Implementing BGP

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

This module provides the conceptual and configuration information for BGP on Cisco IOS XR software.

Note For more information about BGP and complete descriptions of the BGP commands listed in this module, see Related Documents, on page 289 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco ASR 9000 Series Router software master command index.

Feature History for Implementing BGP

Release	Modification
Release 3.7.2	This feature was introduced.
Release 3.9.0	The following features were supported: • BGP Prefix Independent Convergence Unipath Primary Backup • BGP Local Label Retention • Asplain notation for 4-byte Autonomous System Number • BGP Nonstop Routing • Command Line Interface (CLI) consistency for BGP commands • L2VPN Address Family Configuration Mode

Release	Modification
Release 4.0.0	The following features were supported: • BGP Add Path Advertisement • Accumulated iGP (AiGP) • Pre-route • IPv4 BGP-Policy Accounting • IPv6 uRPF
Release 4.1.0	Support for 5000 BGP NSR sessions was added
Release 4.1.1	The following features were added: • BGP Accept Own • BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing
Release 4.2.0	The following features were supported: • Selective VRF Download • BGP Multi-Instance/Multi-AS • BFD Multihop Support for BGP • BGP Error Handling Support for Distributed BGP (bgp distributed speaker) configuration was removed.
Release 4.2.1	The following features were supported: • BGP Prefix Independent Convergence for RIB and FIB • BGP Prefix Origin Validation Based on RPKI
Release 4.2.3	The BGP Attribute Filtering feature was added.
Release 4.3.0	The BGP-RIB Feedback Mechanism for Update Generation feature was added
Release 4.3.1	The following features were supported • BGP VRF Dynamic Route Leaking The **label-allocation-mode** command is renamed the **label mode** command.
Release 4.3.2	The following features were supported: • Per-neighbor Link Bandwidth

Release	Modification
Release 5.3.1	The following features were supported: • L3VPN iBGP-PE-CE configuration • Source-based flow tag • Discard extra paths
Release 5.3.2	The following features were supported: • Graceful Maintenance • Per Neighbor TCP MSS • BGP DMZ Aggregate Bandwidth
Release 6.0.1	The 64-ECMP for BGP feature is supported.
Release 7.4.1	The **label-allocation-mode** is deprecated, the function of this deprecated command can be carried out using label mode command under configured address-family.

Prerequisites for Implementing BGP

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing BGP

To implement BGP, you need to understand the following concepts:

BGP Functional Overview

BGP uses TCP as its transport protocol. Two BGP routers form a TCP connection between one another (peer routers) and exchange messages to open and confirm the connection parameters.

BGP routers exchange network reachability information. This information is mainly an indication of the full paths (BGP autonomous system numbers) that a route should take to reach the destination network. This information helps construct a graph that shows which autonomous systems are loop free and where routing policies can be applied to enforce restrictions on routing behavior.

Any two routers forming a TCP connection to exchange BGP routing information are called peers or neighbors. BGP peers initially exchange their full BGP routing tables. After this exchange, incremental updates are sent as the routing table changes. BGP keeps a version number of the BGP table, which is the same for all of its BGP peers. The version number changes whenever BGP updates the table due to routing information changes. Keepalive packets are sent to ensure that the connection is alive between the BGP peers and notification packets are sent in response to error or special conditions.

Note Other than enabling RTC (route target constraint) with `address-family ipv4 rtfilter` command, there is no separate configuration needed to enable RTC for BGP EVPN.

Note For information on configuring BGP to distribute Multiprotocol Label Switching (MPLS) Layer 3 virtual private network (VPN) information, see the *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*

For information on BGP support for Bidirectional Forwarding Detection (BFD), see the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Configuration Guide* and the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Command Reference.*

BGP Router Identifier

For BGP sessions between neighbors to be established, BGP must be assigned a router ID. The router ID is sent to BGP peers in the OPEN message when a BGP session is established.

BGP attempts to obtain a router ID in the following ways (in order of preference):

- By means of the address configured using the **bgp router-id** command in router configuration mode.
- By using the highest IPv4 address on a loopback interface in the system if the router is booted with saved loopback address configuration.
- By using the primary IPv4 address of the first loopback address that gets configured if there are not any in the saved configuration.

If none of these methods for obtaining a router ID succeeds, BGP does not have a router ID and cannot establish any peering sessions with BGP neighbors. In such an instance, an error message is entered in the system log, and the **show bgp summary** command displays a router ID of 0.0.0.0.

After BGP has obtained a router ID, it continues to use it even if a better router ID becomes available. This usage avoids unnecessary flapping for all BGP sessions. However, if the router ID currently in use becomes invalid (because the interface goes down or its configuration is changed), BGP selects a new router ID (using the rules described) and all established peering sessions are reset.

Note We strongly recommend that the **bgp router-id** command is configured to prevent unnecessary changes to the router ID (and consequent flapping of BGP sessions).

BGP Maximum Prefix - Discard Extra Paths

IOS XR BGP maximum-prefix feature imposes a maximum limit on the number of prefixes that are received from a neighbor for a given address family. Whenever the number of prefixes received exceeds the maximum number configured, the BGP session is terminated after sending a cease notification to the neighbor. The session is down until a manual clear is performed by the user. The session can be resumed by using the **clear bgp** command. It is possible to configure a period after which the session can be automatically brought up by using the **maximum-prefix** command with the **restart** keyword.

Discard Extra Paths

An option to discard extra paths is added to the maximum-prefix configuration. Configuring the discard extra paths option drops all excess prefixes received from the neighbor when the prefixes exceed the configured maximum value. This drop does not, however, result in session flap.

The benefits of discard extra paths option are:

- Limits the memory footstamp of BGP.
- Stops the flapping of the peer if the paths exceed the set limit.

When the discard extra paths configuration is removed, BGP sends a route-refresh message to the neighbor if it supports the refresh capability; otherwise the session is flapped.

On the same lines, the following describes the actions when the maximum prefix value is changed:

- If the maximum value alone is changed, a route-refresh message is sourced, if applicable.
- If the new maximum value is greater than the current prefix count state, the new prefix states are saved.
- If the new maximum value is less than the current prefix count state, then some existing prefixes are deleted to match the new configured state value.

There is currently no way to control which prefixes are deleted.

For detailed configuration steps, see Configuring Discard Extra Paths, on page 142.

Note When the system runs out of physical memory, bgp process exits and you must manually restart bpm. To manually restart, use the **process restart bpm** command.

Restrictions

These restrictions apply to the discard extra paths feature:

- When the router drops prefixes, it is inconsistent with the rest of the network, resulting in possible routing loops.
- If prefixes are dropped, the standby and active BGP sessions may drop different prefixes. Consequently, an NSR switchover results in inconsistent BGP tables.
- The discard extra paths configuration cannot co-exist with the *soft reconfig* configuration.

BGP Enhanced Multipath Selection

Table 1: Feature History Table

Feature Name	Release Name	Description
BGP Enhanced Multipath Selection	Release 7.4.2	This feature gives you the flexibility to select unequal cost multipath (UCMP) load-balancing based on the interior gateway protocol (IGP) route metric. The IGP route metric is the sum of the metrics of all the links that belong to a path, and this feature selects the paths with lower IGP route metrics as multipath. In earlier releases, you could select BGP UCMP only based on age order, where the older path took precedence over the newer path.

The BGP multipath selection algorithm functionality enables the multipath to prefer the older paths over the new paths. Here the age order is a vital criterion for selection of the UCMP. However, this method is less optimal and is nondeterministic in terms of forwarding traffic on the network. The BGP Enhanced Multipath Selection feature allows the multipath functionality to select IGP metric.

Restrictions

- This feature is available in Internal Border Gateway Protocol.
- This feature is configurable on the following address families:
  - IPv4 Unicast
  - IPv6 Unicast
  - IPv4 Multicast
  - IPv6 Multicast

- VPNv4 does not support maximum-paths, so you cannot configure the deterministic aspect in the VPN address-family interfaces. However, you can configure the imported prefixes of VRFs with this feature.

Configuration Example

```
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# maximum-paths ibgp 2 unequal-cost deterministic
```

Running Configuration

```
router bgp 100
 address-family ipv4 unicast
  maximum-paths ibgp 2 unequal-cost deterministic
```

Verification

The following example shows you can select paths with the lower metrics as multipaths.

```
Router# show bgp ipv4 unicast 10.10.0.0/28
Paths: (128 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    22.0.1.6 (metric 20) from 198.51.100.1 (192.0.0.1)
      Origin IGP, localpref 0, valid, internal, best, group-best, multipath
      Received Path ID 1, Local Path ID 1, version 12611
      Originator: 192.0.0.1, Cluster list: 198.51.100.1
…

Path #64: Received by speaker 0
  Not advertised to any peer
  Local
    23.0.11.6 (metric 30) from 203.0.113.1 (210.0.0.10)
      Origin IGP, localpref 0, valid, internal, multipath
      Received Path ID 32, Local Path ID 0, version 0
      Originator: 210.0.0.10, Cluster list: 203.0.113.1, 200.0.0.10

Path #65: Received by speaker 0
  Not advertised to any peer
  Local
    24.0.1.6 (metric 40) from 192.0.2.254 (211.0.0.0)
      Origin IGP, localpref 0, valid, internal
      Received Path ID 1, Local Path ID 0, version 0
      Originator: 211.0.0.0, Cluster list: 192.0.2.254, 201.0.0.0, 202.0.0.0
…

  Path #128: Received by speaker 0
  Not advertised to any peer
  Local
    25.0.23.6 (metric 50) from 198.51.100.233 (195.0.0.23)
      Origin IGP, localpref 0, valid, internal
      Received Path ID 32, Local Path ID 0, version 0
      Originator: 195.0.0.23, Cluster list: 198.51.99.255
```

The following example displays the BGP multipaths installed in the RIB.

```
Router# show route ipv4 200.0.0.0/28
Routing entry for 200.0.0.0/28
```

```
Known via "bgp 1", distance 200, metric 0, type internal
Installed Oct 17 04:06:41.027 for 00:01:22
outing Descriptor Blocks
   10.0.1.6, from 198.51.100.1, BGP multi path
     Route metric is 0
   10.0.2.6, from 198.51.100.1, BGP multi path
     Route metric is 0
   …
   10.0.32.6, from 198.51.100.1, BGP multi path
     Route metric is 0
   198.51.100.253, from 203.0.113.1, BGP multi path
     Route metric is 0
   …
   198.51.100.252, from 203.0.113.1, BGP multi path
     Route metric is 0
 No advertising protos.
```

The following example displays the BGP multipaths installed in Cisco Express Forwarding.

```
Router# show cef ipv4 200.0.0.0/28 detail
Level 1 - Load distribution: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
 54 55 56 57 58 59 60 61 62 63
 [0] via 10.0.1.6/32, recursive
 [1] via 10.0.2.6/32, recursive

 [62] via 203.0.113.211/32, recursive
 [63] via 203.0.112.211/32, recursive

  via 10.0.1.6/32, 257 dependencies, recursive, bgp-multipath [flags 0x6080]
  path-idx 0 NHID 0x0 [0x7a6cdf90 0x0]
  next hop 22.0.1.6/32 via 22.0.0.0/8

  Load distribution: 0 (refcount 1)
  Hash  OK  Interface              Address
  0     Y   TenGigE0/1/0/0/7       remote

 via 203.0.113.211/32, 257 dependencies, recursive, bgp-multipath [flags 0x6080]
 path-idx 62 NHID 0x0 [0x7a6ce058 0x0]
  next hop 203.0.112.211/32 via 203.0.0.0/8

  Load distribution: 0 (refcount 1)

  Hash  OK  Interface                 Address
  2     Y   TenGigE0/1/0/0/4          remote

  via 203.0.32.6/32, 257 dependencies, recursive, bgp-multipath [flags 0x6080]
  path-idx 63 NHID 0x0 [0x7a6ce058 0x0]
  next hop 203.0.32.6/32 via 203.0.0.0/8

  Load distribution: 0 (refcount 1)

  Hash  OK  Interface                 Address
  63    Y   TenGigE0/1/0/0/4          remote
```

BGP Next Hop Tracking

BGP receives notifications from the Routing Information Base (RIB) when next-hop information changes (event-driven notifications). BGP obtains next-hop information from the RIB to:

- Determine whether a next hop is reachable.
- Find the fully recursed IGP metric to the next hop (used in the best-path calculation).

- Validate the received next hops.
- Calculate the outgoing next hops.
- Verify the reachability and connectedness of neighbors.

BGP is notified when any of the following events occurs:

- Next hop becomes unreachable
- Next hop becomes reachable
- Fully recursed IGP metric to the next hop changes
- First hop IP address or first hop interface change
- Next hop becomes connected
- Next hop becomes unconnected
- Next hop becomes a local address
- Next hop becomes a nonlocal address

Note Reachability and recursed metric events trigger a best-path recalculation.

Event notifications from the RIB are classified as critical and noncritical. Notifications for critical and noncritical events are sent in separate batches. However, a noncritical event is sent along with the critical events if the noncritical event is pending and there is a request to read the critical events.

- Critical events are related to the reachability (reachable and unreachable), connectivity (connected and unconnected), and locality (local and nonlocal) of the next hops. Notifications for these events are not delayed.
- Noncritical events include only the IGP metric changes. These events are sent at an interval of 3 seconds. A metric change event is batched and sent 3 seconds after the last one was sent.

The next-hop trigger delay for critical and noncritical events can be configured to specify a minimum batching interval for critical and noncritical events using the **nexthop trigger-delay** command. The trigger delay is address family dependent.

The BGP next-hop tracking feature allows you to specify that BGP routes are resolved using only next hops whose routes have the following characteristics:

- To avoid the aggregate routes, the prefix length must be greater than a specified value.
- The source protocol must be from a selected list, ensuring that BGP routes are not used to resolve next hops that could lead to oscillation.

This route policy filtering is possible because RIB identifies the source protocol of route that resolved a next hop as well as the mask length associated with the route. The **nexthop route-policy** command is used to specify the route-policy.

For information on route policy filtering for next hops using the next-hop attach point, see the *Implementing Routing Policy Language on* Cisco ASR 9000 Series Router module of *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* (this publication).

Scoped IPv4/VPNv4 Table Walk

To determine which address family to process, a next-hop notification is received by first de-referencing the gateway context associated with the next hop, then looking into the gateway context to determine which address families are using the gateway context. The IPv4 unicast and VPNv4 unicast address families share the same gateway context, because they are registered with the IPv4 unicast table in the RIB. As a result, both the global IPv4 unicast table and the VPNv4 table are is processed when an IPv4 unicast next-hop notification is received from the RIB. A mask is maintained in the next hop, indicating if whether the next hop belongs to IPv4 unicast or VPNv4 unicast, or both. This scoped table walk localizes the processing in the appropriate address family table.

Reordered Address Family Processing

The Cisco IOS XR software walks address family tables based on the numeric value of the address family. When a next-hop notification batch is received, the order of address family processing is reordered to the following order:

- IPv4 tunnel
- VPNv4 unicast
- IPv4 labeled unicast
- IPv4 unicast
- IPv4 multicast
- IPv6 unicast

New Thread for Next-Hop Processing

The critical-event thread in the spkr process handles only next-hop, Bidirectional Forwarding Detection (BFD), and fast-external-failover (FEF) notifications. This critical-event thread ensures that BGP convergence is not adversely impacted by other events that may take a significant amount of time.

show, clear, and debug Commands

The **show bgp nexthops** command provides statistical information about next-hop notifications, the amount of time spent in processing those notifications, and details about each next hop registered with the RIB. The **clear bgp nexthop performance-statistics** command ensures that the cumulative statistics associated with the processing part of the next-hop **show** command can be cleared to help in monitoring. The **clear bgp nexthop registration** command performs an asynchronous registration of the next hop with the RIB. See the *BGP Commands on* Cisco ASR 9000 Series Router module of *Routing Command Reference for Cisco ASR 9000 Series Routers*for information on the next-hop **show** and **clear** commands.

The **debug bgp nexthop** command displays information on next-hop processing. The **out** keyword provides debug information only about BGP registration of next hops with RIB. The **in** keyword displays debug information about next-hop notifications received from RIB. The **out** keyword displays debug information about next-hop notifications sent to the RIB. See the *BGP Debug Commands on Cisco ASR 9000 Series Aggregation Services Router* module of *Cisco ASR 9000 Series Aggregation Services Router Routing Debug Command Reference*.

Autonomous System Number Formats in BGP

Autonomous system numbers (ASNs) are globally unique identifiers used to identify autonomous systems (ASs) and enable ASs to exchange exterior routing information between neighboring ASs. A unique ASN is allocated to each AS for use in BGP routing. ASNs are encoded as 2-byte numbers and 4-byte numbers in BGP.

```
RP/0/RP0/CPU0:router(config)# as-format [asdot | asplain]
RP/0/RP0/CPU0:router(config)# as-format asdot
```

Note ASN change for BGP process is not currently supported via **commit replace**command.

2-byte Autonomous System Number Format

The 2-byte ASNs are represented in asplain notation. The 2-byte range is 1 to 65535.

4-byte Autonomous System Number Format

To prepare for the eventual exhaustion of 2-byte Autonomous System Numbers (ASNs), BGP has the capability to support 4-byte ASNs. The 4-byte ASNs are represented both in asplain and asdot notations.

The byte range for 4-byte ASNs in asplain notation is 1-4294967295. The AS is represented as a 4-byte decimal number. The 4-byte ASN asplain representation is defined in draft-ietf-idr-as-representation-01.txt.

For 4-byte ASNs in asdot format, the 4-byte range is 1.0 to 65535.65535 and the format is:

high-order-16-bit-value-in-decimal . *low-order-16-bit-value-in-decimal*

The BGP 4-byte ASN capability is used to propagate 4-byte-based AS path information across BGP speakers that do not support 4-byte AS numbers. See draft-ietf-idr-as4bytes-12.txt for information on increasing the size of an ASN from 2 bytes to 4 bytes. AS is represented as a 4-byte decimal number

as-format Command

The **as-format** command configures the ASN notation to asdot. The default value, if the **as-format** command is not configured, is asplain.

BGP Configuration

BGP in Cisco IOS XR software follows a neighbor-based configuration model that requires that all configurations for a particular neighbor be grouped in one place under the neighbor configuration. Peer groups are not supported for either sharing configuration between neighbors or for sharing update messages. The concept of peer group has been replaced by a set of configuration groups to be used as templates in BGP configuration and automatically generated update groups to share update messages between neighbors.

Configuration Modes

BGP configurations are grouped into modes. The following sections show how to enter some of the BGP configuration modes. From a mode, you can enter the **?** command to display the commands available in that mode.

Router Configuration Mode

The following example shows how to enter router configuration mode:

```
RP/0/RSP0/CPU0:router# configuration
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)#
```

Router Address Family Configuration Mode

The following example shows how to enter router address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

Neighbor Configuration Mode

The following example shows how to enter neighbor configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Defining Source Address for Update-Source Interface

The following configuration defines the source address for Update-Source Interface.

```
RP/0/RSP0/CPU0:router# show run interface Bundle-Ether81
interface Bundle-Ether81
ipv6 address 2001:db8:19:200::1/48
ipv6 address 2001:db8:2:b0::d1/126
RP/0/RSP0/CPU0:router(config)# router bgp 4230
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 2001:db8:4::5043:432b
RP/0/RSP0/CPU0:router(config-bgp)# local address 2001:db8:2:b0::d1
RP/0/RSP0/CPU0:router(config-bgp)# update-source Bundle-Ether81
RP/0/RSP0/CPU0:router(config-bgp)# commit
```

Running Configuration

```
interface Bundle-Ether81
ipv6 address 2001:db8:19:200::1/48
ipv6 address 2001:db8:2:b0::d1/126
!
router bgp 4230
neighbor 2001:db8:4::5043:432b
local address 2001:db8:2:b0::d1
update-source Bundle-Ether81
!
```

Neighbor Address Family Configuration Mode

The following example shows how to enter neighbor address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
```

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

VRF Configuration Mode

The following example shows how to enter VPN routing and forwarding (VRF) configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)#
```

VRF Address Family Configuration Mode

The following example shows how to enter VRF address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#
```

Configuring Resilient Per-CE Label Mode Under VRF Address Family

Perform this task to configure resilient per-ce label mode under VRF address family.

Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**
2. **router bgp***as-number*
3. **vrf***vrf-instance*
4. **address-family {ipv4 | ipv6} unicast**
5. **label mode per-ce**
6. Do one of the following:
   - **end**
   - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)#
```

Enters global configuration mode.

Step 2 **router bgp***as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 666
RP/0/RSP0/CPU0:router(config-bgp)#
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **vrf***vrf-instance*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf-pe
RP/0/RSP0/CPU0:router(config-bgp-vrf)#
```

Configures a VRF instance.

Step 4 **address-family {ipv4 | ipv6} unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#
```

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

Step 5 **label mode per-ce**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# label mode per-ce
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#
```

Configures resilient per-ce label mode.

Step 6 Do one of the following:

- **end**
- **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# end
```

or

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Resilient Per-CE Label Mode Using a Route-Policy

Perform this task to configure resilient per-ce label mode using a route-policy.

Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**
2. **route-policy***policy-name*
3. **set label mode per-ce**
4. Do one of the following:
   - **end**
   - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)#
```

Enters global configuration mode.

Step 2 **route-policy***policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# route-policy route1
RP/0/RSP0/CPU0:router(config-rpl)#
```

Creates a route policy and enters route policy configuration mode.

Step 3 **set label mode per-ce**

Example:

```
RP/0/RSP0/CPU0:router(config-rpl)# set label mode per-ce
RP/0/RSP0/CPU0:router(config-rpl)#
```

Configures resilient per-ce label mode.

Step 4 Do one of the following:

- **end**
- **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-rpl)# end
```

or

```
RP/0/RSP0/CPU0:router(config-rpl)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

VRF Neighbor Configuration Mode

The following example shows how to enter VRF neighbor configuration mode:

```
Router(config)# router bgp 140
Router(config-bgp)# vrf vrf_A
Router(config-bgp-vrf)# neighbor 11.0.1.2
Router(config-bgp-vrf-nbr)#
```

VRF Neighbor Address Family Configuration Mode

The following example shows how to enter VRF neighbor address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 112
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 11.0.1.2
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
```

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)#
```

VPNv4 Address Family Configuration Mode

The following example shows how to enter VPNv4 address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 152
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

L2VPN Address Family Configuration Mode

The following example shows how to enter L2VPN address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn vpls-vpws
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

Neighbor Submode

Cisco IOS XR BGP uses a neighbor submode to make it possible to enter configurations without having to prefix every configuration with the **neighbor** keyword and the neighbor address:

- Cisco IOS XR software has a submode available for neighbors in which it is not necessary for every command to have a "neighbor *x.x.x.x*" prefix:

  In Cisco IOS XR software, the configuration is as follows:

  ```
  RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.23.1.2
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
  ```

- An address family configuration submode inside the neighbor configuration submode is available for entering address family-specific neighbor configurations. In Cisco IOS XR software, the configuration is as follows:

  ```
  RP/0/RSP0/CPU0:router(config-bgp)# neighbor 2002::2
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2023
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast
  RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self
  RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy one in
  ```

- You must enter neighbor-specific IPv4, IPv6, VPNv4, or VPNv6 commands in neighbor address-family configuration submode. In Cisco IOS XR software, the configuration is as follows:

  ```
  RP/0/RSP0/CPU0:router(config)# router bgp 109
  RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
  RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
  RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000
  ```

- You must enter neighbor-specific IPv4 and IPv6 commands in VRF neighbor address-family configuration submode. In Cisco IOS XR software, the configuration is as follows:

```
RP/0/RSP0/CPU0:router(config)# router bgp 110
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 11.0.1.2
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass all in
```

Configuration Templates

The **af-group**, **session-group**, and **neighbor-group** configuration commands provide template support for the neighbor configuration in Cisco IOS XR software.

The **af-group** command is used to group address family-specific neighbor commands within an IPv4, IPv6, or VPNv4, address family. Neighbors that have the same address family configuration are able to use the address family group (af-group) name for their address family-specific configuration. A neighbor inherits the configuration from an address family group by way of the **use** command. If a neighbor is configured to use an address family group, the neighbor (by default) inherits the entire configuration from the address family group. However, a neighbor does not inherit all of the configuration from the address family group if items are explicitly configured for the neighbor. The address family group configuration is entered under the BGP router configuration mode. The following example shows how to enter address family group configuration mode.

