10.0.3/32 mask-length-range exact {
      }
    }
  }
}
```



```

        prefix 10.10.0.4/32 mask-length-range exact {
        }
    }
}

```

8.1.2 Specifying actions

Procedure

The following are valid actions for routes that meet the match conditions in a statement:

- Accept the route.
- Reject the route.
- Modify the AS path of a BGP route by prepending additional AS numbers.
- Modify the AS path of a BGP route by deleting the current AS path and replacing it with a new AS path (a sequence of zero or more AS numbers).
- Add, delete, or replace standard and large communities attached to a BGP route.
- Modify the LOCAL_PREF attribute of a BGP route by specifying a new value.
- Modify the ORIGIN attribute of a BGP route by specifying a new value.
- Replace the MED attribute of a BGP route with a new value, either the route-table metric of the route that resolves the BGP next-hop, or a specified MED value.
- Set the next-hop-self option for a BGP route.
- Set the next-hop-unchanged option for a BGP route.
- Add, delete, or replace the route tag associated with an IS-IS route.

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 specify a new value for the ORIGIN attribute of the BGP route. If the route matches the second statement, the action is to accept the route.

```

--{ candidate shared default }--[ ]--
# info routing-policy
routing-policy {
  policy policy02 {
    statement myEntry100 {
      match {
        protocol bgp
      }
      action {
        bgp {
          origin {
            set egp
          }
        }
      }
    }
  }
  statement 101 {
    match {
      prefix-set western
    }
    action {
      policy-result accept
    }
  }
}

```

```

    }
  }
}

```

8.1.2.1 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.

8.1.3 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 routing-policy
  routing-policy {
    policy policy01 {
```

```

        default-action {
            policy-result reject
        }
    }
}

```

8.2 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

The following example specifies that BGP in the default network-instance applies `policy01` to imported routes:

```

--{ candidate shared default }--[ ]--
# info network-instance default
network-instance default {
    protocols {
        bgp {
            import-policy policy01
        }
    }
}

```

Example

The following example applies `policy02` to BGP routes exported from the peers in peer-group `headquarters1`:

```

--{ candidate shared default }--[ ]--
# info network-instance default
network-instance default {
    protocols {
        bgp {
            group headquarters1 {
                export-policy policy02
            }
        }
    }
}

```

Example

The following example applies `policy02` to BGP routes exported from a specific BGP neighbor:

```

--{ * candidate shared default }--[ ]--
# info network-instance default
network-instance default {
    protocols {
        bgp {
            neighbor 192.168.11.1 {
                export-policy policy02
            }
        }
    }
}

```

```

    }
  }
}

```

8.2.1 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 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
          }
        }
      }
    }
  }
}

```

8.2.2 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 whose action is to replace the AS path in matching routes.

```
--{ candidate shared default }--[ ]--
# info routing-policy
routing-policy {
  policy policy02 {
    statement 100 {
      match {
        protocol bgp
      }
      action {
        bgp {
          as-path {
            replace [
              12
              13
              14
            ]
          }
        }
      }
    }
  }
}
```

8.2.3 BGP community-sets

You can configure a BGP community-set that can be used as a match condition in a routing policy. A BGP community-set contains a list of member elements and a match-set-option that indicates what constitutes a matching route.

An element in a BGP community-set can be a standard BGP community value, regular expression, or well-known name, or else a large BGP community value or regular expression (see [Table 3: Supported community-set member formats](#)). The match-set-option 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.

Table 3: Supported community-set member formats

String format	Definition
<0-65535>:<0-65535>	Standard community matched exactly
<0-4294967295>:<0-4294967295>:<0-4294967295>	Large community matched exactly
<regex1>:<regex2>	Standard community matched by two regular expressions

String format	Definition
<regex1>:<regex2>:<regex3>	Large community matched by three regular expressions
bgp-tunnel-encap:(MPLS VXLAN)	BGP tunnel encapsulation extended community
color:NN:<0-4294967295>	Color extended community matched exactly
no-advertise	Well-defined standard community 'NO-ADVERTISE'
no-export	Well-defined standard community 'NO-EXPORT'
no-export-subconfed	Well-defined standard community 'NO-EXPORT-SUBCONFED'
no-peer	Well-defined standard community 'NO-PEER'
origin:<0-65535>:<0-4294967295>	Type0 route-origin extended community matched exactly
origin:<0-4294967295>:<0-65535>	Type2 route-origin extended community matched exactly
origin:<ipv4-address>:<0-65535>	Type1 route-origin extended community matched exactly
origin:<regex1>:<regex2>	Route-origin extended community matched by two regular expressions
target:<0-65535>:<0-4294967295>	type0 route-target extended community matched exactly
target:<0-4294967295>:<0-65535>	Type2 route-target extended community matched exactly
target:<ipv4-address>:<0-65535>	Type1 route-target extended community matched exactly
target:<regex1>:<regex2>	Route-target extended community matched by two regular expressions

8.2.3.1 Configuring a BGP community-set

Procedure

To configure a BGP 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 BGP 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 routing-policy
routing-policy {
  community-set cs1 {
    match-set-options invert
    member [
      no-advertise
      no-export
    ]
  }
  policy p1 {
    statement 1 {
      match {
        bgp {
          community-set cs1
        }
      }
    }
  }
}
```


Customer document and product support



Customer documentation

[Customer documentation welcome page](#)



Technical support

[Product support portal](#)



Documentation feedback

[Customer documentation feedback](#)