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# af-group afmcast1 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)#
```

The **session-group** command allows you to create a session group from which neighbors can inherit address family-independent configuration. A neighbor inherits the configuration from a session group by way of the **use** command. If a neighbor is configured to use a session group, the neighbor (by default) inherits the entire configuration of the session group. A neighbor does not inherit all of the configuration from a session group if a configuration is done directly on that neighbor. The following example shows how to enter session group configuration mode:

```
RP/0/RSP0/CPU0:router# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# session-group session1
RP/0/RSP0/CPU0:router(config-bgp-sngrp)#
```

The **neighbor-group** command helps you apply the same configuration to one or more neighbors. Neighbor groups can include session groups and address family groups and can comprise the complete configuration for a neighbor. After a neighbor group is configured, a neighbor can inherit the configuration of the group using the **use** command. If a neighbor is configured to use a neighbor group, the neighbor inherits the entire BGP configuration of the neighbor group.

The following example shows how to enter neighbor group configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 123
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)#
```

The following example shows how to enter neighbor group address family configuration mode:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)#
```

- However, a neighbor does not inherit all of the configuration from the neighbor group if items are explicitly configured for the neighbor. In addition, some part of the configuration of the neighbor group could be hidden if a session group or address family group was also being used.

Configuration grouping has the following effects in Cisco IOS XR software:

- Commands entered at the session group level define address family-independent commands (the same commands as in the neighbor submode).
- Commands entered at the address family group level define address family-dependent commands for a specified address family (the same commands as in the neighbor-address family configuration submode).
- Commands entered at the neighbor group level define address family-independent commands and address family-dependent commands for each address family (the same as all available **neighbor** commands), and define the **use** command for the address family group and session group commands.

Template Inheritance Rules

In Cisco IOS XR software, BGP neighbors or groups inherit configuration from other configuration groups.

For address family-independent configurations:

- Neighbors can inherit from session groups and neighbor groups.
- Neighbor groups can inherit from session groups and other neighbor groups.
- Session groups can inherit from other session groups.
- If a neighbor uses a session group and a neighbor group, the configurations in the session group are preferred over the global address family configurations in the neighbor group.

For address family-dependent configurations:

- Address family groups can inherit from other address family groups.
- Neighbor groups can inherit from address family groups and other neighbor groups.
- Neighbors can inherit from address family groups and neighbor groups.

Configuration group inheritance rules are numbered in order of precedence as follows:

1. If the item is configured directly on the neighbor, that value is used. In the example that follows, the advertisement interval is configured both on the neighbor group and neighbor configuration and the advertisement interval being used is from the neighbor configuration:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.1.1.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
```

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# advertisement-interval 20
```

The **show bgp neighbor** output shows the cumulative number for the *Prefix advertised* count if the same prefixes are withdrawn and re-advertised.

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 20 seconds:

```
RP/0/RSP0/CPU0:router# show bgp neighbors 10.1.1.1

BGP neighbor is 10.1.1.1, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 20 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:00:14, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

2. Otherwise, if an item is configured to be inherited from a session-group or neighbor-group and on the neighbor directly, then the configuration on the neighbor is used. If a neighbor is configured to be inherited from session-group or af-group, but no directly configured value, then the value in the session-group or af-group is used. In the example that follows, the advertisement interval is configured on a neighbor group and a session group and the advertisement interval value being used is from the session group:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 20
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use session-group AS_2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.0.1

BGP neighbor is 192.168.0.1, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
```

```
 Sent 0 messages, 0 notifications, 0 in queue
 Minimum time between advertisement runs is 15 seconds

For Address Family: IPv4 Unicast
 BGP neighbor version 0
 Update group: 0.1
 eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
 Route refresh request: received 0, sent 0
 0 accepted prefixes
 Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
 Threshold for warning message 75%

 Connections established 0; dropped 0
 Last reset 00:03:23, due to BGP neighbor initialized
 External BGP neighbor not directly connected.
```

3. Otherwise, if the neighbor uses a neighbor group and does not use a session group or address family group, the configuration value can be obtained from the neighbor group either directly or through inheritance. In the example that follows, the advertisement interval from the neighbor group is used because it is not configured directly on the neighbor and no session group is used:

```
RP/0/RSP0/CPU0:router(config)# router bgp 150
RP/0/RSP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 20
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.1.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
 RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.1.1

BGP neighbor is 192.168.2.2, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  Inbound path policy configured
  Policy for incoming advertisements is POLICY_1
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:01:14, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

To illustrate the same rule, the following example shows how to set the advertisement interval to 15 (from the session group) and 25 (from the neighbor group). The advertisement interval set in the session group overrides the one set in the neighbor group. The inbound policy is set to POLICY_1 from the neighbor group.

```
RP/0/RSP0/CPU0:routerconfig)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# session-group ADV
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group ADV_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 25
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy POLICY_1 in
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.2.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use session-group ADV
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group ADV_2
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 15 seconds:

```
RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.2.2

BGP neighbor is 192.168.2.2, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 15 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.1
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%

  Connections established 0; dropped 0
  Last reset 00:02:03, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

4. Otherwise, the default value is used. In the example that follows, neighbor 10.0.101.5 has the minimum time between advertisement runs set to 30 seconds (default) because the neighbor is not configured to use the neighbor configuration or the neighbor group configuration:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group adv_15
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 10
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
```

```
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.101.5
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.101.10
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group adv_15
```

The following output from the **show bgp neighbors** command shows that the advertisement interval used is 30 seconds:

```
RP/0/RSP0/CPU0:router# show bgp neighbors 10.0.101.5

BGP neighbor is 10.0.101.5, remote AS 1, local AS 140, external link
 Remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:00, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 30 seconds

 For Address Family: IPv4 Unicast
  BGP neighbor version 0
  Update group: 0.2
  eBGP neighbor with no inbound or outbound policy; defaults to 'drop'
  Route refresh request: received 0, sent 0
  0 accepted prefixes
  Prefix advertised 0, suppressed 0, withdrawn 0, maximum limit 524288
  Threshold for warning message 75%
Connections established 0; dropped 0
  Last reset 00:00:25, due to BGP neighbor initialized
  External BGP neighbor not directly connected.
```

The inheritance rules used when groups are inheriting configuration from other groups are the same as the rules given for neighbors inheriting from groups.

Viewing Inherited Configurations

You can use the following **show** commands to view BGP inherited configurations:

show bgp neighbors

Use the **show bgp neighbors** command to display information about the BGP configuration for neighbors.

- Use the **configuration** keyword to display the effective configuration for the neighbor, including any settings that have been inherited from session groups, neighbor groups, or address family groups used by this neighbor.
- Use the **inheritance** keyword to display the session groups, neighbor groups, and address family groups from which this neighbor is capable of inheriting configuration.

The **show bgp neighbors** command examples that follow are based on this sample configuration:

```
RP/0/RSP0/CPU0:router(config)# router bgp 142
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# next-hop-self
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
```

```
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# ebgp-multihop 3
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# send-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit

RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# use af-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# weight 200
```

The following example displays sample output from the **show bgp neighbors** command using the **inheritance** keyword. The example shows that the neighbor inherits session parameters from neighbor group GROUP_1, which in turn inherits from session group GROUP_2. The neighbor inherits IPv4 unicast parameters from address family group GROUP_3 and IPv4 multicast parameters from neighbor group GROUP_1:

```
RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.0.1 inheritance

  Session:        n:GROUP_1 s:GROUP_2
  IPv4 Unicast:   a:GROUP_3
  IPv4 Multicast: n:GROUP_1
```

The following example displays sample output from the **show bgp neighbors** command using the **configuration** keyword. The example shows from where each item of configuration was inherited, or if it was configured directly on the neighbor (indicated by [ ]). For example, the **ebgp-multihop 3** command was inherited from neighbor group GROUP_1 and the **next-hop-self** command was inherited from the address family group GROUP_3:

```
RP/0/RSP0/CPU0:router# show bgp neighbors 192.168.0.1 configuration

neighbor 192.168.0.1
 remote-as 2                    []
 advertisement-interval 15      [n:GROUP_1 s:GROUP_2]
 ebgp-multihop 3                [n:GROUP_1]
 address-family ipv4 unicast    []
  next-hop-self                 [a:GROUP_3]
  route-policy POLICY_1    in   [a:GROUP_3]
  weight 200                    []
 address-family ipv4 multicast [n:GROUP_1]
  default-originate             [n:GROUP_1]
```

show bgp af-group

Use the **show bgp af-group** command to display address family groups:

- Use the **configuration** keyword to display the effective configuration for the address family group, including any settings that have been inherited from address family groups used by this address family group.

- Use the **inheritance** keyword to display the address family groups from which this address family group is capable of inheriting configuration.
- Use the **users** keyword to display the neighbors, neighbor groups, and address family groups that inherit configuration from this address family group.

The **show bgp af-group** sample commands that follow are based on this sample configuration:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_1 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# maximum-prefix 2500 75 warning-only
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# default-originate
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# capability orf prefix both
```

The following example displays sample output from the **show bgp af-group** command using the **configuration** keyword. This example shows from where each configuration item was inherited. The **default-originate** command was configured directly on this address family group (indicated by [ ]). The **remove-private-as** command was inherited from address family group GROUP_2, which in turn inherited from address family group GROUP_3:

```
RP/0/RSP0/CPU0:router# show bgp af-group GROUP_1 configuration

af-group GROUP_1 address-family ipv4 unicast
  capability orf prefix-list both           [a:GROUP_2]
  default-originate                         []
  maximum-prefix 2500 75 warning-only       []
  route-policy POLICY_1 in                  [a:GROUP_2 a:GROUP_3]
  remove-private-AS                         [a:GROUP_2 a:GROUP_3]
  send-community-ebgp                       [a:GROUP_2]
  send-extended-community-ebgp              [a:GROUP_2]
```

The following example displays sample output from the **show bgp af-group** command using the **users** keyword:

```
RP/0/RSP0/CPU0:router# show bgp af-group GROUP_2 users

IPv4 Unicast: a:GROUP_1
```

The following example displays sample output from the **show bgp af-group** command using the **inheritance** keyword. This shows that the specified address family group GROUP_1 directly uses the GROUP_2 address family group, which in turn uses the GROUP_3 address family group:

```
RP/0/RSP0/CPU0:router# show bgp af-group GROUP_1 inheritance
```

```
IPv4 Unicast: a:GROUP_2 a:GROUP_3
```

show bgp session-group

Use the **show bgp session-group** command to display session groups:

- Use the **configuration** keyword to display the effective configuration for the session group, including any settings that have been inherited from session groups used by this session group.
- Use the **inheritance** keyword to display the session groups from which this session group is capable of inheriting configuration.
- Use the **users** keyword to display the session groups, neighbor groups, and neighbors that inherit configuration from this session group.

The output from the **show bgp session-group** command is based on the following session group configuration:

```
RP/0/RSP0/CPU0:router(config)# router bgp 113
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# update-source Loopback 0
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# ebgp-multihop 2
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# dmz-link-bandwidth
```

The following is sample output from the **show bgp session-group** command with the **configuration** keyword in EXEC configuration mode:

```
RP/0/RSP0/CPU0:router# show bgp session-group GROUP_1 configuration

session-group GROUP_1
 ebgp-multihop 2          [s:GROUP_2]
 update-source Loopback0 []
 dmz-link-bandwidth       [s:GROUP_2 s:GROUP_3]
```

The following is sample output from the **show bgp session-group** command with the **inheritance** keyword showing that the GROUP_1 session group inherits session parameters from the GROUP_3 and GROUP_2 session groups:

```
RP/0/RSP0/CPU0:router# show bgp session-group GROUP_1 inheritance

Session: s:GROUP_2 s:GROUP_3
```

The following is sample output from the **show bgp session-group** command with the **users** keyword showing that both the GROUP_1 and GROUP_2 session groups inherit session parameters from the GROUP_3 session group:

```
RP/0/RSP0/CPU0:router# show bgp session-group GROUP_3 users
```

```
Session: s:GROUP_1 s:GROUP_2
```

show bgp neighbor-group

Use the **show bgp neighbor-group** command to display neighbor groups:

- Use the **configuration** keyword to display the effective configuration for the neighbor group, including any settings that have been inherited from neighbor groups used by this neighbor group.
- Use the **inheritance** keyword to display the address family groups, session groups, and neighbor groups from which this neighbor group is capable of inheriting configuration.
- Use the **users** keyword to display the neighbors and neighbor groups that inherit configuration from this neighbor group.

The examples are based on the following group configuration:

```
RP/0/RSP0/CPU0:router(config)# router bgp 140
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# soft-reconfiguration inbound
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# capability orf prefix both
RP/0/RSP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# timers 30 90
RP/0/RSP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1982
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RSP0/CPU0:router(config-nbrgrp)# exit
RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_3
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:routerconfig-bgp-nbrgrp-af)# use af-group GROUP_2
RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
```

The following is sample output from the **show bgp neighbor-group** command with the **configuration** keyword. The configuration setting source is shown to the right of each command. In the output shown previously, the remote autonomous system is configured directly on neighbor group GROUP_1, and the send community setting is inherited from neighbor group GROUP_2, which in turn inherits the setting from address family group GROUP_3:

```
RP/0/RSP0/CPU0:router# show bgp neighbor-group GROUP_1 configuration

   neighbor-group GROUP_1
    remote-as 1982                  []
    timers 30 90                    [n:GROUP_2 s:GROUP_3]
    address-family ipv4 unicast     []
     capability orf prefix-list both [n:GROUP_2 a:GROUP_2]
     remove-private-AS              [n:GROUP_2 a:GROUP_2 a:GROUP_3]
     send-community-ebgp            [n:GROUP_2 a:GROUP_2]
```

```
send-extended-community-ebgp    [n:GROUP_2 a:GROUP_2]
soft-reconfiguration inbound    [n:GROUP_2 a:GROUP_2 a:GROUP_3]
weight 100                      [n:GROUP_2]
```

The following is sample output from the **show bgp neighbor-group** command with the **inheritance** keyword. This output shows that the specified neighbor group GROUP_1 inherits session (address family-independent) configuration parameters from neighbor group GROUP_2. Neighbor group GROUP_2 inherits its session parameters from session group GROUP_3. It also shows that the GROUP_1 neighbor group inherits IPv4 unicast configuration parameters from the GROUP_2 neighbor group, which in turn inherits them from the GROUP_2 address family group, which itself inherits them from the GROUP_3 address family group:

```
RP/0/RSP0/CPU0:router# show bgp neighbor-group GROUP_1 inheritance

    Session:      n:GROUP-2 s:GROUP_3
    IPv4 Unicast: n:GROUP_2 a:GROUP_2 a:GROUP_3
```

The following is sample output from the **show bgp neighbor-group** command with the **users** keyword. This output shows that the GROUP_1 neighbor group inherits session (address family-independent) configuration parameters from the GROUP_2 neighbor group. The GROUP_1 neighbor group also inherits IPv4 unicast configuration parameters from the GROUP_2 neighbor group:

```
RP/0/RSP0/CPU0:router# show bgp neighbor-group GROUP_2 users

Session:      n:GROUP_1
IPv4 Unicast: n:GROUP_1
```

No Default Address Family

BGP does not support the concept of a default address family. An address family must be explicitly configured under the BGP router configuration for the address family to be activated in BGP. Similarly, an address family must be explicitly configured under a neighbor for the BGP session to be activated under that address family. It is not required to have any address family configured under the BGP router configuration level for a neighbor to be configured. However, it is a requirement to have an address family configured at the BGP router configuration level for the address family to be configured under a neighbor.

Neighbor Address Family Combinations

For default VRF, starting from Cisco IOS XR Software Release 6.2.x, both IPv4 Unicast and IPv4 Labeled-unicast address families are supported under the same neighbor.

For non-default VRF, both IPv4 Unicast and IPv4 Labeled-unicast address families are not supported under the same neighbor. However, the configuration is accepted on the Cisco ASR 9000 Series Router with the following error:

```
bgp[1051]: %ROUTING-BGP-4-INCOMPATIBLE_AFI : IPv4 Unicast and IPv4 Labeled-unicast Address
 families together are not supported under the same neighbor.
```

When one BGP session has both IPv4 unicast and IPv4 labeled-unicast AFI/SAF, then the routing behavior is nondeterministic. Therefore, the prefixes may not be correctly advertised. Incorrect prefix advertisement results in reachability issues. In order to avoid such reachability issues, you must explicitly configure a route policy to advertise prefixes either through IPv4 unicast or through IPv4 labeled-unicast address families.

Routing Policy Enforcement

External BGP (eBGP) neighbors must have an inbound and outbound policy configured. If no policy is configured, no routes are accepted from the neighbor, nor are any routes advertised to it. This added security measure ensures that routes cannot accidentally be accepted or advertised in the case of a configuration omission error.

Note This enforcement affects only eBGP neighbors (neighbors in a different autonomous system than this router). For internal BGP (iBGP) neighbors (neighbors in the same autonomous system), all routes are accepted or advertised if there is no policy.

In the following example, for an eBGP neighbor, if all routes should be accepted and advertised with no modifications, a simple pass-all policy is configured:

```
RP/0/RSP0/CPU0:router(config)# route-policy pass-all
RP/0/RSP0/CPU0:router(config-rpl)# pass
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# commit
```

Use the **route-policy (BGP)** command in the neighbor address-family configuration mode to apply the pass-all policy to a neighbor. The following example shows how to allow all IPv4 unicast routes to be received from neighbor 192.168.40.42 and advertise all IPv4 unicast routes back to it:

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 21
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit
```

Use the **show bgp summary** command to display eBGP neighbors that do not have both an inbound and outbound policy for every active address family. In the following example, such eBGP neighbors are indicated in the output with an exclamation (!) mark:

```
RP/0/RSP0/CPU0:router# show bgp all all summary

Address Family: IPv4 Unicast
============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 41
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                41          41          41

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
10.0.101.1        0     1     919     925       41    0    0 15:15:08       10
10.0.101.2        0     2       0       0        0    0    0 00:00:00 Idle
```

```
Address Family: IPv4 Multicast
==============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                  1           1           1

Some configured eBGP neighbors do not have both inbound and
outbound policies configured for IPv4 Multicast address family.
These neighbors will default to sending and/or receiving no
routes and are marked with '!' in the output below. Use the
'show bgp neighbor <nbr_address>' command for details.

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
10.0.101.2        0     2       0       0        0    0    0 00:00:00 Idle!


Address Family: IPv6 Unicast
============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 2
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                  2           2           2

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
2222::2           0     2     920     918        2    0    0 15:15:11        1
2222::4           0     3       0       0        0    0    0 00:00:00 Idle


Address Family: IPv6 Multicast
==============================

BGP router identifier 10.0.0.1, local AS number 1
BGP generic scan interval 60 secs
BGP main routing table version 1
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.

Process         RecvTblVer    bRIB/RIB  SendTblVer
Speaker                  1           1           1

Some configured eBGP neighbors do not have both inbound and
outbound policies configured for IPv6 Multicast address family.
These neighbors will default to sending and/or receiving no
routes and are marked with '!' in the output below. Use the
'show bgp neighbor <nbr_address>' command for details.

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  St/PfxRcd
2222::2           0     2     920     918        0    0    0 15:15:11        0
2222::4           0     3       0       0        0    0    0 00:00:00 Idle!
```

Table Policy

The table policy feature in BGP allows you to configure traffic index values on routes as they are installed in the global routing table. This feature is enabled using the **table-policy** command and supports the BGP policy accounting feature.

BGP policy accounting uses traffic indices that are set on BGP routes to track various counters. See the *Implementing Routing Policy on Cisco ASR 9000 Series Router* module in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers* for details on table policy use. See the *Cisco Express Forwarding Commands on Cisco ASR 9000 Series Router* module in the *IP Addresses and Services Command Reference for Cisco ASR 9000 Series Routers* for details on BGP policy accounting.

Table policy also provides the ability to drop routes from the RIB based on match criteria. This feature can be useful in certain applications and should be used with caution as it can easily create a routing 'black hole' where BGP advertises routes to neighbors that BGP does not install in its global routing table and forwarding table.

Update Groups

The BGP Update Groups feature contains an algorithm that dynamically calculates and optimizes update groups of neighbors that share outbound policies and can share the update messages. The BGP Update Groups feature separates update group replication from peer group configuration, improving convergence time and flexibility of neighbor configuration.

To use this feature, you must understand the following concepts:

Related Topics

BGP Update Generation and Update Groups , on page 31
BGP Update Group , on page 31

BGP Update Generation and Update Groups

The BGP Update Groups feature separates BGP update generation from neighbor configuration. The BGP Update Groups feature introduces an algorithm that dynamically calculates BGP update group membership based on outbound routing policies. This feature does not require any configuration by the network operator. Update group-based message generation occurs automatically and independently.

BGP Update Group

When a change to the configuration occurs, the router automatically recalculates update group memberships and applies the changes.

For the best optimization of BGP update group generation, we recommend that the network operator keeps outbound routing policy the same for neighbors that have similar outbound policies. This feature contains commands for monitoring BGP update groups.

BGP Cost Community

The BGP cost community is a nontransitive extended community attribute that is passed to internal BGP (iBGP) and confederation peers but not to external BGP (eBGP) peers. The cost community feature allows you to customize the local route preference and influence the best-path selection process by assigning cost

values to specific routes. The extended community format defines generic points of insertion (POI) that influence the best-path decision at different points in the best-path algorithm.

The cost community attribute is applied to internal routes by configuring the **set extcommunity cost** command in a route policy. See the *Routing Policy Language Commands on Cisco ASR 9000 Series Router* module of *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference* for information on the **set extcommunity cost** command. The cost community set clause is configured with a cost community ID number (0–255) and cost community number (0–4294967295). The cost community number determines the preference for the path. The path with the lowest cost community number is preferred. Paths that are not specifically configured with the cost community number are assigned a default cost community number of 2147483647 (the midpoint between 0 and 4294967295) and evaluated by the best-path selection process accordingly. When two paths have been configured with the same cost community number, the path selection process prefers the path with the lowest cost community ID. The cost-extended community attribute is propagated to iBGP peers when extended community exchange is enabled.

The following commands include the **route-policy** keyword, which you can use to apply a route policy that is configured with the cost community set clause:

- **aggregate-address**
- **redistribute**
- **network**

How BGP Cost Community Influences the Best Path Selection Process

The cost community attribute influences the BGP best-path selection process at the point of insertion (POI). By default, the POI follows the Interior Gateway Protocol (IGP) metric comparison. When BGP receives multiple paths to the same destination, it uses the best-path selection process to determine which path is the best path. BGP automatically makes the decision and installs the best path in the routing table. The POI allows you to assign a preference to a specific path when multiple equal cost paths are available. If the POI is not valid for local best-path selection, the cost community attribute is silently ignored.

Cost communities are sorted first by POI then by community ID. Multiple paths can be configured with the cost community attribute for the same POI. The path with the lowest cost community ID is considered first. In other words, all cost community paths for a specific POI are considered, starting with the one with the lowest cost community. Paths that do not contain the cost community cost (for the POI and community ID being evaluated) are assigned the default community cost value (2147483647). If the cost community values are equal, then cost community comparison proceeds to the next lowest community ID for this POI.

To select the path with the lower cost community, simultaneously walk through the cost communities of both paths. This is done by maintaining two pointers to the cost community chain, one for each path, and advancing both pointers to the next applicable cost community at each step of the walk for the given POI, in order of community ID, and stop when a best path is chosen or the comparison is a tie. At each step of the walk, the following checks are done:

```
If neither pointer refers to a cost community,
     Declare a tie;

  Elseif a cost community is found for one path but not for the other,
     Choose the path with cost community as best path;
  Elseif the Community ID from one path is less than the other,
     Choose the path with the lesser Community ID as best path;
  Elseif the Cost from one path is less than the other,
     Choose the path with the lesser Cost as best path;
```

```
Else Continue.
```

Note Paths that are not configured with the cost community attribute are considered by the best-path selection process to have the default cost value (half of the maximum value [4294967295] or 2147483647).

Applying the cost community attribute at the POI allows you to assign a value to a path originated or learned by a peer in any part of the local autonomous system or confederation. The cost community can be used as a "tie breaker" during the best-path selection process. Multiple instances of the cost community can be configured for separate equal cost paths within the same autonomous system or confederation. For example, a lower cost community value can be applied to a specific exit path in a network with multiple equal cost exit points, and the specific exit path is preferred by the BGP best-path selection process. See the scenario described inInfluencing Route Preference in a Multiexit IGP Network, on page 34.

Note The cost community comparison in BGP is enabled by default. Use the **bgp bestpath cost-community ignore** command to disable the comparison.

SeeBGP Best Path Algorithm, on page 39 for information on the BGP best-path selection process.

Cost Community Support for Aggregate Routes and Multipaths

The BGP cost community feature supports aggregate routes and multipaths. The cost community attribute can be applied to either type of route. The cost community attribute is passed to the aggregate or multipath route from component routes that carry the cost community attribute. Only unique IDs are passed, and only the highest cost of any individual component route is applied to the aggregate for each ID. If multiple component routes contain the same ID, the highest configured cost is applied to the route. For example, the following two component routes are configured with the cost community attribute using an inbound route policy:

- 10.0.0.1
  - POI=IGP
  - cost community ID=1
  - cost number=100
- 192.168.0.1
  - POI=IGP
  - cost community ID=1
  - cost number=200

If these component routes are aggregated or configured as a multipath, the cost value 200 is advertised, because it has the highest cost.

If one or more component routes do not carry the cost community attribute or the component routes are configured with different IDs, then the default value (2147483647) is advertised for the aggregate or multipath route. For example, the following three component routes are configured with the cost community attribute using an inbound route policy. However, the component routes are configured with two different IDs.

- 10.0.0.1
  - POI=IGP
  - cost community ID=1
  - cost number=100
- 172.16.0.1
  - POI=IGP
  - cost community ID=2
  - cost number=100
- 192.168.0.1
  - POI=IGP
  - cost community ID=1
  - cost number=200

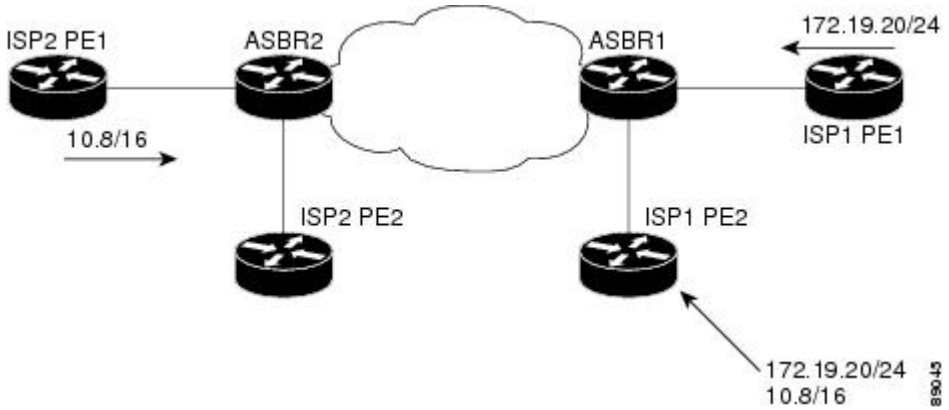
The single advertised path includes the aggregate cost communities as follows:

{POI=IGP, ID=1, Cost=2147483647} {POI-IGP, ID=2, Cost=2147483647}

Influencing Route Preference in a Multiexit IGP Network

This figure shows an IGP network with two autonomous system boundary routers (ASBRs) on the edge. Each ASBR has an equal cost path to network 10.8/16.

Figure 1: Multiexit Point IGP Network



Both paths are considered to be equal by BGP. If multipath loadsharing is configured, both paths to the routing table are installed and are used to balance the load of traffic. If multipath load balancing is not configured, the BGP selects the path that was learned first as the best path and installs this path to the routing table. This behavior may not be desirable under some conditions. For example, the path is learned from ISP1 PE2 first, but the link between ISP1 PE2 and ASBR1 is a low-speed link.

The configuration of the cost community attribute can be used to influence the BGP best-path selection process by applying a lower-cost community value to the path learned by ASBR2. For example, the following configuration is applied to ASBR2:

```
RP/0/RSP0/CPU0:router(config)# route-policy ISP2_PE1
RP/0/RSP0/CPU0:router(config-rpl)# set extcommunity cost (1:1)
```

The preceding route policy applies a cost community number of 1 to the 10.8.0.0 route. By default, the path learned from ASBR1 is assigned a cost community number of 2147483647. Because the path learned from ASBR2 has a lower-cost community number, the path is preferred.

BGP Cost Community Support for EIGRP MPLS VPN PE-CE with Back-door Links

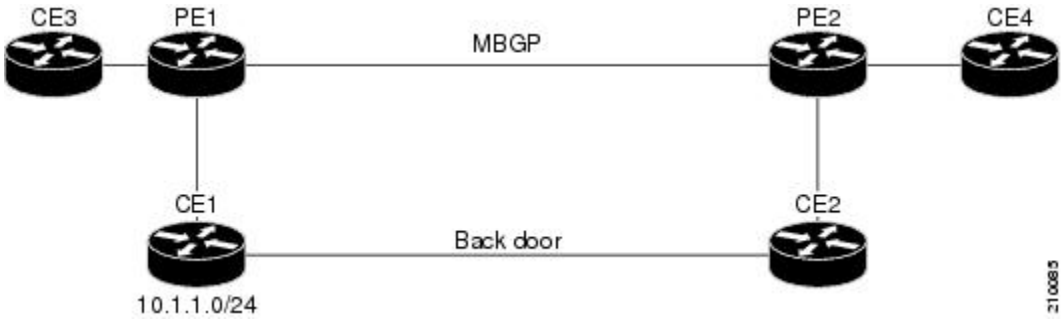
Back-door links in an EIGRP MPLS VPN topology is preferred by BGP if the back-door link is learned first. (A back-door link, or route, is a connection that is configured outside of the VPN between a remote and main site; for example, a WAN leased line that connects a remote site to the corporate network.)

The "prebest path" point of insertion (POI) in the BGP cost community feature supports mixed EIGRP VPN network topologies that contain VPN and back-door links. This POI is applied automatically to EIGRP routes that are redistributed into BGP. The "prebest path" POI carries the EIGRP route type and metric. This POI influences the best-path calculation process by influencing BGP to consider the POI before any other comparison step. No configuration is required. This feature is enabled automatically for EIGRP VPN sites when Cisco IOS XR software is installed on a PE, CE, or back-door router.

For information about configuring EIGRP MPLS VPNs, see the *MPLS Configuration Guide for Cisco ASR 9000 Series Routers*.

Figure 2: Network Showing How Cost Community Can be Used to Support Backdoor Links

This figure shows how cost community can be used to support backdoor links in a network.



The following sequence of events happens in PE1:

1. PE1 learns IPv4 prefix 10.1.1.0/24 from CE1 through EIGRP running a virtual routing and forwarding (VRF) instance. EIGRP selects and installs the best path in the RIB. It also encodes the cost-extended community and adds the information to the RIB.
2. The route is redistributed into BGP (assuming that IGP-to-BGP redistribution is configured). BGP also receives the cost-extended community from the route through the redistribution process.
3. After BGP has determined the best path for the newly redistributed prefix, the path is advertised to PE peers (PE2).

4. PE2 receives the BGP VPNv4 prefix route_distinguisher:10.1.1.0/24 along with the cost community. It is likely that CE2 advertises the same prefix (because of the back-door link between CE1 and CE2) to PE2 through EIGRP. PE2 BGP would have already learned the CE route through the redistribution process along with the cost community value
5. PE2 has two paths within BGP: one with cost community cost1 through multipath BGP (PE1) and another with cost community cost2 through the EIGRP neighbor (CE2).
6. PE2 runs the enhanced BGP best-path calculation.
7. PE2 installs the best path in the RIB passing the appropriate cost community value.
8. PE2 RIB has two paths for 10.1.1.0/24: one with cost community cost2 added by EIGRP and another with the cost community cost1 added by BGP. Because both the route paths have cost community, RIB compares the costs first. The BGP path has the lower cost community, so it is selected and downloaded to the RIB.
9. PE2 RIB redistributes the BGP path into EIGRP with VRF. EIGRP runs a diffusing update algorithm (DUAL) because there are two paths, and selects the BGP-redistributed path.
10. PE2 EIGRP advertises the path to CE2 making the path the next hop for the prefix to send the traffic over the MPLS network.

Adding Routes to the Routing Information Base

If a nonsourced path becomes the best path after the best-path calculation, BGP adds the route to the Routing Information Base (RIB) and passes the cost communities along with the other IGP extended communities.

When a route with paths is added to the RIB by a protocol, RIB checks the current best paths for the route and the added paths for cost extended communities. If cost-extended communities are found, the RIB compares the set of cost communities. If the comparison does not result in a tie, the appropriate best path is chosen. If the comparison results in a tie, the RIB proceeds with the remaining steps of the best-path algorithm. If a cost community is not present in either the current best paths or added paths, then the RIB continues with the remaining steps of the best-path algorithm. See BGP Best Path Algorithm, on page 39 for information on the BGP best-path algorithm.

BGP DMZ Aggregate Bandwidth

BGP supports aggregating *dmz-link bandwidth* values of external BGP (eBGP) multipaths when advertising the route to interior BGP (iBGP) peer.

There is no explicit command to aggregate bandwidth. The bandwidth is aggregated if following conditions are met:

- The network has multipaths and all the multipaths have link-bandwidth values.
- The next-hop attribute set to next-hop-self. The next-hop attribute for all routes advertised to the specified neighbor to the address of the local router.
- There is no out-bound policy configured that might change the dmz-link bandwidth value.

Note

- If the *dmz-link bandwidth* value is not known for any one of the multipaths (eBGP or iBGP), the *dmz-link* value for all multipaths including the best path is not downloaded to routing information base (RIB).
- The *dmz-link bandwidth* value of iBGP multipath is not considered during aggregation.
- The route that is advertised with aggregate value can be best path or add-path.
- Add-path does not qualify for DMZ link bandwidth aggregation as next hop is preserved. Configuring next-hop-self for add-path is not supported.
- For VPNv4 and VPNv6 afi, if *dmz link-bandwidth* value is configured using outbound route-policy, specify the route table or use the **additive** keyword. Else, this will lead to routes not imported on the receiving end of the peer.

```
extcommunity-set bandwidth dmz_ext
   1:8000
 end-set
 !
 route-policy dmz_rp_vpn
   set extcommunity bandwidth dmz_ext additive     <<< 'additive' keyword.
   pass
 end-policy
```

Example

Consider two routers Router 1 and Router 2 that are connected to internal routers in a network. Router 1 advertises a bandwidth of 50 and 20 from two different ISPs. Router 2 advertises a bandwidth of 60 and 30 from two different ISPs. With the best-path algorithm, Router 1 advertises a bandwidth of 50 and Router 2 advertises a bandwidth of 60 to the internal routers. This reduces traffic flow. But by aggregating the bandwidth, Router 1 advertises a bandwidth of 70 (50 + 20) and Router 2 advertises a bandwidth of 90 (60 + 30). This increases the traffic flow.

Configuring BGP DMZ Aggregate Bandwidth: Example

This is a sample configuration for Border Gateway Protocol Demilitarized Zone (BGP DMZ) link bandwidth. Consider the topology, R1---(iBGP)---R2---(iBGP)---R3:

1. On R1:

   ```
   bgp: prefix p/n has:
   path 1(bestpath)       with LB value 100
   path 2(ebgp multipath) with LB value 30
   path 3(ebgp multipath) with LB value 50
   ```

   When best path is advertised to R2, send aggregated dmz-link bandwidth value of 180; aggregated value of paths 1, 2 and 3.
2. On R2:

   ```
   bgp: prefix p/n has:
   path 1(bestpath)       with LB value 60
   path 2(ebgp multipath) with LB value 200
   path 3(ebgp multipath) with LB value 50
   ```

   When best path is advertised to R3, send aggregated dmz-link bandwidth value of 310; aggregated value of paths 1, 2 and 3.

3. On R3:

```
bgp: prefix p/n has:
path 1(bestpath)       with LB 180 {learned from R1}
path 2(ibgp multipath) with LB 310 {learned from R2}
```

Configuring Policy-based Link Bandwidth: Example

This is a sample configuration for policy-based DMZ link bandwidth. The link-bandwidth ext-community can be set on a *per-path* basis either at the neighbor-in or neighbor-out policy attach-points. The *dmz-link-bandwidth* knob is configured under eBGP neighbor configuration mode. All paths received from that particular neighbor will be marked with the link-bandwidth extended community when sent to iBGP peers.

1. Configure inbound or outbound route-policy.

```
extcommunity-set bandwidth dmz_ext
  1:1290400000
end-set
!
route-policy dmz_rp
  set extcommunity bandwidth dmz_ext
  pass
end-policy
!

 neighbor 10.0.101.1
  remote-as 1001
  address-family ipv4 unicast
   route-policy dmz_rp in          <<< Inbound route-policy.
   route-policy pass out
  !
```

2. Configure *dmz-link-bandwidth* under BGP neighbor.

```
neighbor 10.0.101.2
  remote-as 1001
  dmz-link-bandwidth              <<< Under neighbor.
  address-family ipv4 unicast
   route-policy pass in
   route-policy pass out
  !
```

For more information on policy-based extended community set, see the *Implementing Routing Policy* chapter in *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* .

64-ECMP Support for BGP

IOS XR supports configuration of up to 64 equal cost multipath (ECMP) next hops for BGP. 64-ECMP is required in networks, where overloaded routers can load balance the traffic over as many as 64 LSPs.

More than 32 ECMP and UCMP paths are not supported for these features:

- LI
- GRE
- BVI
- NetFlow

- Satellite
- MCAST
- SPAN
- PWHE
- ABF
- P2MP
- MVPN
- VPLS
- L2TPv3
- LISP
- VIDMON
- PBR

BGP Best Path Algorithm

BGP routers typically receive multiple paths to the same destination. The BGP best-path algorithm determines the best path to install in the IP routing table and to use for forwarding traffic. This section describes the Cisco IOS XR software implementation of BGP best-path algorithm, as specified in Section 9.1 of the Internet Engineering Task Force (IETF) Network Working Group draft-ietf-idr-bgp4-24.txt document.

The BGP best-path algorithm implementation is in three parts:

- Part 1—Compares two paths to determine which is better.
- Part 2—Iterates over all paths and determines which order to compare the paths to select the overall best path.
- Part 3—Determines whether the old and new best paths differ enough so that the new best path should be used.

Note The order of comparison determined by Part 2 is important because the comparison operation is not transitive; that is, if three paths, A, B, and C exist, such that when A and B are compared, A is better, and when B and C are compared, B is better, it is not necessarily the case that when A and C are compared, A is better. This nontransitivity arises because the multi exit discriminator (MED) is compared only among paths from the same neighboring autonomous system (AS) and not among all paths.

Comparing Pairs of Paths

Perform the following steps to compare two paths and determine the better path:

1. If either path is invalid (for example, a path has the maximum possible MED value or it has an unreachable next hop), then the other path is chosen (provided that the path is valid).

2. If the paths have unequal pre-bestpath cost communities, the path with the lower pre-bestpath cost community is selected as the best path.
3. If the paths have unequal weights, the path with the highest weight is chosen.

Note The weight is entirely local to the router, and can be set with the **weight** command or using a routing policy.

4. If the paths have unequal local preferences, the path with the higher local preference is chosen.

Note If a local preference attribute was received with the path or was set by a routing policy, then that value is used in this comparison. Otherwise, the default local preference value of 100 is used. The default value can be changed using the **bgp default local-preference** command.

5. If one of the paths is a redistributed path, which results from a **redistribute** or **network** command, then it is chosen. Otherwise, if one of the paths is a locally generated aggregate, which results from an **aggregate-address** command, it is chosen.

Note Step 1 through Step 4 implement the “Path Selection with BGP”of RFC 1268.

6. If the paths have unequal AS path lengths, the path with the shorter AS path is chosen. This step is skipped if **bgp bestpath as-path ignore** command is configured.

Note When calculating the length of the AS path, confederation segments are ignored, and AS sets count as 1.

Note eiBGP specifies internal and external BGP multipath peers. eiBGP allows simultaneous use of internal and external paths.

7. If the paths have different origins, the path with the lower origin is selected. Interior Gateway Protocol (IGP) is considered lower than EGP, which is considered lower than INCOMPLETE.
8. If appropriate, the MED of the paths is compared. If they are unequal, the path with the lower MED is chosen.

   A number of configuration options exist that affect whether or not this step is performed. In general, the MED is compared if both paths were received from neighbors in the same AS; otherwise the MED comparison is skipped. However, this behavior is modified by certain configuration options, and there are also some corner cases to consider.

   If the **bgp bestpath med always** command is configured, then the MED comparison is always performed, regardless of neighbor AS in the paths. Otherwise, MED comparison depends on the AS paths of the two paths being compared, as follows:

- If a path has no AS path or the AS path starts with an AS_SET, then the path is considered to be internal, and the MED is compared with other internal paths.
- If the AS path starts with an AS_SEQUENCE, then the neighbor AS is the first AS number in the sequence, and the MED is compared with other paths that have the same neighbor AS.
- If the AS path contains only confederation segments or starts with confederation segments followed by an AS_SET, then the MED is not compared with any other path unless the **bgp bestpath med confed** command is configured. In that case, the path is considered internal and the MED is compared with other internal paths.
- If the AS path starts with confederation segments followed by an AS_SEQUENCE, then the neighbor AS is the first AS number in the AS_SEQUENCE, and the MED is compared with other paths that have the same neighbor AS.

Note If no MED attribute was received with the path, then the MED is considered to be 0 unless the **bgp bestpath med missing-as-worst** command is configured. In that case, if no MED attribute was received, the MED is considered to be the highest possible value.

9. If one path is received from an external peer and the other is received from an internal (or confederation) peer, the path from the external peer is chosen.
10. If the paths have different IGP metrics to their next hops, the path with the lower IGP metric is chosen.
11. If the paths have unequal IP cost communities, the path with the lower IP cost community is selected as the best path.
12. If all path parameters in Step 1 through Step 10 are the same, then the router IDs are compared. If the path was received with an originator attribute, then that is used as the router ID to compare; otherwise, the router ID of the neighbor from which the path was received is used. If the paths have different router IDs, the path with the lower router ID is chosen.

Note Where the originator is used as the router ID, it is possible to have two paths with the same router ID. It is also possible to have two BGP sessions with the same peer router, and therefore receive two paths with the same router ID.

13. If the paths have different cluster lengths, the path with the shorter cluster length is selected. If a path was not received with a cluster list attribute, it is considered to have a cluster length of 0.
14. Finally, the path received from the neighbor with the lower IP address is chosen. Locally generated paths (for example, redistributed paths) are considered to have a neighbor IP address of 0.

Order of Comparisons

The second part of the BGP best-path algorithm implementation determines the order in which the paths should be compared. The order of comparison is determined as follows:

1. The paths are partitioned into groups such that within each group the MED can be compared among all paths. The same rules as in #unique_76 are used to determine whether MED can be compared between

any two paths. Normally, this comparison results in one group for each neighbor AS. If the **bgp bestpath med always** command is configured, then there is just one group containing all the paths.

2. The best path in each group is determined. Determining the best path is achieved by iterating through all paths in the group and keeping track of the best one seen so far. Each path is compared with the best-so-far, and if it is better, it becomes the new best-so-far and is compared with the next path in the group.
3. A set of paths is formed containing the best path selected from each group in Step 2. The overall best path is selected from this set of paths, by iterating through them as in Step 2.

Best Path Change Suppression

The third part of the implementation is to determine whether the best-path change can be suppressed or not—whether the new best path should be used, or continue using the existing best path. The existing best path can continue to be used if the new one is identical to the point at which the best-path selection algorithm becomes arbitrary (if the router-id is the same). Continuing to use the existing best path can avoid churn in the network.

Note This suppression behavior does not comply with the IETF Networking Working Group draft-ietf-idr-bgp4-24.txt document, but is specified in the IETF Networking Working Group draft-ietf-idr-avoid-transition-00.txt document.

The suppression behavior can be turned off by configuring the **bgp bestpath compare-routerid** command. If this command is configured, the new best path is always preferred to the existing one.

Otherwise, the following steps are used to determine whether the best-path change can be suppressed:

1. If the existing best path is no longer valid, the change cannot be suppressed.
2. If either the existing or new best paths were received from internal (or confederation) peers or were locally generated (for example, by redistribution), then the change cannot be suppressed. That is, suppression is possible only if both paths were received from external peers.
3. If the paths were received from the same peer (the paths would have the same router-id), the change cannot be suppressed. The router ID is calculated using rules in #unique_76.
4. If the paths have different weights, local preferences, origins, or IGP metrics to their next hops, then the change cannot be suppressed. Note that all these values are calculated using the rules in #unique_76.
5. If the paths have different-length AS paths and the **bgp bestpath as-path ignore** command is not configured, then the change cannot be suppressed. Again, the AS path length is calculated using the rules in #unique_76.
6. If the MED of the paths can be compared and the MEDs are different, then the change cannot be suppressed. The decision as to whether the MEDs can be compared is exactly the same as the rules in #unique_76, as is the calculation of the MED value.
7. If all path parameters in Step 1 through Step 6 do not apply, the change can be suppressed.

Administrative Distance

An administrative distance is a rating of the trustworthiness of a routing information source. In general, the higher the value, the lower the trust rating. For information on specifying the administrative distance for BGP, see the BGP Commands module of the *Routing Command Reference for Cisco ASR 9000 Series Routers*

Normally, a route can be learned through more than one protocol. Administrative distance is used to discriminate between routes learned from more than one protocol. The route with the lowest administrative distance is installed in the IP routing table. By default, BGP uses the administrative distances shown in Table 2: BGP Default Administrative Distances, on page 43.

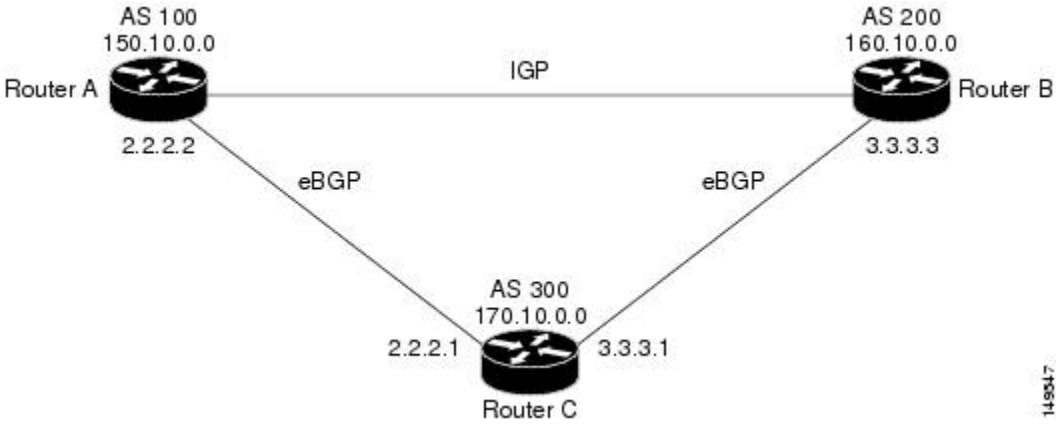
Table 2: BGP Default Administrative Distances

Distance	Default Value	Function
External	20	Applied to routes learned from eBGP.
Internal	200	Applied to routes learned from iBGP.
Local	200	Applied to routes originated by the router.

Note Distance does not influence the BGP path selection algorithm, but it does influence whether BGP-learned routes are installed in the IP routing table.

In most cases, when a route is learned through eBGP, it is installed in the IP routing table because of its distance (20). Sometimes, however, two ASs have an IGP-learned back-door route and an eBGP-learned route. Their policy might be to use the IGP-learned path as the preferred path and to use the eBGP-learned path when the IGP path is down. See Figure 3: Back Door Example , on page 43.

Figure 3: Back Door Example



In Figure 3: Back Door Example , on page 43, Routers A and C and Routers B and C are running eBGP. Routers A and B are running an IGP (such as Routing Information Protocol [RIP], Interior Gateway Routing Protocol [IGRP], Enhanced IGRP, or Open Shortest Path First [OSPF]). The default distances for RIP, IGRP, Enhanced IGRP, and OSPF are 120, 100, 90, and 110, respectively. All these distances are higher than the default distance of eBGP, which is 20. Usually, the route with the lowest distance is preferred.

Router A receives updates about 160.10.0.0 from two routing protocols: eBGP and IGP. Because the default distance for eBGP is lower than the default distance of the IGP, Router A chooses the eBGP-learned route from Router C. If you want Router A to learn about 160.10.0.0 from Router B (IGP), establish a BGP back door. See .

In the following example, a network back-door is configured:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# network 160.10.0.0/16 backdoor
```

Router A treats the eBGP-learned route as local and installs it in the IP routing table with a distance of 200. The network is also learned through Enhanced IGRP (with a distance of 90), so the Enhanced IGRP route is successfully installed in the IP routing table and is used to forward traffic. If the Enhanced IGRP-learned route goes down, the eBGP-learned route is installed in the IP routing table and is used to forward traffic.

Although BGP treats network 160.10.0.0 as a local entry, it does not advertise network 160.10.0.0 as it normally would advertise a local entry.

Multiprotocol BGP

Multiprotocol BGP is an enhanced BGP that carries routing information for multiple network layer protocols and IP multicast routes. BGP carries two sets of routes, one set for unicast routing and one set for multicast routing. The routes associated with multicast routing are used by the Protocol Independent Multicast (PIM) feature to build data distribution trees.

Multiprotocol BGP is useful when you want a link dedicated to multicast traffic, perhaps to limit which resources are used for which traffic. Multiprotocol BGP allows you to have a unicast routing topology different from a multicast routing topology providing more control over your network and resources.

In BGP, the only way to perform interdomain multicast routing was to use the BGP infrastructure that was in place for unicast routing. Perhaps you want all multicast traffic exchanged at one network access point (NAP). If those routers were not multicast capable, or there were differing policies for which you wanted multicast traffic to flow, multicast routing could not be supported without multiprotocol BGP.

Note It is possible to configure BGP peers that exchange both unicast and multicast network layer reachability information (NLRI), but you cannot connect multiprotocol BGP clouds with a BGP cloud. That is, you cannot redistribute multiprotocol BGP routes into BGP.

Figure 4: Noncongruent Unicast and Multicast Routes, on page 45 illustrates simple unicast and multicast topologies that are incongruent, and therefore are not possible without multiprotocol BGP.

Autonomous systems 100, 200, and 300 are each connected to two NAPs that are FDDI rings. One is used for unicast peering (and therefore the exchange of unicast traffic). The Multicast Friendly Interconnect (MFI) ring is used for multicast peering (and therefore the exchange of multicast traffic). Each router is unicast and multicast capable.

Figure 4: Noncongruent Unicast and Multicast Routes

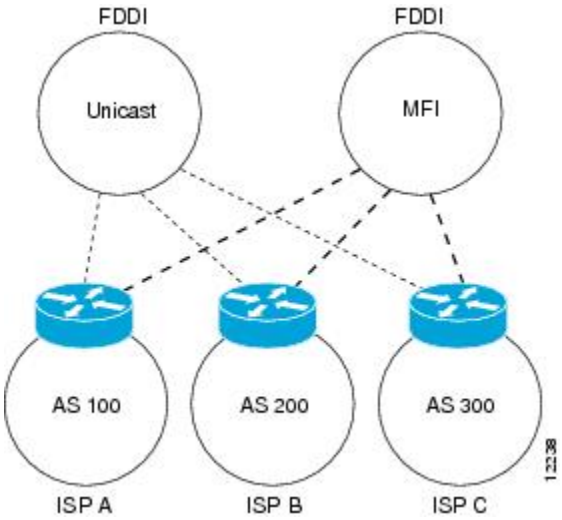


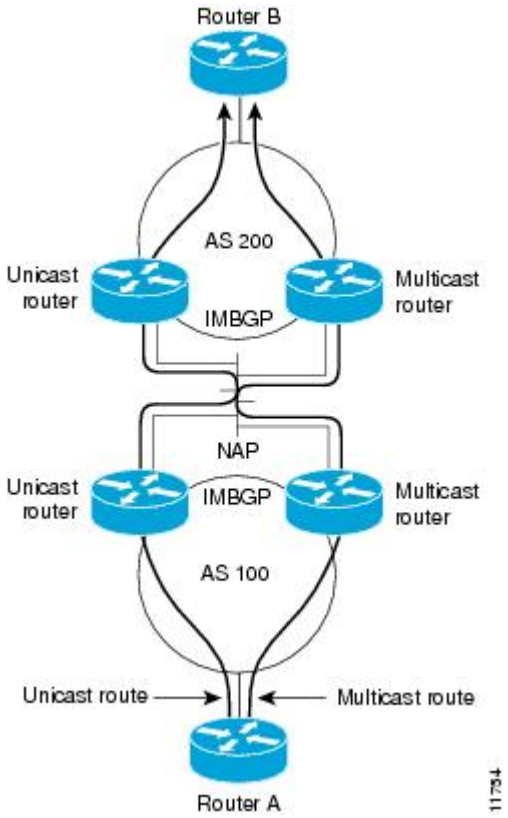
Figure 5: Multicast BGP Environment, on page 46 is a topology of unicast-only routers and multicast-only routers. The two routers on the left are unicast-only routers (that is, they do not support or are not configured to perform multicast routing). The two routers on the right are multicast-only routers. Routers A and B support both unicast and multicast routing. The unicast-only and multicast-only routers are connected to a single NAP.

In Figure 5: Multicast BGP Environment, on page 46, only unicast traffic can travel from Router A to the unicast routers to Router B and back. Multicast traffic could not flow on that path, so another routing table is required. Multicast traffic uses the path from Router A to the multicast routers to Router B and back.

Figure 5: Multicast BGP Environment, on page 46 illustrates a multiprotocol BGP environment with a separate unicast route and multicast route from Router A to Router B. Multiprotocol BGP allows these routes to be incongruent. Both of the autonomous systems must be configured for internal multiprotocol BGP (IMBGP) in the figure.

A multicast routing protocol, such as PIM, uses the multicast BGP database to perform Reverse Path Forwarding (RPF) lookups for multicast-capable sources. Thus, packets can be sent and accepted on the multicast topology but not on the unicast topology.

Figure 5: Multicast BGP Environment



Route Dampening

Route dampening is a BGP feature that minimizes the propagation of flapping routes across an internetwork. A route is considered to be flapping when it is repeatedly available, then unavailable, then available, then unavailable, and so on.

For example, consider a network with three BGP autonomous systems: autonomous system 1, autonomous system 2, and autonomous system 3. Suppose the route to network A in autonomous system 1 flaps (it becomes unavailable). Under circumstances without route dampening, the eBGP neighbor of autonomous system 1 to autonomous system 2 sends a withdraw message to autonomous system 2. The border router in autonomous system 2, in turn, propagates the withdrawal message to autonomous system 3. When the route to network A reappears, autonomous system 1 sends an advertisement message to autonomous system 2, which sends it to autonomous system 3. If the route to network A repeatedly becomes unavailable, then available, many withdrawal and advertisement messages are sent. Route flapping is a problem in an internetwork connected to the Internet, because a route flap in the Internet backbone usually involves many routes.

Minimizing Flapping

The route dampening feature minimizes the flapping problem as follows. Suppose again that the route to network A flaps. The router in autonomous system 2 (in which route dampening is enabled) assigns network A a penalty of 1000 and moves it to history state. The router in autonomous system 2 continues to advertise the status of the route to neighbors. The penalties are cumulative. When the route flaps so often that the penalty

exceeds a configurable suppression limit, the router stops advertising the route to network A, regardless of how many times it flaps. Thus, the route is dampened.

The penalty placed on network A is decayed until the reuse limit is reached, upon which the route is once again advertised. At half of the reuse limit, the dampening information for the route to network A is removed.

Note No penalty is applied to a BGP peer reset when route dampening is enabled, even though the reset withdraws the route.

BGP Routing Domain Confederation

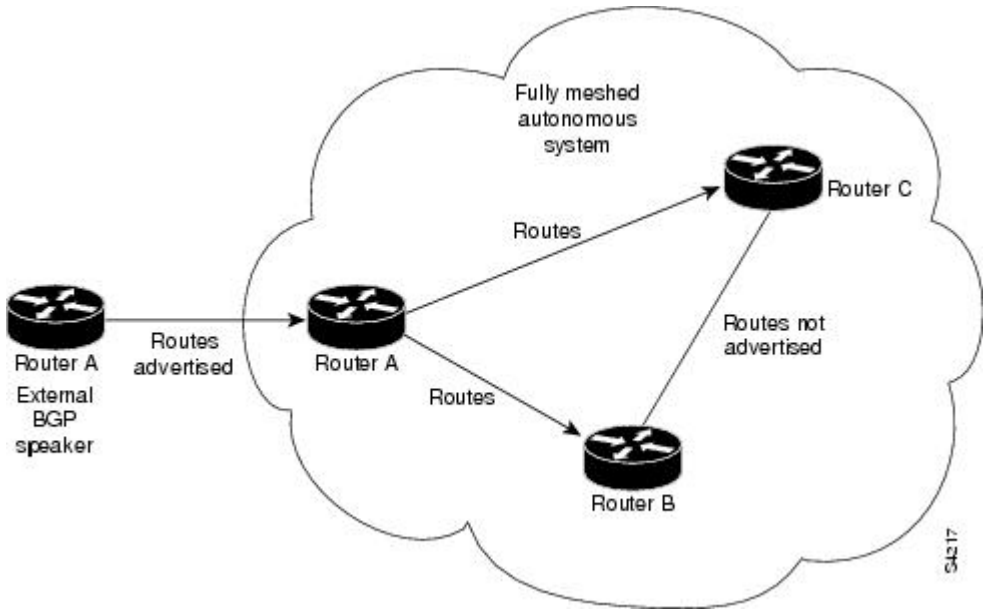
One way to reduce the iBGP mesh is to divide an autonomous system into multiple subautonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Although the peers in different autonomous systems have eBGP sessions, they exchange routing information as if they were iBGP peers. Specifically, the next hop, MED, and local preference information is preserved. This feature allows you to retain a single IGP for all of the autonomous systems.

BGP Route Reflectors

BGP requires that all iBGP speakers be fully meshed. However, this requirement does not scale well when there are many iBGP speakers. Instead of configuring a confederation, you can reduce the iBGP mesh by using a route reflector configuration.

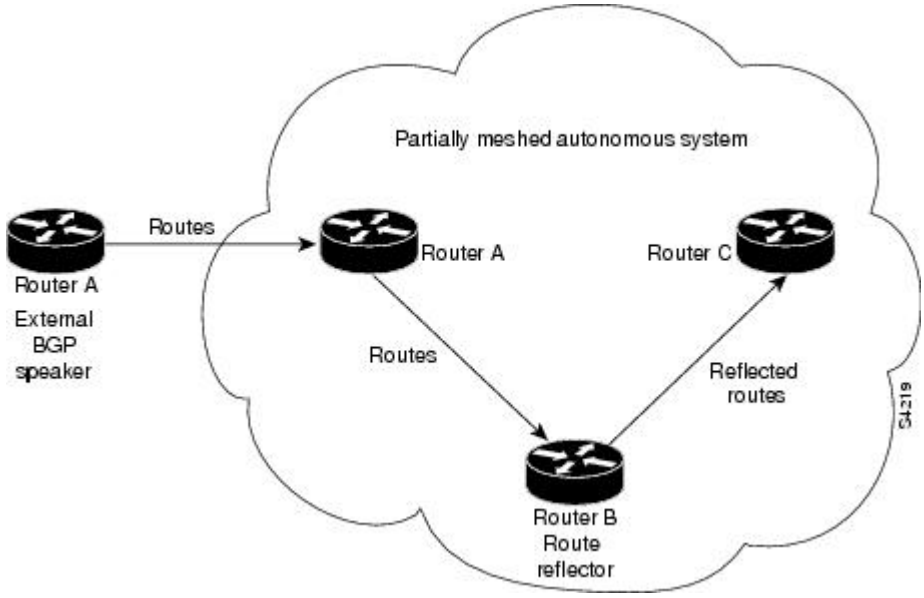
Figure 6: Three Fully Meshed iBGP Speakers, on page 48 illustrates a simple iBGP configuration with three iBGP speakers (routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise the iBGP learned route to other iBGP speakers because the routers do not pass on routes learned from internal neighbors to other internal neighbors, thus preventing a routing information loop.

Figure 6: Three Fully Meshed iBGP Speakers



With route reflectors, all iBGP speakers need not be fully meshed because there is a method to pass learned routes to neighbors. In this model, an iBGP peer is configured to be a route reflector responsible for passing iBGP learned routes to a set of iBGP neighbors. In Figure 7: Simple BGP Model with a Route Reflector, on page 48 , Router B is configured as a route reflector. When the route reflector receives routes advertised from Router A, it advertises them to Router C, and vice versa. This scheme eliminates the need for the iBGP session between routers A and C.

Figure 7: Simple BGP Model with a Route Reflector



The internal peers of the route reflector are divided into two groups: client peers and all other routers in the autonomous system (nonclient peers). A route reflector reflects routes between these two groups. The route reflector and its client peers form a *cluster*. The nonclient peers must be fully meshed with each other, but the

client peers need not be fully meshed. The clients in the cluster do not communicate with iBGP speakers outside their cluster.

Figure 8: More Complex BGP Route Reflector Model

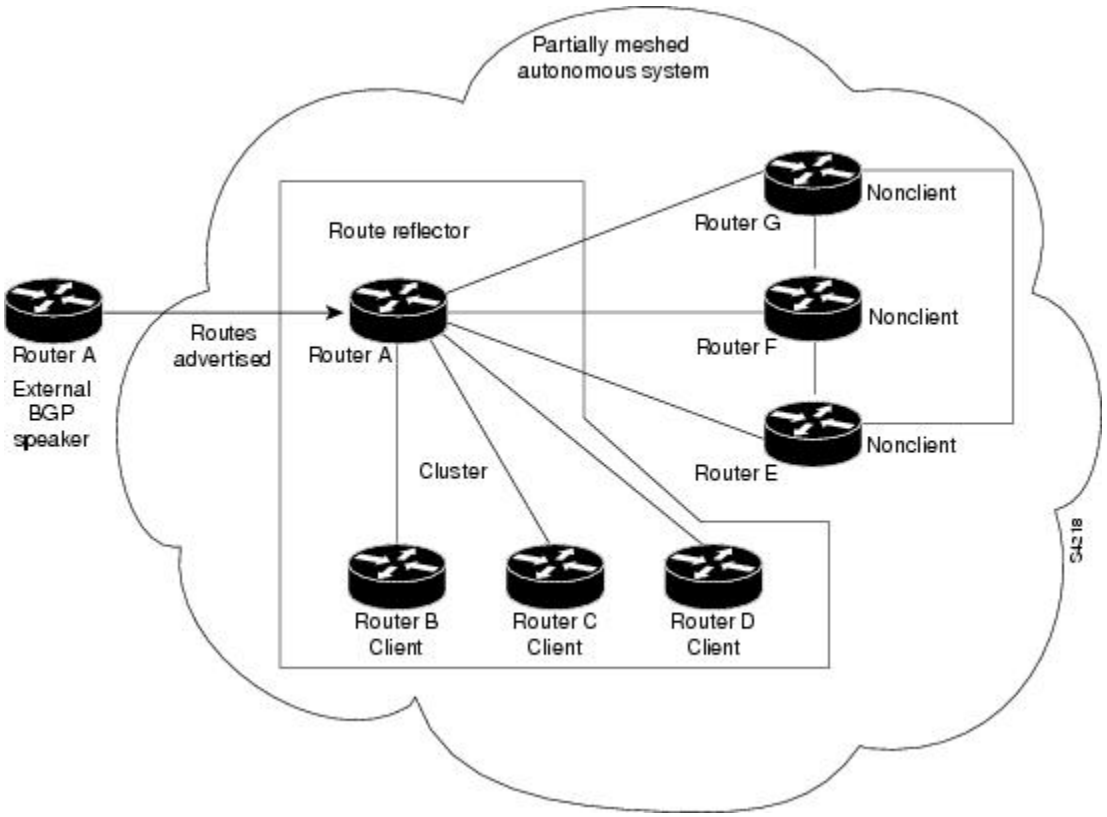


Figure 8: More Complex BGP Route Reflector Model, on page 49 illustrates a more complex route reflector scheme. Router A is the route reflector in a cluster with routers B, C, and D. Routers E, F, and G are fully meshed, nonclient routers.

When the route reflector receives an advertised route, depending on the neighbor, it takes the following actions:

- A route from an external BGP speaker is advertised to all clients and nonclient peers.
- A route from a nonclient peer is advertised to all clients.
- A route from a client is advertised to all clients and nonclient peers. Hence, the clients need not be fully meshed.

Along with route reflector-aware BGP speakers, it is possible to have BGP speakers that do not understand the concept of route reflectors. They can be members of either client or nonclient groups, allowing an easy and gradual migration from the old BGP model to the route reflector model. Initially, you could create a single cluster with a route reflector and a few clients. All other iBGP speakers could be nonclient peers to the route reflector and then more clusters could be created gradually.

An autonomous system can have multiple route reflectors. A route reflector treats other route reflectors just like other iBGP speakers. A route reflector can be configured to have other route reflectors in a client group or nonclient group. In a simple configuration, the backbone could be divided into many clusters. Each route

reflector would be configured with other route reflectors as nonclient peers (thus, all route reflectors are fully meshed). The clients are configured to maintain iBGP sessions with only the route reflector in their cluster.

Usually, a cluster of clients has a single route reflector. In that case, the cluster is identified by the router ID of the route reflector. To increase redundancy and avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. All route reflectors serving a cluster should be fully meshed and all of them should have identical sets of client and nonclient peers.

By default, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients. However, if the clients are fully meshed, the route reflector need not reflect routes to clients.

As the iBGP learned routes are reflected, routing information may loop. The route reflector model has the following mechanisms to avoid routing loops:

- Originator ID is an optional, nontransitive BGP attribute. It is a 4-byte attributed created by a route reflector. The attribute carries the router ID of the originator of the route in the local autonomous system. Therefore, if a misconfiguration causes routing information to come back to the originator, the information is ignored.
- Cluster-list is an optional, nontransitive BGP attribute. It is a sequence of cluster IDs that the route has passed. When a route reflector reflects a route from its clients to nonclient peers, and vice versa, it appends the local cluster ID to the cluster-list. If the cluster-list is empty, a new cluster-list is created. Using this attribute, a route reflector can identify if routing information is looped back to the same cluster due to misconfiguration. If the local cluster ID is found in the cluster-list, the advertisement is ignored.

BGP Optimal Route Reflector

BGP-ORR (optimal route reflector) enables virtual route reflector (vRR) to calculate the best path from a route reflector (RR) client's point of view.

BGP ORR calculates the best path by:

1. Running SPF multiple times in the context of its RR clients or RR clusters (set of RR clients)
2. Saving the result of different SPF runs in separate databases
3. Using these databases to manipulate BGP best path decision and thereby allowing BGP to use and announce best path that is optimal from the client’s point of view

Note Enabling the ORR feature increases the memory footprint of BGP and RIB. With increased number of vRR configured in the network, ORR adversely impacts convergence for BGP.

In an autonomous system, a BGP route reflector acts as a focal point and advertises routes to its peers (RR clients) along with the RR's computed best path. Since the best path advertised by the RR is computed from the RR's point of view, the RR's placement becomes an important deployment consideration.

With network function virtualization (NFV) becoming a dominant technology, service providers (SPs) are hosting virtual RR functionality in a cloud using servers. A vRR can run on a control plane device and can be placed anywhere in the topology or in a SP data center. Cisco IOS XRv 9000 Router can be implemented

as vRR over a NFV platform in a SP data center. vRR allows SPs to scale memory and CPU usage of RR deployments significantly. Moving a RR out of its optimal placement requires vRRs to implement ORR functionality that calculates the best path from a RR client's point of view.

BGP ORR offers these benefits:

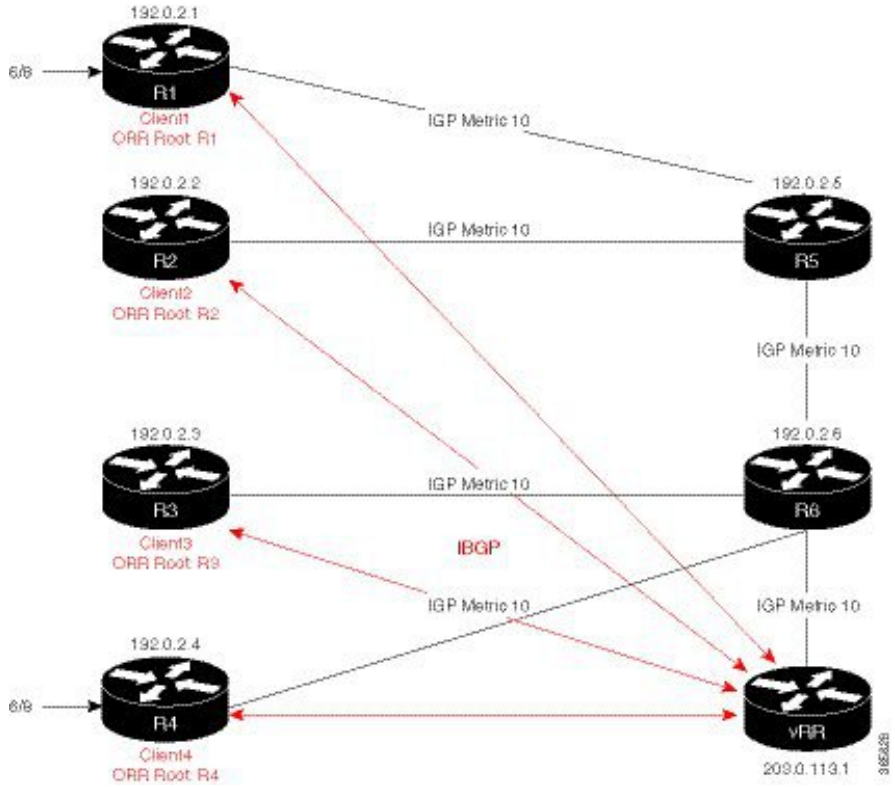
- calculates the bestpath from the point of view of a RR client.
- enables vRR to be placed anywhere in the topology or in a SP data center.
- allows SPs to scale memory and CPU usage of RR deployments.

Use Case

Consider a BGP Route Reflector topology where:

- Router R1, R2, R3, R4, R5 and R6 are route reflector clients
- Router R1 and R4 advertise 6/8 prefix to vRR

Figure 9: BGP-ORR Topology



vRR receives prefix 6/8 from R1 and R4. Without BGP ORR configured in the network, the vRR selects R4 as the closest exit point for RR clients R2, R3, R5, and R6, and reflects the 6/8 prefix learned from R4 to these RR clients R2, R3, R5, and R6. From the topology, it is evident that for R2 the best path is R1 and not R4. This is because the vRR calculates best path from the RR's point of view.

When the BGP ORR is configured in the network, the vRR calculates the shortest exit point in the network from R2's point of view (ORR Root: R2) and determines that R1 is the closest exit point to R2. vRR then reflects the 6/8 prefix learned from R1 to R2.

Configuring BGP ORR includes:

- enabling ORR on the RR for the client whose shortest exit point is to be determined
- applying the ORR configuration to the neighbor

Enabling ORR on vRR for R2 (RR client)

For example to determine shortest exit point for R2; configure ORR on vRR with an IP address of R2 that is 192.0.2.2. Use 6500 as AS number and g1 as orr (root) policy name:

```
router bgp 6500
 address-family ipv4 unicast
   optimal-route-reflection g1 192.0.2.2
commit
```

Applying the ORR configuration to the neighbor

Next, apply the ORR policy to BGP neighbor R2 (this enables RR to advertise best path calculated using the root IP address, 192.0.2.2, configured in orr (root) policy g1 to R2):

```
router bgp 6500
 neighbor 192.0.2.2
  address-family ipv4 unicast
   optimal-route-reflection g1
commit
```

Verification

To verify whether R2 received the best exit, execute the **show bgp <prefix>** command (from R2) in EXEC mode. In the above example, R1 and R4 advertise the 6/8 prefix; run the **show bgp 6.0.0.0/8** command:

```
R2# show bgp 6.0.0.0/8
Tue Apr  5 20:21:58.509 UTC
BGP routing table entry for 6.0.0.0/8
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 8           8
Last Modified: Apr  5 20:00:44.022 for 00:21:14
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.0.2.1 (metric 20) from 203.0.113.1 (192.0.2.1)
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 1, version 8
      Originator: 192.0.2.1, Cluster list: 203.0.113.1
```

The above show output states that the best path for R2 is through R1, whose IP address is 192.0.2.1 and the metric of the path is 20.

Execute the **show bgp** command from the vRR to determine the best path calculated for R2 by ORR. R2 has its own update-group because it has a different best path (or different policy configured) than those of other peers:

```
VRR#show bgp 6.0.0.0/8
Thu Apr 28 13:36:42.744 UTC
BGP routing table entry for 6.0.0.0/8
Versions:
Process bRIB/RIB SendTblVer
Speaker 13 13
Last Modified: Apr 28 13:36:26.909 for 00:00:15
Paths: (2 available, best #2)
Advertised to update-groups (with more than one peer):
0.2
Path #1: Received by speaker 0
ORR bestpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.1 (metric 30) from 192.0.2.1 (192.0.2.1)
Origin incomplete, metric 0, localpref 100, valid, internal, add-path
Received Path ID 0, Local Path ID 2, version 13
Path #2: Received by speaker 0
Advertised to update-groups (with more than one peer):
0.2
ORR addpath for update-groups (with more than one peer):
0.1
Local, (Received from a RR-client)
192.0.2.4 (metric 20) from 192.0.2.4 (192.0.2.4)
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
Received Path ID 0, Local Path ID 1, version 13
```

Note Path #1 is advertised to update-group 0.1. R2 is in update-group 0.1.

Execute the **show bgp** command for update-group 0.1 verify whether R2 is in update-group 0.1.

```
VRR#show bgp update-group 0.1
Thu Apr 28 13:38:18.517 UTC

Update group for IPv4 Unicast, index 0.1:
Attributes:
Neighbor sessions are IPv4
Internal
Common admin
First neighbor AS: 65000
Send communities
Send GSHUT community if originated
Send extended communities
Route Reflector Client
ORR root (configured): g1; Index: 0
4-byte AS capable
Non-labeled address-family capable
Send AIGP
Send multicast attributes
Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 5, replicated: 5
All neighbors are assigned to sub-group(s)
Neighbors in sub-group: 0.2, Filter-Groups num:1
```

```
Neighbors in filter-group: 0.2(RT num: 0)
192.0.2.2
```

For further verification, check the contents of the table created on vRR as a result of configuring the g1 policy. From R2's point of view, the cost of reaching R1 is 20 and the cost of reaching R4 is 30. Therefore, the closest and best exit for R2 is through R1:

```
VRR#show orrspf database g1
Thu Apr 28 13:39:20.333 UTC

ORR policy: g1, IPv4, RIB tableid: 0xe0000011
Configured root: primary: 192.0.2.2, secondary: NULL, tertiary: NULL
Actual Root: 192.0.2.2, Root node: 2000.0100.1002.0000

Prefix Cost
203.0.113.1 30
192.0.2.1 20
192.0.2.2 0
192.0.2.3 30
192.0.2.4 30
192.0.2.5 10
192.0.2.6 20

Number of mapping entries: 8
```

RPL - if prefix is-best-path/is-best-multipath

Border Gateway Protocol (BGP) routers receive multiple paths to the same destination. As a standard, by default the BGP best path algorithm decides the best path to install in IP routing table. This is used for traffic forwarding.

BGP assigns the first valid path as the current best path. It then compares the best path with the next path in the list. This process continues, until BGP reaches the end of the list of valid paths. This contains all rules used to determine the best path. When there are multiple paths for a given address prefix, BGP:

- Selects one of the paths as the best path as per the best-path selection rules.
- Installs the best path in its forwarding table. Each BGP speaker advertises only the best-path to its peers.

Note The advertisement rule of sending only the best path does not convey the full routing state of a destination, present on a BGP speaker to its peers.

After the BGP speaker receives a path from one of its peers; the path is used by the peer for forwarding packets. All other peers receive the same path from this peer. This leads to a consistent routing in a BGP network. To improve the link bandwidth utilization, most BGP implementations choose additional paths satisfy certain conditions, as multi-path, and install them in the forwarding table. Incoming packets for such are load-balanced across the best-path and the multi-path(s). You can install the paths in the forwarding table that are not advertised to the peers. The RR route reflector finds out the best-path and multi-path. This way the route reflector uses different communities for best-path and multi-path. This feature allows BGP to signal the local decision done by RR or Border Router. With this new feature, selected by RR using community-string (if is-best-path then community 100:100). The controller checks which best path is sent to all R's. Border Gateway Protocol routers receive multiple paths to the same destination. While carrying out best path computation

there will be one best path, sometimes equal and few non-equal paths. Thus, the requirement for abest-path and is-equal-best-path.

The BGP best path algorithm decides the best path in the IP routing table and used for forwarding traffic. This enhancement within the RPL allows creating policy to take decisions. Adding community-string for local selection of best path. With introduction of BGP Additional Path (Add Path), BGP now signals more than the best Path. BGP can signal the best path and the entire path equivalent to the best path. This is in accordance to the BGP multi-path rules and all backup paths.

Route Reflect Add-Path Control Per Neighbor

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Route Reflect Add-Path Control Per- Neighbor	Release 7.3.1	This feature allows you to configure multipath protection. Multipath protection functionality allows you to install and advertise multipath protection along with multipath. This gives additional flexibility if the backup path selection is not preferred or chosen. The **multipath-protect-advertise** keyword is added to the **set path-selection**command.

The **multipath-protect-advertise** command allows you to install and advertise the multipath protection along the multipath. You can use this option when you do not prefer the backup path selection.

Prior to this release, the device advertised only the bestpath and backup path when backup path selection is selected with the advertise option. Multipath protection was installed and was not advertised.

Remotely Triggered Null Route Filtering with RPL Next-hop Discard Configuration

Remotely triggered black hole (RTBH) filtering is a technique that provides the ability to drop undesirable traffic before it enters a protected network. RTBH filtering provides a method for quickly dropping undesirable traffic at the edge of the network, based on either source addresses or destination addresses by forwarding it to a null0 interface. RTBH filtering based on a destination address is commonly known as Destination-based RTBH filtering. Whereas, RTBH filtering based on a source address is known as Source-based RTBH filtering.

RTBH filtering is one of the many techniques in the security toolkit that can be used together to enhance network security in the following ways:

- Effectively mitigate DDoS and worm attacks
- Quarantine all traffic destined for the target under attack
- Enforce blocklist filtering

Note RTBH is not supported in cases such as L3VPN iBGP route over NULL0.

Note On Jericho2 TCAM-based platforms, when you configure a NULL interface, both destination-based RTBH filtering (D-RTBH) and source-based RTBH filtering (S-RTBH) are triggered.

Configuring Destination-based RTBH Filtering

RTBH is implemented by defining a route policy (RPL) to discard undesirable traffic at next-hop using **set next-hop discard** command.

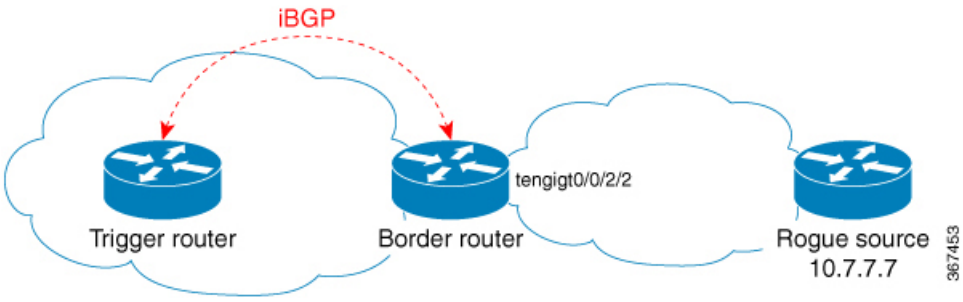
RTBH filtering sets the next-hop of the victim's prefix to the null interface. The traffic destined to the victim is dropped at the ingress.

The **set next-hop discard** configuration is used in the neighbor inbound policy. When this config is applied to a path, though the primary next-hop is associated with the actual path but the RIB is updated with next-hop set to Null0. Even if the primary received next-hop is unreachable, the RTBH path is considered reachable and will be a candidate in the bestpath selection process. The RTBH path is readvertised to other peers with either the received next-hop or nexthop-self based on normal BGP advertisement rules.

A typical deployment scenario for RTBH filtering would require running internal Border Gateway Protocol (iBGP) at the access and aggregation points and configuring a separate device in the network operations center (NOC) to act as a trigger. The triggering device sends iBGP updates to the edge, that cause undesirable traffic to be forwarded to a null0 interface and dropped.

Consider below topology, where a rogue router is sending traffic to a border router.

Figure 10: Topology to Implement RTBH Filtering



Configurations applied on the Trigger Router

Configure a static route redistribution policy that sets a community on static routes marked with a special tag, and apply it in BGP:

```
route-policy RTBH-trigger
  if tag is 777 then
    set community (1234:4321, no-export) additive
    pass
  else
    pass
  endif
  end-policy

router bgp 65001
 address-family ipv4 unicast
  redistribute static route-policy RTBH-trigger
 !
```

```
 neighbor 192.168.102.1
  remote-as 65001
  address-family ipv4 unicast
   route-policy bgp_all in
   route-policy bgp_all out
```

Configure a static route with the special tag for the source prefix that has to be block-holed:

```
router static
 address-family ipv4 unicast
 10.7.7.7/32 Null0 tag 777
```

Configurations applied on the Border Router

Configure a route policy that matches the community set on the trigger router and configure set next-hop discard:

```
route-policy RTBH
  if community matches-any (1234:4321) then
    set next-hop discard
  else
    pass
  endif
end-policy
```

Apply the route policy on the iBGP peers:

```
router bgp 65001
 address-family ipv4 unicast
 !
 neighbor 192.168.102.2
  remote-as 65001
  address-family ipv4 unicast
   route-policy RTBH in
   route-policy bgp_all out
```

Verification

On the border router, the prefix 10.7.7.7/32 is flagged as Nexthop-discard:

```
RP/0/RSP0/CPU0:router#show bgp
BGP router identifier 10.210.0.5, local AS number 65001
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000   RD version: 12
BGP main routing table version 12
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
N>i10.7.7.7/32        192.168.102.2            0    100      0 ?
```

```
RP/0/RSP0/CPU0:router#show bgp 10.7.7.7/32
BGP routing table entry for 10.7.7.7/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                12          12
Last Modified: Jul  4 14:37:29.048 for 00:20:52
Paths: (1 available, best #1, not advertised to EBGP peer)
```

```
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    192.168.102.2 (discarded) from 192.168.102.2 (10.210.0.2)
      Origin incomplete, metric 0, localpref 100, valid, internal best, group-best
      Received Path ID 0, Local Path ID 1, version 12
      Community: 1234:4321 no-export

RP/0/RSP0/CPU0:router#show route 10.7.7.7/32

Routing entry for 10.7.7.7/32
  Known via "bgp 65001", distance 200, metric 0, type internal
  Installed Jul 4 14:37:29.394 for 01:47:02
  Routing Descriptor Blocks
    directly connected, via Null0
      Route metric is 0
  No advertising protos.
```

Default Address Family for show Commands

Most of the **show** commands provide address family (AFI) and subaddress family (SAFI) arguments (see RFC 1700 and RFC 2858 for information on AFI and SAFI). The Cisco IOS XR software parser provides the ability to set the afi and safi so that it is not necessary to specify them while running a **show** command. The parser commands are:

- **set default-afi** { **ipv4** | **ipv6** | **all** }
- **set default-safi** { **unicast** | **multicast** | **all** }

The parser automatically sets the default afi value to **ipv4** and default safi value to **unicast** . It is necessary to use only the parser commands to change the default afi value from **ipv4** or default safi value from **unicast** . Any **afi** or **safi** keyword specified in a **show** command overrides the values set using the parser commands. Use the following **show default-afi-safi-vrf** command to check the currently set value of the afi and safi.

TCP Maximum Segment Size

Maximum Segment Size (MSS) is the largest amount of data that a computer or a communication device can receive in a single, unfragmented TCP segment. All TCP sessions are bounded by a limit on the number of bytes that can be transported in a single packet; this limit is MSS. TCP breaks up packets into chunks in a transmit queue before passing packets down to the IP layer.

The TCP MSS value is dependent on the maximum transmission unit (MTU) of an interface, which is the maximum length of data that can be transmitted by a protocol at one instance. The maximum TCP packet length is determined by both the MTU of the outbound interface on the source device and the MSS announced by the destination device during the TCP setup process. The closer the MSS is to the MTU, the more efficient is the transfer of BGP messages. Each direction of data flow can use a different MSS value.

Per Neighbor TCP MSS

The per neighbor TCP MSS feature allows you to create unique TCP MSS profiles for each neighbor. Per neighbor TCP MSS is supported in two modes: neighbor group and session group. Before, TCP MSS configuration was available only at the global level in the BGP configuration.

The per neighbor TCP MSS feature allows you to:

- Enable per neighbor TCP MSS configuration.
- Disable TCP MSS for a particular neighbor in the neighbor group or session group using the **inheritance-disable** command.
- Unconfigure TCP MSS value. On unconfiguration, TCP MSS value in the protocol control block (PCB) is set to the default value.

Note The default TCP MSS value is 536 (in octets) or 1460 (in bytes). The MSS default of 1460 means that TCP segments the data in the transmit queue into 1460-byte chunks before passing the packets to the IP layer.

To configure per neighbor TCP MSS, use the **tcp mss** command under per neighbor, neighbor group or session group configuration.

For detailed configuration steps, see Configuring Per Neighbor TCP MSS, on page 143.

For detailed steps to disable per neighbor TCP MSS, see Disabling Per Neighbor TCP MSS, on page 145.

MPLS VPN Carrier Supporting Carrier

Carrier supporting carrier (CSC) is a term used to describe a situation in which one service provider allows another service provider to use a segment of its backbone network. The service provider that provides the segment of the backbone network to the other provider is called the *backbone carrier*. The service provider that uses the segment of the backbone network is called the *customer carrier*.

A backbone carrier offers Border Gateway Protocol and Multiprotocol Label Switching (BGP/MPLS) VPN services. The customer carrier can be either:

- An Internet service provider (ISP) (By definition, an ISP does not provide VPN service.)
- A BGP/MPLS VPN service provider

You can configure a CSC network to enable BGP to transport routes and MPLS labels between the backbone carrier provider edge (PE) routers and the customer carrier customer edge (CE) routers using multiple paths. The benefits of using BGP to distribute IPv4 routes and MPLS label routes are:

- BGP takes the place of an Interior Gateway Protocol (IGP) and Label Distribution Protocol (LDP) in a VPN routing and forwarding (VRF) table. You can use BGP to distribute routes and MPLS labels. Using a single protocol instead of two simplifies the configuration and troubleshooting.
- BGP is the preferred routing protocol for connecting two ISPs, mainly because of its routing policies and ability to scale. ISPs commonly use BGP between two providers. This feature enables those ISPs to use BGP.

For detailed information on configuring MPLS VPN CSC with BGP, see the *Implementing MPLS Layer 3 VPNs on* Cisco ASR 9000 Series Router module of the *MPLS Configuration Guide for Cisco ASR 9000 Series Routers*.

BGP Keychains

BGP keychains enable keychain authentication between two BGP peers. The BGP endpoints must both comply with draft-bonica-tcp-auth-05.txt and a keychain on one endpoint and a password on the other endpoint does not work.

See the *System Security Configuration Guide for Cisco ASR 9000 Series Routers* for information on keychain management.

BGP is able to use the keychain to implement hitless key rollover for authentication. Key rollover specification is time based, and in the event of clock skew between the peers, the rollover process is impacted. The configurable tolerance specification allows for the accept window to be extended (before and after) by that margin. This accept window facilitates a hitless key rollover for applications (for example, routing and management protocols).

The key rollover does not impact the BGP session, unless there is a keychain configuration mismatch at the endpoints resulting in no common keys for the session traffic (send or accept).

BGP Session Authentication and Integrity using TCP Authentication Option Overview

BGP Session Authentication and Integrity using TCP Authentication Option feature enables you to use stronger Message Authentication Codes that protect against replays, even for long-lived TCP connections. This feature also provides more details on the association of security with TCP connections than TCP MD5 Signature option (TCP MD5).

This feature supports the following functionalities of TCP MD5:

- Protection of long-lived connections such as BGP and LDP.
- Support for larger set of MACs with minimal changes to the system and operations

BGP Session Authentication and Integrity using TCP Authentication Option feature supports IPv6. It supports these two cryptographic algorithms: HMAC-SHA-1-96 and AES-128-CMAC-96.

You can use two sets of keys, namely Master Key Tuples and traffic keys to authenticate incoming and outgoing segments.

This feature applies different option identifier than TCP MD5. This feature cannot be used simultaneously with TCP MD5.

Master Key Tuples

Traffic keys are the keying material used to compute the message authentication codes of individual TCP segments.

The BGP Session Authentication and Integrity using TCP Authentication Option (AO) feature uses the existing keychain fucntionality to define the key string, message authentication codes algorithm, and key lifetimes.

Master Key Tuples (MKTs) enable you to derive unique traffic keys, and to include the keying material required to generate those traffic keys. MKTs indicate the parameters under which the traffic keys are configured. The parameters include whether TCP options are authenticated, and indicators of the algorithms used for traffic key derivation and MAC calculation.

Each MKT has two identifiers, namely **SendID** and a **RecvID**. The SendID identifier is inserted as the KeyID identifier of the TCP AO option of the outgoing segments.The **RecvID** is matched against the TCP AO KeyID of the incoming segments.

Configure BGP Session Authentication and Integrity using TCP Authentication Option

This section describes how you can configure BGP Session Authentication and Integrity using TCP Authentication Option (TCP AO) feature :

- Configure Keychain

Note Configure send-life and accept-lifetime keywords with identical values in the keychain configuration, otherwise the values become invalid.

- Configure TCP

Note The Send ID and Receive ID you configured on the device must match the Receive ID and Send ID configured on the peer respectively.

- Configure BGP

Configuration Example

Configure a keychain.

```
Router# configure
Router#(config)# key chain tcpao1
Router#(config-tcpao1)# key 1
Router#(config-tcpao1-1)# cryptographic-algorithm HMAC-SHA-1-96
Router#(config-tcpao1-1)# key-string keys1
Router#(config-tcpao1-1)# send-lifetime 16:00:00 march 3 2018 infinite
Router#(config-tcpao1-1)# accept-lifetime 16:00:00 march 3 2018 infinite
```

Configure TCP

```
Router# tcp ao
Router(config-tcp-ao)# keychain tcpao1
Router(config-tcp-ao-tpcao1)# key 1 sendID 5 receiveID 5
/* Configure BGP */
Router#(config-bgp)# router bgp 1
Router(config-bgp)# bgp router-id 10.101.101.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.51.51.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# ao tcpao1 include-tcp-options disable accept-ao-mismatch-connection
```

Configure BGP

```
Router#(config-bgp)# router bgp 1
Router(config-bgp)# bgp router-id 10.101.101.1
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.51.51.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# ao tcpao1 include-tcp-options disable accept-ao-mismatch-connection
```

Verification

Verify the keychain information configured for BGP Session Authentication and Integrity using TCP Authentication Option feature.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays details of a key, such as Send Id, Receive Id, and cryptographic algorithm.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays the state of the BGP neighbors.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays the state of a particular BGP neighbor.

```
Router# show bgp sessions | i 10.51.51.1
Wed Mar 21 12:55:57.812 UTC
10.51.51.1  default  1     1     0     0    Established  None
```

The following output displays brief information of the protocol control block (PCB) of the neighbor.

```
Router# show tcp brief | i 10.51.51.2
Wed Mar 21 12:55:13.652 UTC
0x143df858 0x60000000  0  0  10.51.51.2:43387  10.51.51.1:179 ESTAB
```

The following output displays authentication details of the PCB:

```
Router# show tcp detail pcb 0x143df858 location 0/rsp0/CPU0 | begin Authen
Wed Mar 21 12:56:46.129 UTC
Authentication peer details:
    Peer: 10.51.51.1/32, OBJ_ID: 0x40002fd8
    Port: BGP, vrf_id: 0x60000000, type: AO, debug_on:0
    Keychain_name: tcpao1, options: 0x00000000, linked peer: 0x143e00 □ Keychain name
    Send_SNE: 0, Receive_SNE: 0, Send_SNE_flag: 0
    Recv_SNE_flag: 0, Prev_send_seq: 4120835405, Prev_receive_seq: 2461932863
    ISS: 4120797604, IRS: 2461857361
    Current key: 2
    Traffic keys: send_non_SYN: 006a2975, recv_non_SYN: 00000000
```

```
RNext key: 2
Traffic keys: send_non_SYN: 00000000, recv_non_SYN: 00000000
Last 1 keys used:
    key: 2, time: Mar 20 03:52:35.969.151, reason: No current key set
```

BGP Nonstop Routing

The Border Gateway Protocol (BGP) Nonstop Routing (NSR) with Stateful Switchover (SSO) feature enables all bgp peerings to maintain the BGP state and ensure continuous packet forwarding during events that could interrupt service. Under NSR, events that might potentially interrupt service are not visible to peer routers. Protocol sessions are not interrupted and routing states are maintained across process restarts and switchovers.

BGP NSR provides nonstop routing during the following events:

- Route processor switchover
- Process crash or process failure of BGP or TCP

Note BGP NSR is enabled by default. Use the **nsr disable** command to turn off BGP NSR. The **no nsr disable** command can also be used to turn BGP NSR back on if it has been disabled.

In case of process crash or process failure, NSR will be maintained only if **nsr process-failures switchover** command is configured. In the event of process failures of active instances, the **nsr process-failures switchover** configures failover as a recovery action and switches over to a standby route processor (RP) or a standby distributed route processor (DRP) thereby maintaining NSR. An example of the configuration command is RP/0/RSP0/CPU0:router(config) # nsr process-failures switchover

The **nsr process-failures switchover** command maintains both the NSR and BGP sessions in the event of a BGP or TCP process crash. Without this configuration, BGP neighbor sessions flap in case of a BGP or TCP process crash. This configuration does not help if the BGP or TCP process is restarted in which case the BGP neighbors are expected to flap.

When the *l2vpn_mgr* process is restarted, the NSR client (te-control) flaps between the **Ready** and **Not Ready** state. This is the expected behavior and there is no traffic loss.

During route processor switchover and In-Service System Upgrade (ISSU), NSR is achieved by stateful switchover (SSO) of both TCP and BGP.

NSR does not force any software upgrades on other routers in the network, and peer routers are not required to support NSR.

When a route processor switchover occurs due to a fault, the TCP connections and the BGP sessions are migrated transparently to the standby route processor, and the standby route processor becomes active. The existing protocol state is maintained on the standby route processor when it becomes active, and the protocol state does not need to be refreshed by peers.

Events such as soft reconfiguration and policy modifications can trigger the BGP internal state to change. To ensure state consistency between active and standby BGP processes during such events, the concept of post-it is introduced that act as synchronization points.

BGP NSR provides the following features:

- NSR-related alarms and notifications
- Configured and operational NSR states are tracked separately
- NSR statistics collection
- NSR statistics display using **show** commands
- XML schema support
- Auditing mechanisms to verify state synchronization between active and standby instances
- CLI commands to enable and disable NSR
- Support for 5000 NSR sessions

BGP Local Label Retention

When a primary PE-CE link fails, BGP withdraws the route corresponding to the primary path along with its local label and programs the backup path in the Routing Information Base (RIB) and the Forwarding Information Base (FIB), by default.

However, until all the internal peers of the primary PE reconverge to use the backup path as the new bestpath, the traffic continues to be forwarded to the primary PE with the local label that was allocated for the primary path. Hence the previously allocated local label for the primary path must be retained on the primary PE for some configurable time after the reconvergence. BGP Local Label Retention feature enables the retention of the local label for a specified period. If no time is specified, the local lable is retained for a default value of five minutes.

The **retain local-label** command enables the retention of the local label until the network is converged.

Command Line Interface (CLI) Consistency for BGP Commands

From Cisco IOS XR Release 3.9.0 onwards, the Border Gateway Protocol (BGP) commands use **disable** keyword to disable a feature. The keyword **inheritance-disable** disables the inheritance of the feature properties from the parent level.

BGP Additional Paths

The Border Gateway Protocol (BGP) Additional Paths feature modifies the BGP protocol machinery for a BGP speaker to be able to send multiple paths for a prefix. This gives 'path diversity' in the network. The add path enables BGP prefix independent convergence (PIC) at the edge routers.

BGP add path enables add path advertisement in an iBGP network and advertises the following types of paths for a prefix:

- Backup paths—to enable fast convergence and connectivity restoration.
- Group-best paths—to resolve route oscillation.

- All paths—to emulate an iBGP full-mesh.

Note Add path is not supported with MDT, IPv4 tunnel, IPv4/IPv6 Multicast, VPLS, and RT constraint.

iBGP Multipath Load Sharing

When a Border Gateway Protocol (BGP) speaking router that has no local policy configured, receives multiple network layer reachability information (NLRI) from the internal BGP (iBGP) for the same destination, the router will choose one iBGP path as the best path. The best path is then installed in the IP routing table of the router.

The iBGP Multipath Load Sharing feature enables the BGP speaking router to select multiple iBGP paths as the best paths to a destination. The best paths or multipaths are then installed in the IP routing table of the router.

When there are multiple border BGP routers having reachability information heard over eBGP, if no local policy is applied, the border routers will choose their eBGP paths as best. They advertise that bestpath inside the ISP network. For a core router, there can be multiple paths to the same destination, but it will select only one path as best and use that path for forwarding. iBGP multipath load sharing adds the ability to enable load sharing among multiple equi-distant paths.

Configuring multiple iBGP best paths enables a router to evenly share the traffic destined for a particular site.

The iBGP Multipath Load Sharing feature functions similarly in a Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) with a service provider backbone.

For multiple paths to the same destination to be considered as multipaths, the following criteria must be met:

- All attributes must be the same. The attributes include weight, local preference, autonomous system path (entire attribute and not just length), origin code, Multi Exit Discriminator (MED), and Interior Gateway Protocol (iGP) distance.
- The next hop router for each multipath must be different.

Even if the criteria are met and multiple paths are considered multipaths, the BGP speaking router will still designate one of the multipaths as the best path and advertise this best path to its neighbors.

Note After a change in multipath, IGP metrics are not considered while evaluating eiBGP multipath candidates and a sub-optimal path can be used.

Per-vrf label mode is not supported for Carrier Supporting Carrier (CSC) network with internal and external BGP multipath setup

Per VRF label mode cannot be used for BGP PIC edge with eiBGP multipath as that might cause loops. Only per prefix label supports per VRF label mode.

Persistent Loadbalancing

Traditional ECMP or equal cost multipath loadbalances traffic over a number of available paths towards a destination. When one path fails, the traffic gets re-shuffled over the available number of paths. This flow distribution can be a problem in data center loadbalancing.

Persistent Loadbalancing or Sticky ECMP defines a prefix in such a way that it do not rehash flows on existing paths and only replace those bucket assignments of the failed server. The advantage is that the established sessions to servers will not get rehashed.

The following section describes how you can configure persistent load balancing:

```
/*Configure persistent load balancing. */

Router(config)# router bgp 7500
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy sticky-ecmp
Router(config-bgp-af)# bgp attribute-download
Router(config-bgp-af)# maximum-paths ebgp 64
Router(config-bgp-af)# maximum-paths ibgp 32
Router(config-bgp-af)# exit
Router(config-bgp)# exit
Router(config)# route-policy sticky-ecmp
Router(config-rpl)# if destination in (192.1.1.1/24) then
Router(config-rpl-if)# set load-balance ecmp-consistent
Router(config-rpl-if)# else
Router(config-rpl-else)# pass
Router(config-rpl-else)# endif
RP/0/0/CPU0:ios(config-rpl)# end-policy
RP/0/0/CPU0:ios(config)#

/* Enable autocovery and hence recover the original hashing state
after failed paths become active. */
Router(config)# cef consistent-hashing auto-recovery

/* Recover to the original hashing state after failed paths come up
and avoid affecting newly formed flows after path failure.  */
Router(config)# clear route 192.0.2.0/24
```

Running Configuration

```
/* Configure persistent loadbalancing. */
router bgp 7500
 address-family ipv4 unicast
  table-policy sticky-ecmp
  bgp attribute-download
  maximum-paths ebgp 64
  maximum-paths ibgp 32


cef consistent-hashing auto-recovery

clear route 192.0.2.0/24
```

Verification

Verify that the path distribution with persistent loadbalancing is configured.

The following show output displays the status of path distribution before a link fails. In this output, three paths are identified with three next hops (10.1/2/3.0.1) through three different GigabitEthernet interfaces.

```
show cef 192.0.2.0/24
```

```
 LDI Update time Sep  5 11:22:38.201
   via 10.1.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x57ac4e74 0x0]
    next hop 10.1.0.1/32 via 10.1.0.1/32
   via 10.2.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x57ac4a74 0x0]
    next hop 10.2.0.1/32 via 10.2.0.1/32
   via 10.3.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 2 NHID 0x0 [0x57ac4f74 0x0]
    next hop 10.3.0.1/32 via 10.3.0.1/32


    Load distribution (consistent): 0 1 2 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/0/0/0    10.1.0.1
    1     Y   GigabitEthernet0/0/0/1    10.2.0.1
    2     Y   GigabitEthernet0/0/0/2    10.3.0.1
```

The following show output displays the status of the path distribution after a link fails. The replacement of bucket 1 with GigabitEthernet 0/0/0/0 and the "*" symbol denotes that this path is a replacement for a failed path.

```
show cef 192.0.2.0/24
 LDI Update time Sep  5 11:23:13.434
   via 10.1.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 0 NHID 0x0 [0x57ac4e74 0x0]
    next hop 10.1.0.1/32 via 10.1.0.1/32
   via 10.3.0.1/32, 3 dependencies, recursive, bgp-multipath [flags 0x6080]
    path-idx 1 NHID 0x0 [0x57ac4f74 0x0]
    next hop 10.3.0.1/32 via 10.3.0.1/32

    Load distribution (consistent) : 0 1 2 (refcount 1)
    Hash  OK  Interface                 Address
    0     Y   GigabitEthernet0/0/0/0    10.1.0.1
 1*    Y   GigabitEthernet0/0/0/0    10.1.0.1
    2     Y   GigabitEthernet0/0/0/2    10.3.0.1
```

BGP Selective Multipath

Traditional BGP multipath feature allows a router receiving parallel paths to the same destination to install the multiple paths in the routing table. By default, this multipath feature is applied to all configured peers. BGP selective multipath allows application of the multipath feature only to selected peers.

The BGP router receiving multiple paths is configured with the **maximum-paths ... selective** option. The iBGP/eBGP neighbors sharing multiple paths are configured with the **multipath** option, while being added as neighbors on the BGP router.

Note Use **next-hop-unchanged multipath** command to avoid overwriting next-hop information before advertising multipaths.

The following behavior is to be noted while using BGP selective multipath:

- BGP selective multipath does not impact best path calculations. A best path is always included in the set of multipaths.

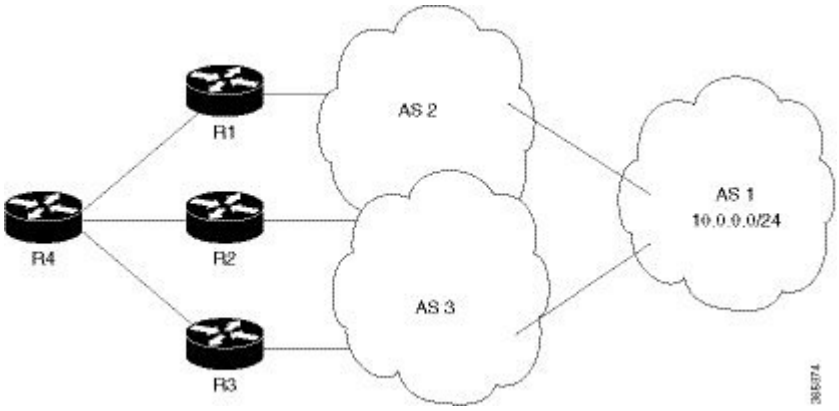
- For VPN prefixes, the PE paths are *always* eligible to be multipaths.

For information on the **maximum-paths** and **multipath** commands, see the *Cisco ASR 9000 Series Aggregation Services Router Routing Command Reference*.

Topology

A sample topology to illustrate the configuration used in this section is shown in the following figure.

Figure 11: BGP Selective Multipath



Router R4 receives parallel paths from Routers R1, R2 and R3 to the same destination. If Routers R1 and R2 are configured as selective multipath neighbors on Router R4, only the parallel paths from these routers are installed in the routing table of Router R4.

Configuration

Note Configure your network topology with iBGP/eBGP running on your routers, before configuring this feature.

To configure BGP selective multipath on Router R4, use the following steps.

1. Configure Router R4 to accept selective multiple paths in your topology.

```
/* To configure selective multipath for iBGP/eBGP
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths ibgp 4 selective
RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths ebgp 5 selective
RP/0/RSP0/CPU0:router(config-bgp-af)# commit

/* To configure selective multipath for eiBGP
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths eibgp 6 selective
RP/0/RSP0/CPU0:router(config-bgp-af)# commit
```

2. Configure neighbors for Router R4.

   Routers R1 (1.1.1.1) and R2 (2.2.2.2) are configured as neighbors with the **multipath** option.

Router R3 (3.3.3.3) is configured as a neighbor without the **multipath** option, and hence the routes from this router are not eligible to be chosen as multipaths.

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 1.1.1.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit

RP/0/RSP0/CPU0:router(config-bgp-nbr)# neighbor 2.2.2.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit

RP/0/RSP0/CPU0:router(config-bgp-nbr)# neighbor 3.3.3.3
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit
```

You have successfully configured the BGP selective multipath feature.

Accumulated Interior Gateway Protocol Attribute

The Accumulated Interior Gateway Protocol (AiGP)Attribute is an optional non-transitive BGP Path Attribute. The attribute type code for the AiGP Attribute is to be assigned by IANA. The value field of the AiGP Attribute is defined as a set of Type/Length/Value elements (TLVs). The AiGP TLV contains the Accumulated IGP Metric.

The AiGP feature is required in the 3107 network to simulate the current OSPF behavior of computing the distance associated with a path. OSPF/LDP carries the prefix/label information only in the local area. Then, BGP carries the prefix/lable to all the remote areas by redistributing the routes into BGP at area boundaries. The routes/labels are then advertised using LSPs. The next hop for the route is changed at each ABR to local router which removes the need to leak OSPF routes across area boundaries. The bandwidth available on each of the core links is mapped to OSPF cost, hence it is imperative that BGP carries this cost correctly between each of the PEs. This functionality is achieved by using the AiGP.

Per VRF and Per CE Label for IPv6 Provider Edge

The per VRF and per CE label for IPv6 feature makes it possible to save label space by allocating labels per default VRF or per CE nexthop.

All IPv6 Provider Edge (6PE) labels are allocated per prefix by default. Each prefix that belongs to a VRF instance is advertised with a single label, causing an additional lookup to be performed in the VRF forwarding table to determine the customer edge (CE) next hop for the packet.

However, use the **label mode** command with the **per-ce** keyword or the **per-vrf** keyword to avoid the additional lookup on the PE router and conserve label space.

Use **per-ce** keyword to specify that the same label be used for all the routes advertised from a unique customer edge (CE) peer router. Use the **per-vrf** keyword to specify that the same label is to be used for all the routes advertised from a unique VRF. In 6PE, the label is IPV6 explicit null label.

IPv4 BGP-Policy Accounting on Cisco ASR 9000's A9K-SIP-700

Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is sent to, or received from, different peers. Policy accounting is enabled on an individual input or output interface basis. Counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Using BGP policy accounting, you can account for traffic according to the route it traverses. Service providers can identify and account for all traffic by customer and bill accordingly.

For more information on BGP policy accounting and how to configure BGP policy accounting, refer the *Implementing Cisco Express Forwarding* module in *Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Configuration Guide*.

IPv6 Unicast Routing on Cisco ASR 9000's A9K-SIP-700

Cisco ASR 9000's A9K-SIP-700 provides complete Internet Protocol Version 6 (IPv6) unicast capability.

An IPv6 unicast address is an identifier for a single interface, on a single node. A packet that is sent to a unicast address is delivered to the interface identified by that address. Cisco IOS XR software supports the following IPv6 unicast address types:

- Global aggregatable address
- Site-local address
- Link-local address
- IPv4-compatible IPv6 address

For more information on IPv6 unicase addressing, refer the *Implementing Network Stack IPv4 and IPv6* module in *Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Configuration Guide*.

IPv6 uRPF Support on Cisco ASR 9000's A9K-SIP-700

Unicast IPv6 Reverse Path Forwarding (uRPF) mitigates problems caused by the introduction of malformed or spoofed IP source addresses into a network by discarding IP packets that lack a verifiable IP source address. Unicast RPF does this by doing a reverse lookup in the Cisco Express Forwarding (CEF) table. Therefore, uRPF is possible only if CEF is enabled on the router.

Use the **ipv6 verify unicast source reachable-via {any | rx} [allow-default] [allow-self-ping]** command in interface configuration mode to enable IPV6 uRPF.

For more information on IPv6 uRPF, refer *Implementing Cisco Express Forwarding* module in *IP Addresses and Services Command Reference for Cisco ASR 9000 Series Routers*

Remove and Replace Private AS Numbers from AS Path in BGP

Private autonomous system numbers (ASNs) are used by Internet Service Providers (ISPs) and customer networks to conserve globally unique AS numbers. Private AS numbers cannot be used to access the global Internet because they are not unique. AS numbers appear in eBGP AS paths in routing updates. Removing

private ASNs from the AS path is necessary if you have been using private ASNs and you want to access the global Internet.

Public AS numbers are assigned by InterNIC and are globally unique. They range from 1 to 64511. Private AS numbers are used to conserve globally unique AS numbers, and they range from 64512 to 65535. Private AS numbers cannot be leaked to a global BGP routing table because they are not unique, and BGP best path calculations require unique AS numbers. Therefore, it might be necessary to remove private AS numbers from an AS path before the routes are propagated to a BGP peer.

External BGP (eBGP) requires that globally unique AS numbers be used when routing to the global Internet. Using private AS numbers (which are not unique) would prevent access to the global Internet. The remove and replace private AS Numbers from AS Path in BGP feature allows routers that belong to a private AS to access the global Internet. A network administrator configures the routers to remove private AS numbers from the AS path contained in outgoing update messages and optionally, to replace those numbers with the ASN of the local router, so that the AS Path length remains unchanged.

The ability to remove and replace private AS numbers from the AS Path is implemented in the following ways:

- The **remove-private-as** command:
  - Removes private AS numbers from the AS path even if the path contains both public and private ASNs.
  - Removes private AS numbers even if the AS path contains only private AS numbers. There is no likelihood of a 0-length AS path because this command can be applied to eBGP peers only, in which case the AS number of the local router is appended to the AS path.
  - Removes private AS numbers even if the private ASNs appear before the confederation segments in the AS path.
- The **replace-as** command replaces the private AS numbers being removed from the path with the local AS number, thereby retaining the same AS path length.

The feature can be applied to a neighbor in the address family configuration mode. Therefore, if you apply the feature for a neighbor in an address family, only the outbound update messages are impacted.

Use **show bgp neighbors** and **show bgp update-group** commands to verify that the that private AS numbers were removed or replaced.

Replace BGP AS Path with Custom Values

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Replace BGP AS Path with Custom Values	Release 7.5.2	You can now configure route policies to replace the Autonomous System (AS) Path in BGP with custom values to control the best path selection process. This feature introduces the **replace as-path all** command.

BGP routers typically receive multiple paths to the same destination. The BGP best-path algorithm determines the best path to install in the IP routing table and to use for forwarding traffic. The overall best path is selected based on various attributes. .

AS path is one of the attributes used for best path selection. By default, BGP always prefers the route with shortest AS path as the best path. The best path selected by BGP might have traffic engineering issues, like heavy traffic that leads to congestion. In such cases, you can alter the best path by replacing the AS path with custom values.

The following are the custom values you can use to replace the AS path:

- **None**: Use this option to modify an AS path as the shortest path in the network. When you choose this option, the AS path is replaced with a null or empty value. Use the **replace as-path all none** command to replace with none.
- **Auto**: Use this option to advertise the local AS number or the neigbor's AS number as the AS path. When you choose this option, AS path is replaced based on the route policy:
  - For inbound route policy, AS path is replaced with AS path of BGP neighbor from where the prefix is received.
  - For outbound route policy, AS path is replaced with the local AS number.

  Use the **replace as-path all auto** command to replace with auto.
- **'x'**: Use this option to replace AS path with any specified value. Use the **replace as-path all** *'x'* command to replace with this option, where *'x'* can be a single AS number or a sequence of AS numbers separated by space.
- Optionally, you can repeat replacing the AS path for a specified number of times. This option is supported only for the **auto** and *'x'* parameters. Use the **replace as-path all** {**auto** | *'x'*} [*n*] command to enable the repeat option.
- Optionally, you can use a parameter name along with the repeat option. The parameter name must be preceded with a "$." You can attach the route policy with the parameter to a neighbor and specify the number of times the AS path replacement should be repeated. This opton allows you to apply the same route policy to different neighbors with different AS path values.

  Use the **replace as-path all** {**auto** | *'x'*} [*n*] [*parameter*] command to enable the parameter along with repeat option.

You can replace the AS path for inbound eBGP, outbound eBGP, and outbound iBGP paths.

Note For outbound eBGP paths, the AS number of the local router is always prepended to the replaced AS path.

Interoperability with BGP Confederation

BGP confederation is a group of multiple autonomous systems that looks like a single autonomous system to the outside world. When confederation is configured on BGP peers, the AS path is replaced as follows:

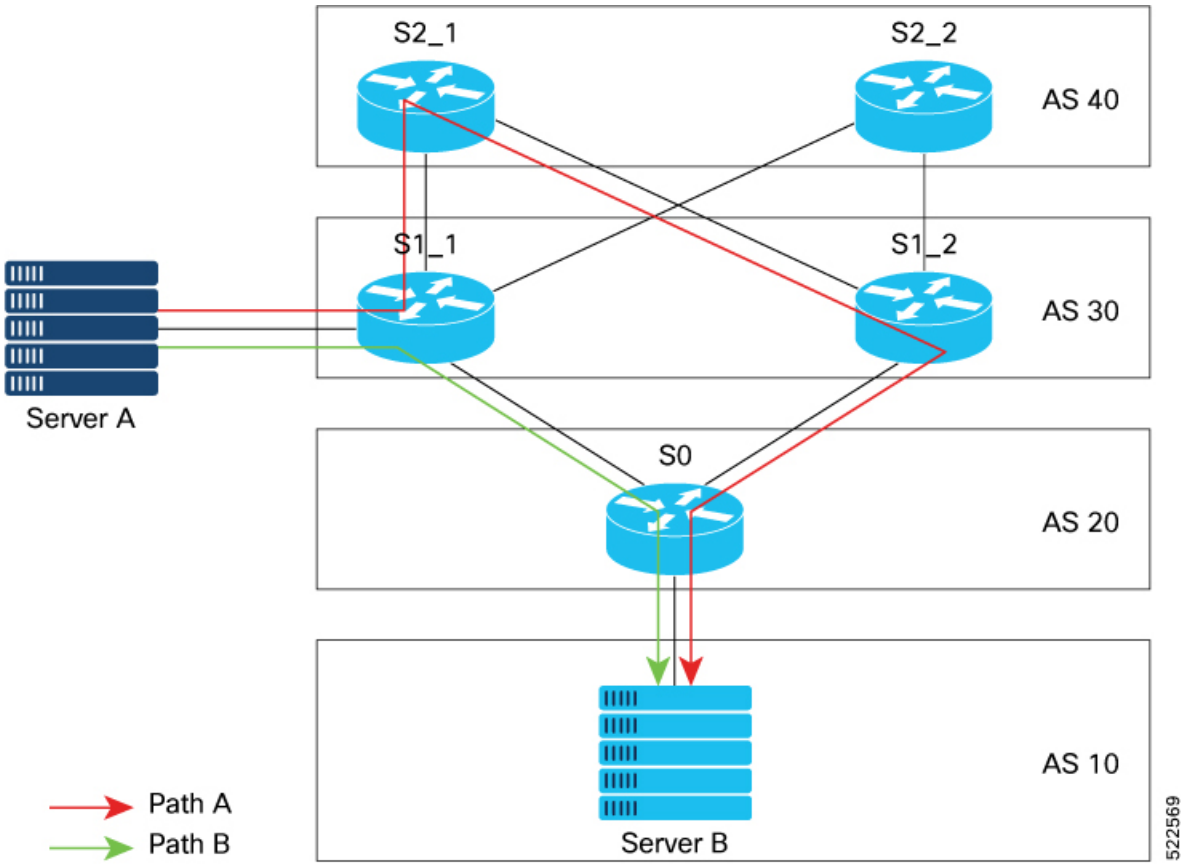
- When you replace the AS path in an outbound BGP router, which receives prefix from a BGP neighbor configured with confederation, the specified AS path value is appended to the confederation sequence.

- When you replace the AS path in an inbound BGP router configured with confederation, the confederation sequence is replaced with the specified AS path value.

Deployment Scenario

Consider a BGP network configured with AS paths. By default, BGP selects the route with shortest AS path to reach the destination. You can alter the default route by using the replace BGP AS path feature.

In the following figure, the network consists of BGP routers configured with AS Path values. To reach Server B, Server A typically selects Path B (via S1_1, S0), as the AS path value of S0 is shorter.



You may want to use Path A to reach the destination (via S1_1, S2_1, S1_2, S0), for traffic engineering purpose. For example, Path A may be less congested and is better than Path B. To use Path A, you can replace the AS path values with one of the following options:

- Replace AS path of Router S2_1 with a shorter value.
- Replace AS path of Router S0 with a longer value.

Restrictions

- The **replace as-path all** command isn't supported on inbound iBGP paths.
- The **replace as-path all** command isn't supported on a route policy that is already configured with **remove-private-as** or **replace as** commands.

- You can apply the route policy configured with **replace as-path all** only on neighbor-in or neighbor-out attach points.

Configuration Example

To replace BGP AS path with custom values, perform the following tasks on a BGP router:

This example shows how to replace AS path with null value.

```
/*Configure route policy to replace AS path with none*/
Router(config)#hw-module profile stats ?
Router(config)# route-policy aspath-none
Router(config-rpl)# replace as-path all none
Router(config-rpl)# end-policy

/* Apply route policy to BGP neighbor */
Router(config)# router bgp 65530
Router(config-bgp)# neighbor 111.0.0.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy aspath-none in
```

This example shows how to replace AS path with auto option.

```
/*Configure route policy to replace AS path with auto*/
Router(config)#route-policy aspath-auto
Router(config-rpl)# replace as-path all auto
Router(config-rpl)# end-policy


/* Apply route policy to BGP neighbor */
Router(config)# router bgp 65530
Router(config-bgp)# neighbor 111.0.0.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy aspath-auto out
```

This example shows how to replace AS path with a specified sequence of AS numbers. In this example, sequence **'10 100 200 300'** is used.

```
/*Configure route policy to replace AS path with 'x'*/
Router(config)# route-policy aspath-str
Router(config-rpl)# replace as-path all '10 100 200 300'
Router(config-rpl)# end-policy


/* Apply route policy to BGP neighbor */
Router(config)# router bgp 1
Router(config-bgp)# neighbor 111.0.0.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy aspath-str in
```

This example shows how to use **replace as-path all** command along with parameter to replace the AS path with specified sequence of values, repeated for specified number of times. In this example, AS path is replaced with sequence **'45 55'**, repeated for **6** times.

```
/*Configure route policy to replace AS path with parameter ($n)*/
Router(config)# route-policy aspath-par($n)
Router(config-rpl)# replace as-path all '45 55' $n
Router(config-rpl)# end-policy
```

```
/* Apply route policy to BGP neighbor */
Router(config)# router bgp 1
Router(config-bgp)# neighbor 111.0.0.1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy aspath-par(6) in
```

Verification

In the following output, AS path is replaced with **null** value.

```
Router# show bgp
Network               Next Hop            Metric LocPrf Weight Path
*> 192.168.3.0/24     192.168.3.1                      0      0    i
```

In the following output, AS path is replaced with auto for an outbound path, where the AS path of local router is **[40]**.

```
Router# show bgp
Network            Next Hop            Metric LocPrf Weight Path
*> 111.0.0.2/32    200.0.0.5                        0         40 i
```

In the following output, AS path is replaced with the sequence **'10 100 200 300'**.

```
Router# show bgp
Network            Next Hop            Metric LocPrf Weight Path
*>111.0.0.2/32      200.0.0.5                        0 10 100 200 300 i
```

In the following output, AS path is replaced with the sequence **'45 55'**, repeated for 6 times.

```
Router# show bgp
Network            Next Hop            Metric LocPrf Weight Path
*>111.0.0.8/32      200.0.0.5                        0        45 55 45 55 45 55 45 55 45 55
45 55 i
```

Configure Replace BGP AS Path with Custom Values

Perform the following steps to replace BGP AS path with custom values.

SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **replace as-path all** { **none** | **auto** | *x* } [*n*] [*parameter*]
4. Use the **show bgp** command to verify the replaced AS path.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **route-policy** *route-policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# route-policy test-match-all
```

Defines the route policy and enters route-policy configuration mode.

Step 3 **replace as-path all** { **none** | **auto** | *x* } [*n*] [*parameter*]

Replaces the entire AS path with specified values.

- **none** – Replaces AS path with null or empty value.
- **auto** – For Inbound route policy, replaces AS path with AS path of neighbor from where the prefix is received. For Outbound route policy, replaces AS path with the configured local AS path.
- *x* – Replaces AS path with specified value, where *x* can be a single AS number or a sequence of AS numbers separated by space.
- [*n*] – (Optional) Repeats the AS path replacement for specified number of times. Range is from 2 to 64. This option is supported only for the parameters **auto** and *x*.
- [*parameter*] – (Optional) Parameter name used along with repeat option. The parameter name must be preceded with a "$." You can attach the route policy with the parameter to a neighbor and specify the number of times the AS path replacement should be repeated.

Note You can apply the route policy configured with **replace as-path all** only to neighbor-in or neighbor-out attach points.

Example:

In this example, AS path is replaced with null value.

```
RP/0/RSP0/CPU0:router(config-rpl)# replace as-path all none
```

In this example, AS path is replaced with **auto**, repeated for 2 times.

```
RP/0/RSP0/CPU0:router(config-rpl)# replace as-path all auto 2
```

In this example, AS path is replaced with '77' for 3 times.

```
RP/0/RSP0/CPU0:router(config-rpl)# replace as-path all '77' 3
```

The following example uses parameter *$n* to replace the AS path with **auto**, repeated for 2 times.

```
RP/0/RSP0/CPU0:router(config)# route-policy replace-auto($n)
RP/0/RSP0/CPU0:router(config-rpl)# replace as-path all auto $n
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
RP/0/RSP0/CPU0:router(config)# router bgp 65530
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.101.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy replace-auto(2) in
```

Step 4 Use the **show bgp** command to verify the replaced AS path.

Example:

```
RP/0/RSP0/CPU0:router# show bgp
Network              Next Hop            Metric LocPrf Weight Path
```

```
*>i1.1.1.0/24         192.168.1.1              0    100      0 i
*> 2.2.2.0/24         0.0.0.0                  0         32768 i
*>i3.3.3.0/24         192.168.2.2              0    100      0 i
*> 4.4.4.0/24         192.168.3.1              0             0 35 35 1000 2000 i
*>i5.5.5.0/24         192.168.2.2              0    100      0 300 i
*> 192.168.3.0/24     192.168.3.1              0             0 35 35 1000 2000 i
*>i192.168.4.0/24     192.168.2.2              0    100      0 300 i
```

Selective VRF Download

Selective VRF Download (SVD) feature enables the downloading of only those prefixes and labels to a line card that are actively required to forward traffic through the line card.

To meet the demand for a consolidated edge MSE platform, the number of VRFs, VRF interfaces, and the prefix capacity increase. Convergence timings differ in different line card engines. One of the major factors that determine convergence timing is the time taken to process and program a prefix and its associated data structures. A lesser number of prefixes and labels ensure better convergence timing. By enabling selective download of VRF routes, SVD increases scalability and reduces convergence problems in Layer 3 VPNs (L3VPNs).

Line Card Roles and Filters in Selective VRF Download

In a selective VRF download (SVD) context, line cards have these roles:

- Core LC: a line card that has only core facing interfaces (interfaces that connect to other P/PEs)
- Customer LC: a line card that has one or more customer facing interfaces (interfaces that connect to CEs in different VRFs)

The line cards handle these prefixes:

- Local Prefix: a prefix that is received from a CE connected to the router in a configured VRF context
- Remote Prefix: a prefix received from another PE and is imported to a configured VRF

These filters are applicable to each line card type:

- A core LC needs all te local prefixes and VRF labels so that the label or IP forwarding, or both is set up correctly.
- A customer LC needs both local and remote prefixes for all the VRFs to which it is connected, and for other VRFs which some connected VRFs have dependency. This is based on the import/export RT configuration; VRF ‘A’ may have imported routes from VRF ‘B’, so the imported route in VRF ‘A’ points to a next-hop that is in VRF ‘B’. For route resolution, VRF ‘B’ routes need to be downloaded to each line card that has a VRF ‘A’ interface.
- If a line card is hosts both core facing and customer facing interfaces, then it does not need to do any filtering. All tables and all routes are present on such line cards. These line cards have a role called “standard”. All RPs and DRPs have the standard role.
- To correctly resolve L3VPN routes, the IPv4 default table needs to be present an all nodes. However, if the line card does not have any IPv6 interface, it can filter out all IPv6 tables and routes. In such a case, the line card can be deemed “not interested” in the IPv6 AFI. Then it behaves as if IPv6 is not supported by it.

Selective VRF Download Disable

Selective VRF Download (SVD) functionality is disabled, by default. To enable SVD, configure the **svd platform enable** command in administrative configuration mode and reload the chassis using the **reload location all** command. To disable SVD that is already enabled, use the **no svd platform enable** command and reload the chassis using the **reload location all** command.

Calculating Routes Downloaded to Line Card with or without SVD

The number of routes that will be downloaded to the line card with or without selective VRF download option can be calculated by following the Total Tables and Routes Downloaded by Line Card Type table below.

This table summarizes the total routes and tables downloaded on the line cards of each SVD card type. Savings can be calculated by the difference between the numbers in the Without SVD row.

Table 5: Total Tables and Routes Downloaded by Line Card Type

Card Type	Tables Downloaded	Routes Downloaded
Customer	(o+Y)	(o+Y)R
Core	n	nxR
Without SVD	n	nR

- n is the total number of VRFs present
- o is the number of VRF directly provisioned/configured on the card, (n is greater than or equal to o)
- R is routes per VRF
- x is the ratio of SVD local: total routes
- Y is the number of VRFs dependant on directly provisioned VRFs (o), (Y is greater than or equal to 0)

Here is an example calculation:

A customer has 100 VRFs configured on the system, with five line cards. For the IPv4 address family, four line cards are working as customer facing with equal VRF distribution, while one is core facing. Inter-table dependencies do not exist. In this example, n = 100, o = 25, x = 3/10, Y = 0, and R = 1000.

Number of routes downloaded:

- Without SVD: (nR) = 100,000
- On customer-facing card: (o+Y)R = 25,000
- On core-facing card: (nxR) = 30,000

In this example, the SVD feature brings close to 70 per cent savings.

The total number of VRFs present (n) can be found by using the **show cef tables summary location** *node-id* command on the RSP card.

```
RP/0/RSP0/CPU0:router#show cef tables summary location 0/rsp0/cpu0

Role change timestamp     : Apr  3 07:21:46.759
Current Role              : Core
No. of times Eod received : 2
```

```
Eod received              : Apr  3 07:21:46.980

No. of Tables                 :       106
No. of Converged Tables       :       106
No. of Deleted Tables         :         0
No. of Bcdl Subscribed Tables :       106
No. of Marked Tables          :         0
```

The number of VRFs provisioned on the line card (o) is derived from the "No. Of Tables" field in the **show cef tables summary location 0/0/cpu0**. This provides the tables specific to the Linecard 0/0/cpu0.

The routes per VRF (R) can be found using the **show cef tables location** *node-id* command.

```
RP/0/RSP0/CPU0:router#show cef tables location 0/1/CPU0
Sat Apr  6 01:22:32.471 UTC

Codes:   L - SVD Local Routes, R - SVD Remote Routes
         T - Total Routes
         C - Table Converged,  D - Table Deleted
         M - Table Marked,     S - Table Subscribed

Table           Table ID             L          R           T C D M S
default         0xe0000000           9          3          23 Y N N Y
**nVSatellite   0xe0000010           1          0           6 Y N N Y
cdn             0xe0000011           0          0           5 Y N N Y
oir             0xe0000012           0          0           5 Y N N Y
vrf1            0xe0000013           3          1          11 Y N N Y
```

For the vrf "vrf1" the total routes is in the "T" column which is 11. So if the number of routes per VRF are not the same for all vrfs then total of "routes in all non-default" vrfs will have to be calculated and divided by the number of VRFs, to arrive at the Average Routes per VRF.

The ratio of SVD local: total routes (x) can be found using the number of SVD Local Routes and the number of Total Routes for a given VRF. For example, in the above sample output of **show cef tables location 0/1/CPU0**, in the L column, the number represents the Local Routes, and in the T column number represents Total routes in that Vrf. So ratio of L column to T column number will give the ratio for a given vrf. Again if the ratio is not same for all vrfs, it will have to be averaged out over all vrfs.

The number of VRFs dependant on directly provisioned VRFs (Y) will have to manually calculated because it depends on the router configuration. For example, if route import targets in Dependant VRF import from routes exported by other VRFs. A VRF is dependant if it depends on a nexthops being in some other VRF which is directly provisioned. There is no show command to automatically calculate Y, since it depends completely on the way router is configured to import routes in various VRFs.

BGP Accept Own

The BGP Accept Own feature enables handling of self-originated VPN routes, which a BGP speaker receives from a route-reflector (RR). A "self-originated" route is one which was originally advertized by the speaker itself. As per BGP protocol [RFC4271], a BGP speaker rejects advertisements that were originated by the speaker itself. However, the BGP Accept Own mechanism enables a router to accept the prefixes it has advertised, when reflected from a route-reflector that modifies certain attributes of the prefix. A special community called ACCEPT-OWN is attached to the prefix by the route-reflector, which is a signal to the receiving router to bypass the ORIGINATOR_ID and NEXTHOP/MP_REACH_NLRI check. Generally, the BGP speaker detects prefixes that are self-originated through the self-origination check (ORIGINATOR_ID,

NEXTHOP/MP_REACH_NLRI) and drops the received updates. However, with the Accept Own community present in the update, the BGP speaker handles the route.

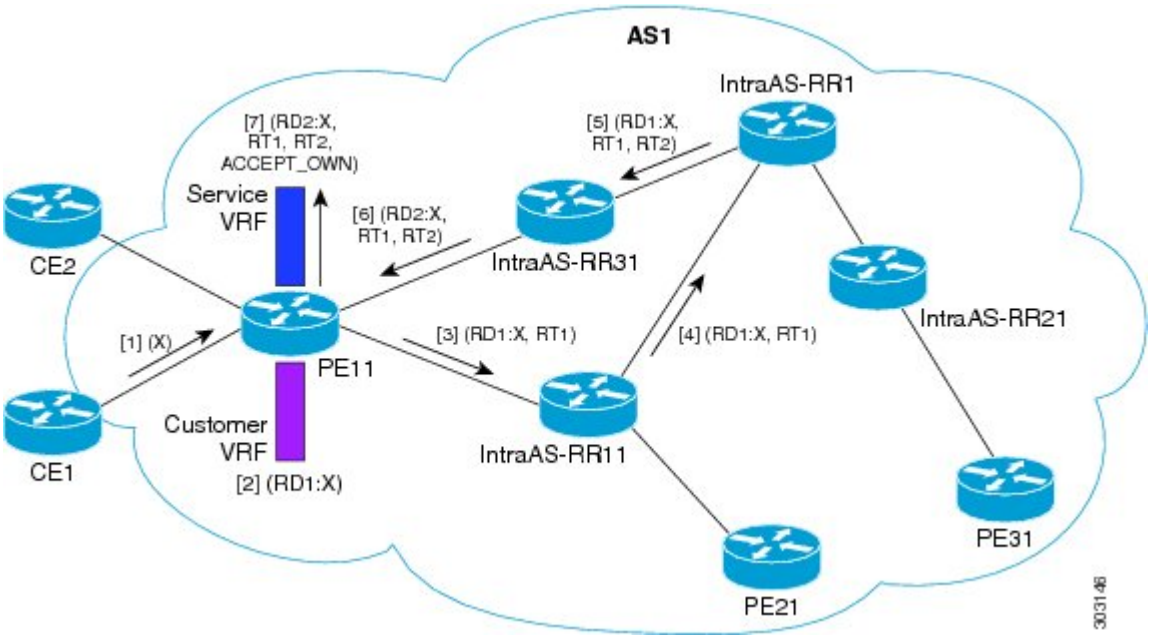
One of the applications of BGP Accept Own is auto-configuration of extranets within MPLS VPN networks. In an extranet configuration, routes present in one VRF is imported into another VRF on the same PE. Normally, the extranet mechanism requires that either the import-rt or the import policy of the extranet VRFs be modified to control import of the prefixes from another VRF. However, with Accept Own feature, the route-reflector can assert that control without the need for any configuration change on the PE. This way, the Accept Own feature provides a centralized mechanism for administering control of route imports between different VRFs.

BGP Accept Own is supported only for VPNv4 and VPNv6 address families in neighbor configuration mode.

Route-Reflector Handling Accept Own Community and RTs

The ACCEPT_OWN community is originated by the InterAS route-reflector (InterAS-RR) using an outbound route-policy. To minimize the propagation of prefixes with the ACCEPT_OWN community attribute, the attribute will be attached on the InterAS-RR using an outbound route-policy towards the originating PE. The InterAs-RR adds the ACCEPT-OWN community and modifies the set of RTs before sending the new Accept Own route to the attached PEs, including the originator, through intervening RRs. The route is modified via route-policy.

Accept Own Configuration Example



In this configuration example:

- PE11 is configured with Customer VRF and Service VRF.
- OSPF is used as the IGP.
- VPNv4 unicast and VPNv6 unicast address families are enabled between the PE and RR neighbors and IPv4 and IPv6 are enabled between PE and CE neighbors.

The Accept Own configuration works as follows:

1. CE1 originates prefix X.
2. Prefix X is installed in customer VRF as (RD1:X).
3. Prefix X is advertised to IntraAS-RR11 as (RD1:X, RT1).
4. IntraAS-RR11 advertises X to InterAS-RR1 as (RD1:X, RT1).
5. InterAS-RR1 attaches RT2 to prefix X on the inbound and ACCEPT_OWN community on the outbound and advertises prefix X to IntraAS-RR31.
6. IntraAS-RR31 advertises X to PE11.
7. PE11 installs X in Service VRF as (RD2:X,RT1, RT2, ACCEPT_OWN).

Remote PE: Handling of Accept Own Routes

Remote PEs (PEs other than the originator PE), performs bestpath calculation among all the comparable routes. The bestpath algorithm has been modified to prefer an Accept Own path over non-Accept Own path. The bestpath comparison occurs immediately before the IGP metric comparison. If the remote PE receives an Accept Own path from route-reflector 1 and a non-Accept Own path from route-reflector 2, and if the paths are otherwise identical, the Accept Own path is preferred. The import operates on the Accept Own path.

BGP DMZ Link Bandwidth for Unequal Cost Recursive Load Balancing

Border Gateway Protocol demilitarized zone (BGP DMZ) Link Bandwidth for Unequal Cost Recursive Load Balancing provides support for unequal cost load balancing for recursive prefixes on local node using BGP DMZ Link Bandwidth. The unequal load balance is achieved by using the **dmz-link-bandwidth** command in BGP Neighbor configuration mode and the **bandwidth** command in Interface configuration mode.

BFD Multihop Support for BGP

Bi-directional Forwarding Detection Multihop (BFD-MH) support is enabled for BGP. BFD Multihop establishes a BFD session between two addresses that may span multiple network hops. Cisco IOS XR Software BFD Multihop is based on RFC 5883. For more information on BFD Multihop, refer *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers* and *Interface and Hardware Component Command Reference for Cisco ASR 9000 Series Routers*.

BGP Multi-Instance and Multi-AS

Multiple BGP instances are supported on the router corresponding to a Autonomous System (AS). Each BGP instance is a separate process running on the same or on a different RP/DRP node. The BGP instances do not share any prefix table between them. No need for a common adj-rib-in (bRIB) as is the case with distributed BGP. The BGP instances do not communicate with each other and do not set up peering with each other. Each individual instance can set up peering with another router independently.

Multi-AS BGP enables configuring each instance of a multi-instance BGP with a different AS number.

Multi-Instance and Multi-AS BGP provides these capabilities:

- Mechanism to consolidate the services provided by multiple routers using a common routing infrastructure into a single IOS-XR router.

- Mechanism to achieve AF isolation by configuring the different AFs in different BGP instances.
- Means to achieve higher session scale by distributing the overall peering sessions between multiple instances.
- Mechanism to achieve higher prefix scale (especially on a RR) by having different instances carrying different BGP tables.
- Improved BGP convergence under certain scenarios.
- All BGP functionalities including NSR are supported for all the instances.
- The load and commit router-level operations can be performed on previously verified or applied configurations.

Restrictions

- The router supports maximum of 4 BGP instances.
- Each BGP instance needs a unique router-id.
- Only one Address Family can be configured under each BGP instance (VPNv4, VPNv6 and RT-Constrain can be configured under multiple BGP instances).
- IPv4/IPv6 Unicast should be within the same BGP instance in which IPv4/IPv6 Labeled-Unicast is configured.
- IPv4/IPv6 Multicast should be within the same BGP instance in which IPv4/IPv6 Unicast is configured.
- All configuration changes for a single BGP instance can be committed together. However, configuration changes for multiple instances cannot be committed together.
- Cisco recommends that BGP update-source should be unique in the default VRF over all instances while peering with the same remote router.

BGP Prefix Origin Validation Based on RPKI

A BGP route associates an address prefix with a set of autonomous systems (AS) that identify the interdomain path the prefix has traversed in the form of BGP announcements. This set is represented as the AS_PATH attribute in BGP and starts with the AS that originated the prefix.

To help reduce well-known threats against BGP including prefix mis-announcing and monkey-in-the-middle attacks, one of the security requirements is the ability to validate the origination AS of BGP routes. The AS number claiming to originate an address prefix (as derived from the AS_PATH attribute of the BGP route) needs to be verified and authorized by the prefix holder.

The Resource Public Key Infrastructure (RPKI) is an approach to build a formally verifiable database of IP addresses and AS numbers as resources. The RPKI is a globally distributed database containing, among other things, information mapping BGP (internet) prefixes to their authorized origin-AS numbers. Routers running BGP can connect to the RPKI to validate the origin-AS of BGP paths.

The BGP RPKI Bind Source feature allows you to specify the source IP address and interface used for the RPKI server connection. This feature enables you to have RPKI session that source from loopback interface, for example.

BGP origin-as validation is enabled by default.

Configuring RPKI Cache-server

Perform this task to configure Resource Public Key Infrastructure (RPKI) cache-server parameters.

Configure the RPKI cache-server parameters in rpki-server configuration mode. Use the **rpki server** command in router BGP configuration mode to enter into the rpki-server configuration mode

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **rpki server** {*host-name* | *ip-address*}
4. **bind-source interface** *name*
5. Use one of these commands:
   - **transport ssh port** *port_number*
   - **transport tcp port** *port_number*
6. (Optional) **username** *user_name*
7. (Optional) **password** *password*
8. **preference** *preference_value*
9. **purge-time** *time*
10. Use one of these commands.
    - **refresh-time** *time*
    - **refresh-time off**
11. Use one these commands.
    - **response-time** *time*
    - **response-time off**
12. Use the **commit** or **end** command.
13. (Optional) **shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)#router bgp 100`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**rpki server** {*host-name* | *ip-address*} **Example:** `RP/0/RSP0/CPU0:router(config-bgp)#rpki server 10.2.3.4`	Enters rpki-server configuration mode and enables configuration of RPKI cache parameters.

	Command or Action	Purpose
Step 4	**bind-source interface** *name* **Example:** `Router#(config-bgp)# bind-source interface Loopback2`	Specifies a Loopback interface as the source interface used for the RPKI server connection.

Step 5	Use one of these commands:
	• **transport ssh port** *port_number*
	• **transport tcp port** *port_number*
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#transport ssh port 22`
	Or
	`RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#transport tcp port 2`
		Specifies a transport method for the RPKI cache.
		• **ssh**—Select **ssh** to connect to the RPKI cache using SSH.
		• **tcp**—Select **tcp** to connect to the RPKI cache using TCP (unencrypted).
		• **port** *port_number*—Specify the port number for the RPKI cache transport over TCP and SSH protocols. The port number ranges from 1 to 65535.
		Note SSH supports custom ports in addition to the default port number 22.
		Note You can set the transport to either TCP or SSH. Change of transport causes the cache session to flap.

Step 6	(Optional) **username** *user_name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#username ssh_rpki_uname`	Specifies a (SSH) username for the RPKI cache-server.
Step 7	(Optional) **password** *password* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#password ssh_rpki_pass`	Specifies a (SSH) password for the RPKI cache-server. **Note** The "username" and "password" configurations only apply if the SSH method of transport is active.
Step 8	**preference** *preference_value* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#preference 1`	Specifies a preference value for the RPKI cache. Range for the preference value is 1 to 10. Setting a lower preference value is better.
Step 9	**purge-time** *time* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#purge-time 30`	Configures the time BGP waits to keep routes from a cache after the cache session drops. Set purge time in seconds. Range for the purge time is 30 to 360 seconds.
Step 10	Use one of these commands. • **refresh-time** *time* • **refresh-time off** **Example:**	Configures the time BGP waits in between sending periodic serial queries to the cache. Set refresh-time in seconds. Range for the refresh time is 15 to 3600 seconds. Configure the **off** option to specify not to send serial-queries periodically.

	Command or Action	Purpose

	`RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#refresh-time 20`
	Or
	`RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#refresh-time off`	

Step 11	Use one these commands.
	• **response-time** *time*
	• **response-time off**
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#response-time 30`
	Or
	`RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#response-time off`
		Configures the time BGP waits for a response after sending a serial or reset query. Set response-time in seconds. Range for the response time is 15 to 3600 seconds.
		Configure the **off** option to wait indefinitely for a response.

Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 13	(Optional) **shutdown** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-rpki-server)#shutdown`	Configures shut down of the RPKI cache.

Configure BGP Prefix Validation

Starting from Release 6.5.1, origin-as validation is disabled by default, you must enable it per address family. From Release 6.5.1, use the following task to configure RPKI Prefix Validation.

Origin-as validation is enabled by default.

```
Router(config)# router bgp 100
/* The bgp origin-as validation time and bgp origin-as validity signal ibgp commands are
optional. */.
Router(config-bgp)# bgp origin-as validation time 50
Router(config-bgp)# bgp origin-as validation time off
Router(config-bgp)# bgp origin-as validation signal ibgp
Router(config-bgp)# address-family ipv4 unicast

!
Router# bgp 65000
Router(config-bgp)# address-family ipv4 unicast
```

```
Router(config-bgp-af)# bgp origin-as validation enable
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp-af)# bgp origin-as validation enable
```

Use the following commands to verify the origin-as validation configuration:

```
Router# show bgp origin-as validity

Thu Mar 14 04:18:09.656 PDT
BGP router identifier 10.1.1.1, local AS number 1
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 514
BGP main routing table version 514
BGP NSR Initial initsync version 2 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Origin-AS validation codes: V valid, I invalid, N not-found, D disabled
    Network                 Next Hop            Metric LocPrf Weight Path
 *> 209.165.200.223/27      0.0.0.0                  0         32768 ?

 *> 209.165.200.225/27      0.0.0.0                  0         32768 ?

 *> 19.1.2.0/24             0.0.0.0                  0         32768 ?

 *> 19.1.3.0/24             0.0.0.0                  0         32768 ?

 *> 10.1.2.0/24             0.0.0.0                  0         32768 ?

 *> 10.1.3.0/24             0.0.0.0                  0         32768 ?

 *> 10.1.4.0/24             0.0.0.0                  0         32768 ?

 *> 198.51.100.1/24         0.0.0.0                  0         32768 ?

 *> 203.0.113.235/24        0.0.0.0                  0         32768 ?

V*> 209.165.201.0/27        10.1.2.1                 0         4002 i

N*> 198.51.100.2/24         10.1.2.1                 0         4002 i

I*> 198.51.100.1/24         10.1.2.1                 0          4002 i

 *> 192.0.2.1.0/24          0.0.0.0                   0         32768 ?
```

```
Router# show bgp process
Mon Jul  9 16:47:39.428 PDT

BGP Process Information:
...
Use origin-AS validity in bestpath decisions
Allow (origin-AS) INVALID paths
Signal origin-AS validity state to neighbors

Address family: IPv4 Unicast
...
Origin-AS validation is enabled for this address-family
Use origin-AS validity in bestpath decisions for this address-family
Allow (origin-AS) INVALID paths for this address-family
Signal origin-AS validity state to neighbors with this address-family
```

Configuring RPKI Bestpath Computation

Perform this task to configure RPKI bestpath computation options.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp bestpath origin-as use validity**
4. **bgp bestpath origin-as allow invalid**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)#router bgp 100`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp bestpath origin-as use validity** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)#bgp bestpath origin-as use validity`	Enables the validity states of BGP paths to affect the path's preference in the BGP best path process. This configuration can also be done in router BGP address family submode.
Step 4	**bgp bestpath origin-as allow invalid** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)#bgp bestpath origin-as allow invalid`	Allows all "invalid" paths to be considered for BGP bestpath computation. **Note** This configuration can also be done at global address family, neighbor, and neighbor address family submodes. Configuring bgp bestpath origin-as allow invalid in router BGP and address family submodes allow all "invalid" paths to be considered for BGP bestpath computation. By default, all such paths are not bestpath candidates. Configuring bgp bestpath origin-as allow invalid in neighbor and neighbor address family submodes allow all "invalid" paths from that specific neighbor or neighbor address family to be considered as bestpath candidates. The neighbor must be an eBGP neighbor. This configuration takes effect only when the **bgp bestpath origin-as use validity** configuration is enabled.

	Command or Action	Purpose
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Prefix Independent Convergence for RIB and FIB

BGP PIC for RIB and FIB adds support for static recursive as PE-CE and faster backup activation by using fast re-route trigger.

The BGP PIC for RIB and FIB feature supports:

- FRR-like trigger for faster PE-CE link down detection, to further reduce the convergence time (Fast PIC-edge activation).
- PIC-edge for static recursive routes.
- BFD single-hop trigger for PIC-Edge without any explicit /32 static route configuration.
- Recursive PIC activation at third level and beyond, on failure trigger at the first (IGP) level.
- BGP path recursion constraints in FIB to ensure that FIB is in sync with BGP with respect to BGP next-hop resolution.

When BGP PIC Edge is configured, configuring the **neighbor shutdown** command does not trigger CEF to switch to the backup path. Instead, BGP starts to feed CEF again one by one from the top prefix of the routing table to the end thus causing a time delay.

Caution The time delay causes a traffic outage in the network. As a workaround, you must route the traffic to the backup path manually before configuring the **neighbor shutdown** command.

Delay BGP Route Advertisements

Table 6: Feature History Table

Feature Name	Release Information	Feature Description

Delay BGP Route Advertisements	Release 7.5.3	You can now prevent traffic loss due to premature advertising of BGP routes and subsequent packet loss in a network. You can achieve this by setting the delay time of the BGP start-up in the router until the Routing Information Base (RIB) is synchronized with the Forward Information Base (FIB) in the routing table. This delays the BGP update generation and prevents traffic loss in a network. You can configure a minimum delay of 1 second and a maximum delay of 600 seconds. This feature introduces the **update wait-install delay startup** command.

When BGP forwards traffic, it waits for feedback from the RIB until the RIB is ready to forward traffic. Once the RIB is ready, BGP sends the route updates to the BGP neighbors and peer-groups. Advertising routes before the RIB is synchronized in the FIB results in traffic loss. To avoid this problem, the router must delay the BGP start-up process to delay the BGP update generation so that no traffic loss happens.

To accomplish this, you must configure the **update wait-install delay startup** command to delay the generation of BGP updates. The **show bgp process** command displays the delay of the BGP process update since the last router reload.

This feature allows you to configure the minimum and maximum delay periods. The range of the delay is from 1 second to 600 seconds. As a result, network traffic loss is avoided.

Restrictions

This feature is applicable for the following Address Family Indicators (AFIs):

- IPv4 unicast
- IPv6 unicast
- VPNv4 unicast
- VPNv6 unicast

Configuration

1. Enter the IOS XR configuration mode.

   ```
   Router# configure
   ```

2. Specify the BGP Autonomous System Number (AS Number).

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

3. Specify the IP address from the address-family (Pv4, IPv6, VPNv4, or VPNv6) options.

   ```
   Router(config-bgp)# address-family {ipv4| ipv6| vpnv4| vpn6} unicast
   ```

   For example,

   ```
   Router(config-bgp)# address-family ipv4 unicast
   ```

4. Schedule the delay of the BGP process to prevent routes from being advertised to peers until RIB is synchronized.

   ```
   Router(config-bgp-af)# update wait-install delay startup (time in seconds)
   ```

For example,

```
Router(config-bgp-af)# update wait-install delay startup 10
```

5. Commit the changes.

```
Router(config-bgp-af)#commit
```

Note The delay time ranges from 1 second to 600 seconds.

Running Configuration

```
configure
router bgp 1
 address-family ipv4 unicast
  update wait-install delay startup 10
!
```

Verification Example

The following command displays the delay of the BGP process update:

```
Router# show running-config router bgp 1
router bgp 1
address-family ipv4 unicast
update wait-install delay startup 10
```

Disable the BGP Fast Reroute Reset Timer

For BGP PIC-edge scenarios where dual-home CE and BFD or BGP are running between PE and CE routers, a BGP Fast Reroute (FRR) reset timer ensures that in case the best path isn't available, traffic is forwarded for four minutes on the backup path to prevent traffic loss. However, when an interface or BFD flap occurs, the BGP FRR may continue forwarding traffic on the backup path even though the primary path is restored. Such a scenario may cause prolonged traffic outages. To prevent such potential outages, run the **cef fast-reroute follow bgp-pic** command to turn off the BGP FRR reset timer.

Note that:

- Before Release 7.3.x, the BGP FRR reset timer is enabled by default, and you must run the **cef fast-reroute follow bgp-pic** command to turn it off.
- From Release 7.3.x, the BGP FRR reset timer is disabled, and the **cef fast-reroute follow bgp-pic** command is deprecated. You can't enable the timer.
- From Release 7.6.x, the **cef fast-reroute follow bgp-pic** command is removed.

BGP Update Message Error Handling

The BGP UPDATE message error handling changes BGP behavior in handling error UPDATE messages to avoid session reset. Based on the approach described in IETF IDR *I-D:draft-ietf-idr-error-handling*, the Cisco IOS XR BGP UPDATE Message Error handling implementation classifies BGP update errors into various categories based on factors such as, severity, likelihood of occurrence of UPDATE errors, or type of attributes. Errors encountered in each category are handled according to the draft. Session reset will be avoided as much

as possible during the error handling process. Error handling for some of the categories are controlled by configuration commands to enable or disable the default behavior.

According to the base BGP specification, a BGP speaker that receives an UPDATE message containing a malformed attribute is required to reset the session over which the offending attribute was received. This behavior is undesirable as a session reset would impact not only routes with the offending attribute, but also other valid routes exchanged over the session.

BGP Attribute Filtering

The BGP Attribute Filter feature checks integrity of BGP updates in BGP update messages and optimizes reaction when detecting invalid attributes. BGP Update message contains a list of mandatory and optional attributes. These attributes in the update message include MED, LOCAL_PREF, COMMUNITY etc. In some cases, if the attributes are malformed, there is a need to filter these attributes at the receiving end of the router. The BGP Attribute Filter functionality filters the attributes received in the incoming update message. The attribute filter can also be used to filter any attributes that may potentially cause undesirable behavior on the receiving router.

Some of the BGP updates are malformed due to wrong formatting of attributes such as the network layer reachability information (NLRI) or other fields in the update message. These malformed updates, when received, causes undesirable behavior on the receiving routers. Such undesirable behavior may be encountered during update message parsing or during re-advertisement of received NLRIs. In such scenarios, its better to filter these corrupted attributes at the receiving end.

BGP Attribute Filter Actions

The Attribute-filtering is configured by specifying a single or a range of attribute codes and an associated action. The allowed actions are:

- " Treat-as-withdraw"— The associated IPv4-unicast or MP_REACH NLRIs, if present, are withdrawn from the neighbor's Adj-RIB-In.
- "Discard Attribute"—The matching attributes alone are discarded and the rest of the Update message is processed normally.

When a received Update message contains one or more filtered attributes, the configured action is applied on the message. Optionally, the Update message is also stored to facilitate further debugging and a syslog message is generated on the console.

When an attribute matches the filter, further processing of the attribute is stopped and the corresponding action is taken.

Use the **attribute-filter group** command to enter Attribute-filter group command mode. Use the **attribute** command in attribute-filter group command mode to either discard an attribute or treat the update message as a "Withdraw" action.

BGP Error Handling and Attribute Filtering Syslog Messages

When a router receives a malformed update packet, an ios_msg of type ROUTING-BGP-3-MALFORM_UPDATE is printed on the console. This is rate limited to 1 message per minute across all neighbors. For malformed packets that result in actions "Discard Attribute" (A5) or "Local Repair" (A6), the ios_msg is printed only once per neighbor per action. This is irrespective of the number of malformed updates received since the neighbor last reached an "Established" state.

This is a sample BGP error handling syslog message:

```
%ROUTING-BGP-3-MALFORM_UPDATE : Malformed UPDATE message received from neighbor 13.0.3.50
- message length 90 bytes,
 error flags 0x00000840, action taken "TreatAsWithdraw".
Error details: "Error 0x00000800, Field "Attr-missing", Attribute 1 (Flags 0x00, Length 0),
 Data []"
```

This is a sample BGP attribute filtering syslog message for the "discard attribute" action:

```
[4843.46]RP/0/0/CPU0:Aug 21 17:06:17.919 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
One or more attributes were filtered from UPDATE message received from neighbor 40.0.101.1
 - message length 173 bytes,
 action taken "DiscardAttr".
 Filtering details: "Attribute 16 (Flags 0xc0): Action "DiscardAttr"". NLRIs: [IPv4 Unicast]
 88.2.0.0/17
```

This is a sample BGP attribute filtering syslog message for the "treat-as-withdraw" action:

```
[391.01]RP/0/0/CPU0:Aug 20 19:41:29.243 : bgp[1037]: %ROUTING-BGP-5-UPDATE_FILTERED :
 One or more attributes were filtered from UPDATE message received from neighbor 40.0.101.1
 - message length 166 bytes,
 action taken "TreatAsWdr".
 Filtering details: "Attribute 4 (Flags 0xc0): Action "TreatAsWdr"". NLRIs: [IPv4 Unicast]
 88.2.0.0/17
```

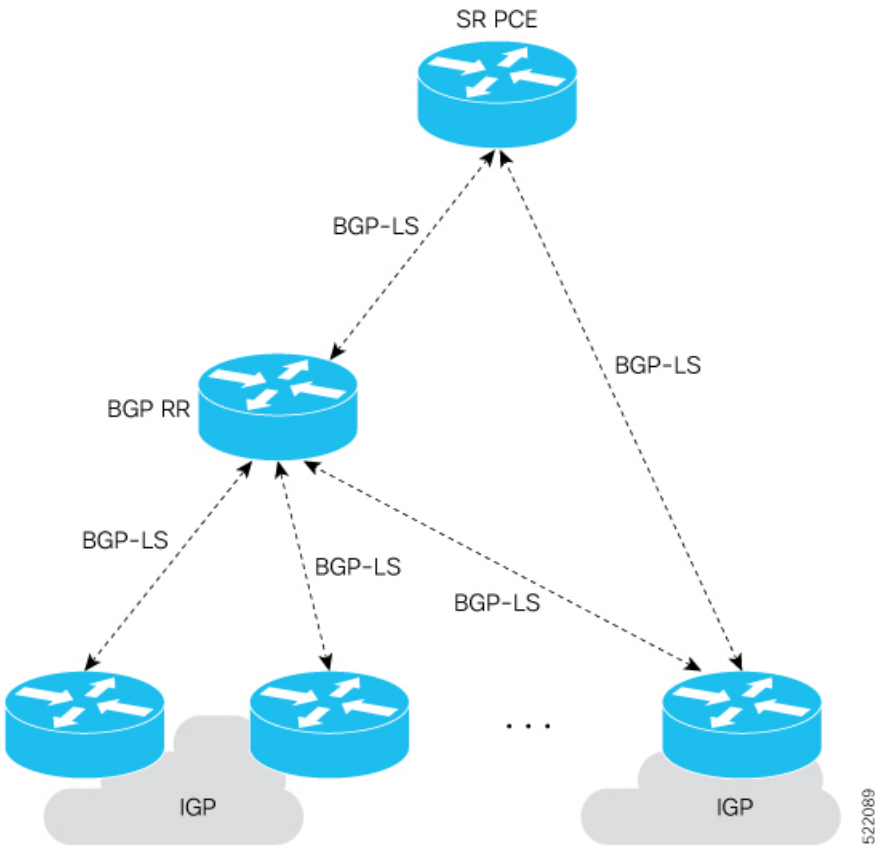
BGP Link-State

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

Note IGPs do not use BGP LS data from remote peers. BGP does not download the received BGP LS data to any other component on the router.

An example of a BGP-LS application is the Segment Routing Path Computation Element (SR-PCE). The SR-PCE can learn the SR capabilities of the nodes in the topology and the mapping of SR segments to those nodes. This can enable the SR-PCE to perform path computations based on SR-TE and to steer traffic on paths different from the underlying IGP-based distributed best-path computation.

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



Exchange Link State Information with BGP Neighbor

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

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

IGP Link-State Database Distribution

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

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

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

Usage Guidelines and Limitations

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

Table 7: IOS XR Supported BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
256	Local Node Descriptors	X	X	—
257	Remote Node Descriptors	X	X	—
258	Link Local/Remote Identifiers	X	X	—
259	IPv4 interface address	X	X	—
260	IPv4 neighbor address	X		
261	IPv6 interface address	X	—	—
262	IPv6 neighbor address	X	—	—
263	Multi-Topology ID	X	—	—
264	OSPF Route Type	—	X	—
265	IP Reachability Information	X	X	—
266	Node MSD TLV	X	X	—
267	Link MSD TLV	X	X	—
512	Autonomous System	—	—	X
513	BGP-LS Identifier	—	—	X

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
514	OSPF Area-ID	—	X	—
515	IGP Router-ID	X	X	—
516	BGP Router-ID TLV	—	—	X
517	BGP Confederation Member TLV	—	—	X
1024	Node Flag Bits	X	X	—
1026	Node Name	X	X	—
1027	IS-IS Area Identifier	X	—	—
1028	IPv4 Router-ID of Local Node	X	X	—
1029	IPv6 Router-ID of Local Node	X	—	—
1030	IPv4 Router-ID of Remote Node	X	X	—
1031	IPv6 Router-ID of Remote Node	X	—	—
1034	SR Capabilities TLV	X	X	—
1035	SR Algorithm TLV	X	X	—
1036	SR Local Block TLV	X	X	—
1039	Flex Algo Definition (FAD) TLV	X	X	—
1044	Flex Algorithm Prefix Metric (FAPM) TLV	X	X	—
1088	Administrative group (color)	X	X	—
1089	Maximum link bandwidth	X	X	—
1090	Max. reservable link bandwidth	X	X	—
1091	Unreserved bandwidth	X	X	—
1092	TE Default Metric	X	X	—
1093	Link Protection Type	X	X	—
1094	MPLS Protocol Mask	X	X	—
1095	IGP Metric	X	X	—
1096	Shared Risk Link Group	X	X	—
1099	Adjacency SID TLV	X	X	—
1100	LAN Adjacency SID TLV	X	X	—
1101	PeerNode SID TLV	—	—	X
1102	PeerAdj SID TLV	—	—	X
1103	PeerSet SID TLV	—	—	X
1114	Unidirectional Link Delay TLV	X	X	—

TLV Code Point	Description	Produced by IS-IS	Produced by OSPFv2	Produced by BGP
1115	Min/Max Unidirectional Link Delay TLV	X	X	—
1116	Unidirectional Delay Variation TLV	X	X	—
1117	Unidirectional Link Loss	X	X	—
1118	Unidirectional Residual Bandwidth	X	X	—
1119	Unidirectional Available Bandwidth	X	X	—
1120	Unidirectional Utilized Bandwidth	X	X	—
1122	Application-Specific Link Attribute TLV	X	X	—
1152	IGP Flags	X	X	—
1153	IGP Route Tag	X	X	—
1154	IGP Extended Route Tag	X	—	—
1155	Prefix Metric	X	X	—
1156	OSPF Forwarding Address	—	X	—
1158	Prefix-SID	X	X	—
1159	Range	X	X	—
1161	SID/Label TLV	X	X	—
1170	Prefix Attribute Flags	X	X	—
1171	Source Router Identifier	X	—	—
1172	L2 Bundle Member Attributes TLV	X	—	—
1173	Extended Administrative Group	X	X	—

BGP Permanent Network

BGP permanent network feature supports static routing through BGP. BGP routes to IPv4 or IPv6 destinations (identified by a route-policy) can be administratively created and selectively advertised to BGP peers. These routes remain in the routing table until they are administratively removed.

A permanent network is used to define a set of prefixes as permanent, that is, there is only one BGP advertisement or withdrawal in upstream for a set of prefixes. For each network in the prefix-set, a BGP permanent path is created and treated as less preferred than the other BGP paths received from its peer. The BGP permanent path is downloaded into RIB when it is the best-path.

The **permanent-network** command in global address family configuration mode uses a route-policy to identify the set of prefixes (networks) for which permanent paths is to be configured. The **advertise permanent-network** command in neighbor address-family configuration mode is used to identify the peers to whom the permanent paths must be advertised. The permanent paths is always advertised to peers having the advertise permanent-network configuration, even if a different best-path is available. The permanent path is not advertised to peers that are not configured to receive permanent path.

The permanent network feature supports only prefixes in IPv4 unicast and IPv6 unicast address-families under the default Virtual Routing and Forwarding (VRF).

Restrictions

These restrictions apply while configuring the permanent network:

- Permanent network prefixes must be specified by the route-policy on the global address family.
- You must configure the permanent network with route-policy in global address family configuration mode and then configure it on the neighbor address family configuration mode.
- When removing the permanent network configuration, remove the configuration in the neighbor address family configuration mode and then remove it from the global address family configuration mode.

BGP-RIB Feedback Mechanism for Update Generation

The Border Gateway Protocol-Routing Information Base (BGP-RIB) feedback mechanism for update generation feature avoids premature route advertisements and subsequent packet loss in a network. This mechanism ensures that routes are installed locally, before they are advertised to a neighbor.

BGP waits for feedback from RIB indicating that the routes that BGP installed in RIB are installed in forwarding information base (FIB) before BGP sends out updates to the neighbors. RIB uses the the BCDL feedback mechanism to determine which version of the routes have been consumed by FIB, and updates the BGP with that version. BGP will send out updates of only those routes that have versions up to the version that FIB has installed. This selective update ensures that BGP does not send out premature updates resulting in attracting traffic even before the data plane is programmed after router reload, LC OIR, or flap of a link where an alternate path is made available.

To configure BGP to wait for feedback from RIB indicating that the routes that BGP installed in RIB are installed in FIB, before BGP sends out updates to neighbors, use the **update wait-install** command in router address-family IPv4 or router address-family VPNv4 configuration mode. The **show bgp**, **show bgp neighbors**, and **show bgp process performance-statistics** commands display the information from update wait-install configuration.

BGP VRF Dynamic Route Leaking

The Border Gateway Protocol (BGP) dynamic route leaking feature provides the ability to import routes between the default-vrf (Global VRF) and any other non-default VRF, to provide connectivity between a global and a VPN host. The import process installs the Internet route in a VRF table or a VRF route in the Internet table, providing connectivity.

Note A leaked route should not cover or override any routes in the destination VRF. For example consider two connected routers R1 with destination VRF 'dest-vrf' and R2 with source VRF 'source-vrf'. The source-vrf connected route CR-1 is leaked to dest-vrf. In this case, the route from dest-vrf is covered or overridden by the leaked route CR-1 from the source-vrf.

The dynamic route leaking is enabled by:

- Importing from default-VRF to non-default-VRF, using the **import from default-vrf route-policy** *route-policy-name* [**advertise-as-vpn**] command in VRF address-family configuration mode.

If the **advertise-as-vpn** option is configured, the paths imported from the default-VRF to the non-default-VRF are advertised to the PEs as well as to the CEs. If the **advertise-as-vpn** option is not configured, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PE. However, the paths are still advertised to the CEs.

- Importing from non-default-VRF to default VRF, using the **export to default-vrf route-policy** *route-policy-name* command in VRF address-family configuration mode.

A route-policy is mandatory to filter the imported routes. This reduces the risk of unintended import of routes between the Internet table and the VRF tables and the corresponding security issues.

There is no hard limit on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths. However, each VRF importing global routes adds workload equivalent to a neighbor receiving the global table. This is true even if the user filters out all but a few prefixes. Hence, importing five to ten VRFs is ideal.

Note With dynamic route-leaking enabled, BGP bestpath change suppression for eBGP paths might be skipped. BGP convergence might be impacted.

EVPN Default VRF Route Leaking

The EVPN Default VRF Route Leaking feature leak routes between EVPN address-family and IPv4/IPv6 unicast address-family (Default-VRF), enabling the data center hosts to access the Internet. This feature is an extension of Border Gateway Protocol (BGP) VRF Dynamic route leaking feature that provides connectivity between non-default VRF hosts and Default VRF hosts by exchanging routes between the non-default VRF and Default VRF. EVPN Default VRF Route Leaking feature extends the BGP VRF Dynamic leaking feature, by allowing EVPN/L3VPN hosts to communicate with Default VRF hosts.

The import process installs the Internet route in a VRF table or a VRF route in the Internet table, providing connectivity.

The BGP VRF Dynamic route leaking feature is enabled by:

- Importing from default-VRF to non-default-VRF using the following command in VRF address-family configuration mode.

  import from default-vrf route-policy *route-policy-name* **[advertise-as-vpn]**

  If the **advertise-as-vpn** keyword is used, the paths imported from the default-VRF to the non-default-VRF are advertised to the (EVPN/L3VPN) PEs as well as to the CEs. If the **advertise-as-vpn** keyword is not used, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PEs. However, the paths are still advertised to the CEs.

  The EVPN Default VRF Route Leaking feature with **advertise-as-vpn** keyword, enables to advertise the paths imported from default-VRF to non-default VRFs to EVPN PE peers as well.

  A new command **advertise vpnv4/vpnv6 unicast imported-from-default-vrf disable** is added under neighbor address-family configuration mode for EVPN and VPNv4/VPNv6 unicast to disable advertisement of Default-VRF leaked routes to that neighbor.

- Importing from non-default-VRF to default-VRF using the following command in VRF address-family configuration mode.

  export to default-vrf route-policy *route-policy-name* **[advertise-as-vpn]**

The Dynamic Route Leaking feature enables leaking of local and CE routes to Default-VRF.

A new optional keyword **allow-imported-vpn** is added to the above command, when configured, enables the leaking of EVPN and L3VPN imported/re-originated routes to the Default-VRF.

A route-policy is mandatory to filter the imported routes. This reduces the risk of unintended import of routes between the Internet table and the VRF tables and the corresponding security issues. There is no hard limit on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths.

Note Each VRF importing global routes adds workload equivalent to a neighbor receiving the global table. This is true even if the user filters out all but a few prefixes.

Scale Limitation of Default Route Leaking

Default VRF route leaking uses Dynamic Route Leaking feature to leak prefixes between the default VRF and the DC VRF. Do not use Dynamic Route Leaking feature to leak default VRF prefixes to large number of DC VRFs, even if you filter out all prefixes except a few that are to be leaked.

The following are the key factors that affect the performance:

- The default VRF prefix scale, which is approximately 0.7 million internet prefixes.
- The number of DC VRFs the default VRF prefixes that are to be imported.

To improve the scale, either the prefix scale or the number of VRFs whose prefixes that are to be imported must be reduced.

To manage the scale limitation, Cisco recommends you to do the following:

- Host the Internet prefixes on an adjacent PE with IPv4 unicast peering with DCI, and advertise a default route towards the DCI. On the DCI, import the default route from default VRF to DC VRFs.
- Host the Internet prefixes on an adjacent PE with IPv4 unicast peering with DCI. On the DCI, configure a static default route in the DC VRF with the next hop of the default VRF pointing to the adjacent PE address.
- Configure the static default route 0.0.0.0/0 on DC VRF with nexthop as “vrf default”.

Note If the static routes are re-distributed to BGP, make sure it is not unintentionally advertised out.

EVPN Default VRF Route Leaking on the DCI for Internet Connectivity

The EVPN Default VRF Route Leaking feature leak routes between the Default-VRF and Data Center-VRF on the DCI to provide Internet access to data center hosts.

This feature is enabled by:

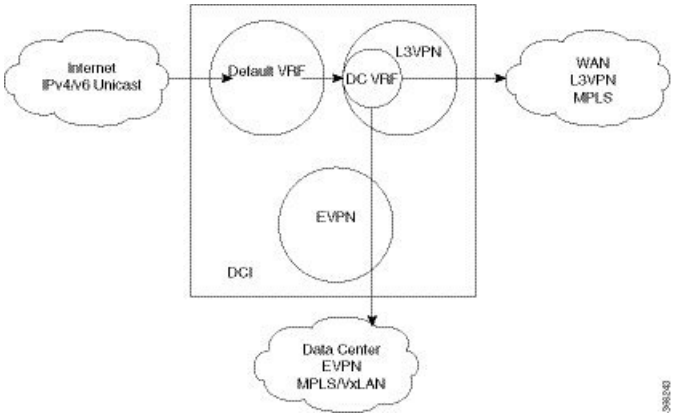
- Leaking routes from Default-VRF to Data Center-VRF

- Leaking routes to Default-VRF from Data Center-VRF

Leaking Routes from Default-VRF to Data Center-VRF

This section explains the process of leaking Default-VRF routes to Data Center-VRF.

Figure 12: Leaking Routes from Default-VRF to Data Center-VRF



Step 1 The Internet routes are present in the Default-VRF on the DCI.

Note A static default-route (0/0) can be configured under Default-VRF router static address-family configuration and redistributed to BGP.

Step 2 A route-policy is configured to select the routes to be leaked from Default-VRF to Data Center-VRF.

Example:

```
route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!
```

Note Instead of leaking the internet routes, you can leak the default-route 0/0 from Default-VRF to Data Center-VRF using the following policy.

```
route-policy import-from-default-policy
  if destination in (0.0.0.0/0) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (0::0/0) then
    pass
  endif
end-policy
!
```

Step 3 Leak Default-VRF routes specified in the route-policy to Data Center-VRF by configuring **import from default-vrf route-policy import-from-default-policy(-v6)** under Data Center VRF address-family configuration mode.

Example:

```
vrf data-center-vrf
 address-family ipv4 unicast
  import from default-vrf route-policy import-from-default-policy
!
address-family ipv6 unicast
  import from default-vrf route-policy import-from-default-policy-v6
!
```

Step 4 Advertise the leaked (Default-VRF) routes in the Data Center-VRF as EVPN routes towards Data Center routers by configuring **advertise-as-vpn** option.

Example:

```
vrf data-center-vrf
 address-family ipv4 unicast
  import from default-vrf route-policy import-from-default-policy advertise-as-vpn
!
address-family ipv6 unicast
  import from default-vrf route-policy import-from-default-policy-v6 advertise-as-vpn
!
```

Note To advertise any routes from L3VPN address-family to EVPN peers, use **advertise vpnv4/vpnv6 unicast re-originated [stitching-rt]** command under neighbor address-family L2VPN EVPN.

EVPN Default-originate

Instead of advertising the Default-VRF routes towards Data Center routers, default-originate can be configured under the EVPN neighbor address-family to advertise the default route. When default-originate is configured under the neighbor address-family for EVPN/L3VPN, there is no need to advertise the Default-VRF leaked routes to the data center and **advertise-as-vpn** need not be configured.

Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
```

```
     default-originate

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate
 !
  address-family ipv6 unicast
    allow vpn default-originate
```

Step 5 To block advertisement of the Default-VRF leaked routes towards a particular EVPN/L3VPN peer, use **advertise vpnv4/vpnv6 unicast imported-from-default-vrf disable** command under respective neighbor address-family.

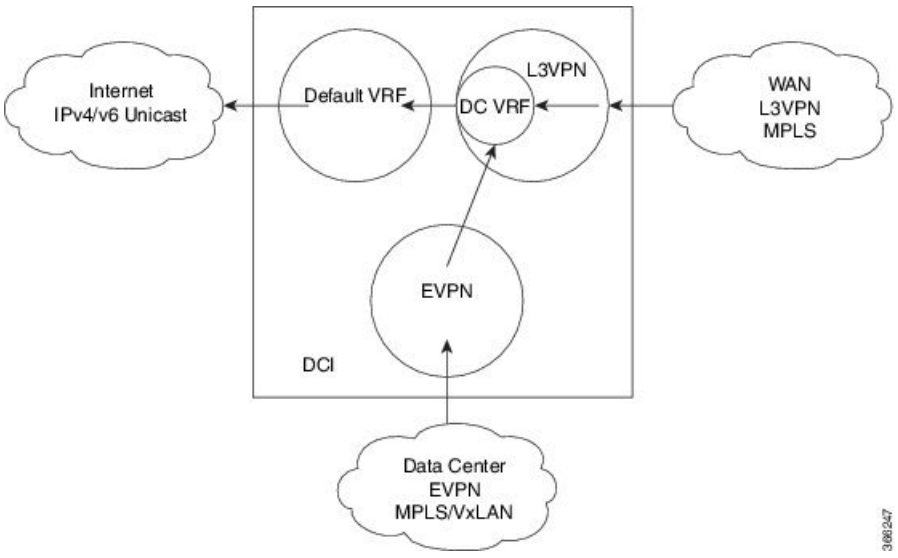
Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      advertise vpnv4 unicast imported-from-default-vrf disable
      advertise vpnv6 unicast imported-from-default-vrf disable
!
router bgp 100
  neighbor 60.0.0.1
    address-family vpnv4 unicast
      advertise vpnv4 unicast imported-from-default-vrf disable
    address-family vpnv6 unicast
      advertise vpnv6 unicast imported-from-default-vrf disable
```

Leaking Routes to Default-VRF from Data Center-VRF

This section explains the process of leaking Data Center-VRF routes to Default-VRF.

Figure 13: Leaking Routes to Default-VRF from Data Center-VRF



Step 1 Data Center routes are received on the DCI as EVPN Route-type 2 and Route-type 5 NLRI and imported to the Data Center VRFs.

Step 2 A route-policy is configured to select the routes to be leaked from Data Center-VRF to Default-VRF.

Example:

```
route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!
```

Step 3 Leak Data Center-VRF routes specified in the above policy to Default-VRF by configuring **export to default-vrf route-policy export-to-default-policy(-v6) [allow-imported-vpn]** under Data Center-VRF address-family configuration mode.

Normally only local and CE VRF routes are allowed to be leaked to the Default-VRF, but **allow-imported-vpn** configuration enables leaking of EVPN/L3VPN imported routes to the Default-VRF.

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    export to default-vrf route-policy export-to-default-policy [allow-imported-vpn]
  !
  address-family ipv6 unicast
    export to default-vrf route-policy export-to-default-policy-v6 [allow-imported-vpn]
  !
```

Step 4 The Leaked routes in the Default VRF are advertised to the Internet.

Note Instead of advertising the leaked routes to the Internet, an aggregate can be configured and advertised to the Internet.

Sample Router Configuration

The following sample configuration specifies how EVPN Default VRF Route Leaking feature is configured on a DCI router to provide Internet access to the data center hosts.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from default-vrf route-policy import-from-default-policy advertise-as-vpn
    export to default-vrf route-policy export-to-default-policy allow-imported-vpn
  !
```

```
  address-family ipv6 unicast
    import from default-vrf route-policy import-from-default-policy-v6 advertise-as-vpn
    export to default-vrf route-policy export-to-default-policy-v6 allow-imported-vpn
  !

route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt

  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-default-vrf disable

    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-default-vrf disable
```

Sample Router Configuration: with default-originate

The following sample configuration specifies how EVPN Default VRF Route Leaking feature is configured along with default-originate on a DCI router to provide Internet access to data center hosts.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from default-vrf route-policy import-from-default-policy <= Remove
advertise-as-vpn=>
    export to default-vrf route-policy export-to-default-policy allow-imported-vpn
  !
  address-family ipv6 unicast
```

```
    import from default-vrf route-policy import-from-default-policy-v6 <= Remove
advertise-as-vpn=>
    export to default-vrf route-policy export-to-default-policy-v6 allow-imported-vpn
  !
route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!
route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!
route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!
route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt
      default-originate <= Added=>

  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-default-vrf disable

    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-default-vrf disable

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate <= Added=>
  !
  address-family ipv6 unicast
    allow vpn default-originate <= Added=>
```

EVPN Service VRF Route Leaking

The EVPN Service VRF Route Leaking feature enables connectivity to the services in the Service VRF to customers in EVPN Data Center VRF. The Service VRF and Data Center VRF routes can be IPv4 and/or IPv6 addresses. The Services VRF is any L3 VRF providing services reachable through connected, static, re-distributed IGP or BGP routes.

This feature leaks routes between Data Center VRF and Service VRF, enabling the EVPN/L3VPN hosts to access the Services in the Service VRF. This feature rely on Border Gateway Protocol (BGP) VRF extranet feature that imports routes between two VRFs.

The import process installs the Data Center VRF routes in a Service VRF table or a Service VRF routes in the Data Center VRF table, providing connectivity.

The BGP Service VRF route leaking feature is enabled by:

- Importing routes from Service VRF to Data Center VRF and advertising it as EVPN/L3VPN route from Data Center VRF.
  - Importing Service VRF routes to Data Center VRF by attaching Data Center VRF import RTs to Service VRF routes.

    This can be achieved by configuring one or more Data Center VRF import RTs as export RT of Service VRF, or configuring a Service VRF export route-policy to attach import RT EXTCOMM to Service VRF routes matching the import RTs of Data Center VRF using the following command in Service VRF address-family configuration mode.

    export route-policy service-vrf-export-route-policy-name

    Where the route-policy "service-vrf-export-route-policy-name" attaches the RT EXTCOMM matching the one or more import RTs of Data Center VRF to Service VRF routes.
  - Advertising Data Center VRF imported routes that are exported from Service VRFs as EVPN/L3VPN NLRI from Data Center VRF using the following command in Data Center VRF address-family configuration mode.

    import from vrf advertise-as-vpn

    If the **advertise-as-vpn** keyword is used, the paths imported from the Service VRF to the Data Center VRF are advertised to the (EVPN/L3VPN) PEs as well as to the CEs. If the **advertise-as-vpn** keyword is not used, the paths imported from the Service VRF to the Data Center VRF are not advertised to the PEs. However, the paths are still advertised to the CEs.
  - Block advertising Data Center VRF leaked routes from being advertised to a neighbor using the following command in neighbor address-family configuration mode.

    advertise vpnv4/vpnv6 unicast imported-from-vrf disable

    A new command **advertise vpnv4/vpnv6 unicast imported-from-vrf disable** is added under neighbor address-family configuration mode for EVPN and VPNv4/VPNv6 unicast to disable advertisement of VRF to VRF leaked routes to that neighbor.

- Importing EVPN/L3VPN routes from Data Center VRF to Service VRF
  - Importing EVPN/L3VPN routes from Data Center VRF to Service VRF by attaching Service VRF import RTs.

    This can be achieved by configuring one or more Service VRF import RTs as export RT of Data Center VRF, or configuring a Data Center VRF export route-policy to attach import RT EXTCOMM to Data Center VRF routes matching the import RTs of Service VRF using the following command in Data Center VRF address-family configuration mode.

    export route-policy data-center-vrf-export-route-policy-name

    The route-policy "data-center-vrf-export-route-policy-name" attaches the RT EXTCOMM matching one or more import RTs of Service VRF.

- Allow leaking of Data Center VRF routes to Service VRF by using the following command in Data Center VRF address-family configuration mode.

  export to vrf allow-imported-vpn

Note In order to prevent un-intended import of routes to VRFs, select unique RT's to import routes between Service VRF and Data Center VRF, which are not used for normal import of VPN/EVPN routes to Data Center VRFs.

The Extranet Route Leaking feature enables leaking of local and CE routes from one VRF to another VRF. A new command **export to vrf allow-imported-vpn** is added to enable the leaking of EVPN and L3VPN imported/re-originated Data Center VRF routes to the Service VRF.

Note A route-policy is preferred to filter the imported routes. This reduces the risk of unintended import of routes between the Data Center VRF and the Service VRF, and the corresponding security issues. There is no hard limit on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths.

Note This feature does not advertise EVPN/L3VPN PE routes imported to Data Center VRF and leaked to Service VRF as EVPN/L3VPN PE route.

EVPN Service VRF Route Leaking on the DCI for Service Connectivity

The EVPN Service VRF Route Leaking feature leaks routes between the Service VRF and Data Center VRF on the DCI to provide access to Services to data center hosts.

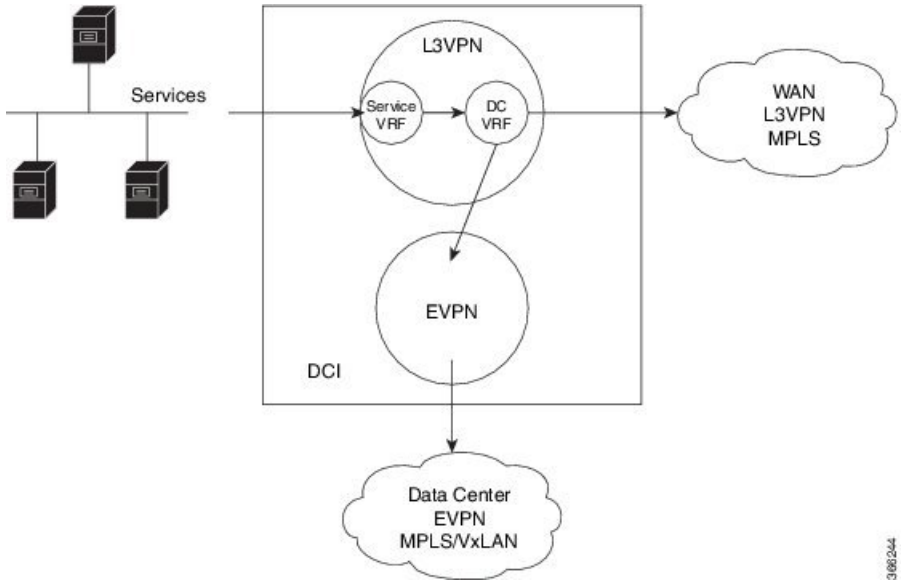
This feature is enabled by:

- Leaking routes from Service VRF to Data Center VRF
- Leaking routes to Service VRF from Data Center VRF

Leaking Routes from Service VRF to Data Center VRF

This section explains the process of leaking Service VRF routes to Data Center VRF.

Figure 14: Leaking Routes from Service VRF to Data Center VRF



Step 1 The Service routes are present in the Service VRF on the DCI.

Step 2 A route-policy is configured to select the routes to be leaked from Service VRF to Data Center VRF.

Example:

```
route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive <--- matches import RT of Data Center-VRF
  endif
end-policy
!
route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive <--- matches import RT of Data Center-VRF
  endif
end-policy
!
```

Step 3 Leak Service VRF routes specified in the route-policy to Data Center VRF by configuring **export route-policy service-vrf-export-policy(-v6)** under Service VRF address-family configuration mode.

Example:

```
vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
```

```
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !
```

Step 4 Advertise the leaked (Service VRF) routes in the Data Center VRF as EVPN/L3VPN routes towards Data Center routers by configuring **import from vrf advertise-as-vpn** under Data Center VRF address-family configuration mode..

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-target
      100:1
      200:1 stitching
  !
```

Note To advertise any routes from L3VPN address-family to EVPN peers, use **advertise vpnv4/vpnv6 unicast re-originated [stitching-rt]** command under neighbor address-family L2VPN EVPN.

EVPN Default-originate

Instead of advertising the Service VRF routes towards Data Center routers, default-originate can be configured under the EVPN neighbor address-family to advertise the default route. When **allow vpn default-originate** is configured under the Data Center VRF, there is no need to advertise the Service VRF leaked routes to the data center and **advertise-as-vpn** need not be configured.

Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      default-originate

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate
 !
  address-family ipv6 unicast
    allow vpn default-originate
```

Step 5 To block advertisement of the Service VRF leaked routes towards a particular EVPN/L3VPN peer, use **advertise vpnv4/vpnv6 unicast imported-from-vrf disable** command under respective neighbor address-family.

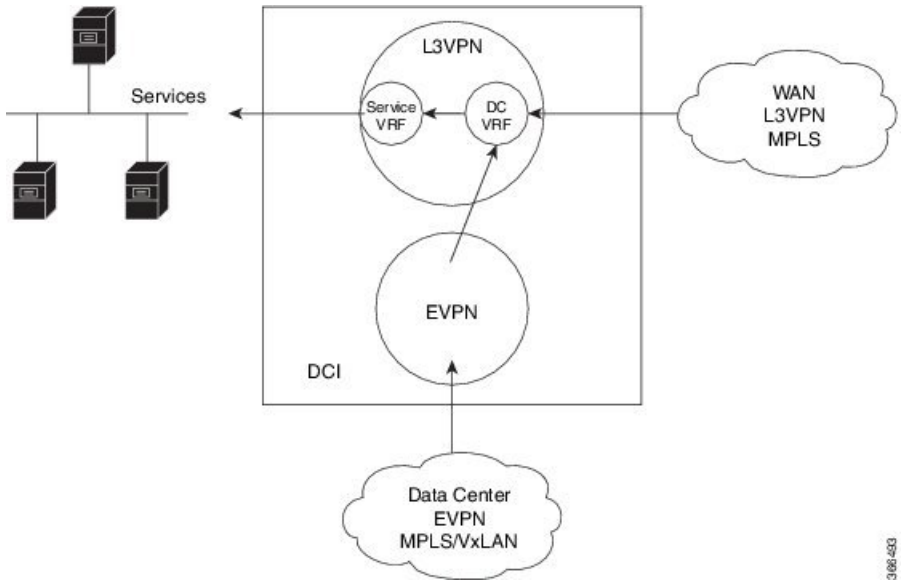
Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv4 unicast imported-from-vrf disable
      advertise vpnv6 unicast re-originated stitching-rt
      advertise vpnv6 unicast imported-from-vrf disable
    !
router bgp 100
  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
```

Leaking Routes to Service VRF from Data Center VRF

This section explains the process of leaking Data Center VRF routes to Service VRF.

Figure 15: Leaking Routes to Service VRF from Data Center VRF



Step 1 Data Center routes are received on the DCI as EVPN Route-type 2 and Route-type 5 NLRI and imported to the Data Center VRFs.

Step 2 A route-policy is configured to select the routes to be leaked from Data Center VRF to Service VRF.

The policy attaches RT EXTCOMM to Data Center VRF routes matching one or more import RT of the Service VRF.

Example:

```
route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then <--- EVPN PE route
    set extcommunity rt (4:1) additive <--- matches import stitching-RT of service-VRF
  if destination in (200.168.0.0/16) then <--- VPNv4 PE route
    set extcommunity rt (3:1) additive <--- matches import RT of service-VRF
  endif
end-policy
!
route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then <--- EVPN PE route
    set extcommunity rt (4:1) additive <--- matches import stitching-RT of service-VRF
  elseif destination in (200:168::0/64) then <--- VPNv6 PE route
    set extcommunity rt (3:1) additive <--- matches import RT of service-VRF
  endif
end-policy
!
```

Note An EVPN/L3VPN route received from a neighbor configured locally with "import stitching-rt re-originate" is imported to Data Center VRF if the route's RT EXTCOMM matches with one or more Data Center VRF import stitching RTs, and is leaked to Service VRF if the Data Center VRF route's RT EXTCOMM matches with one or more Service VRF import stitching RTs.

Step 3 Leak Data Center VRF routes specified in the above policy to Service VRF by configuring **export route-policy data-center-vrf-export-policy(-v6)** under Data Center VRF address-family configuration mode.

Normally only local and CE VRF routes are allowed to be leaked to the Service VRF, but **allow-imported-vpn** configuration enables leaking of EVPN/L3VPN imported routes to the Service VRF.

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
```

Step 4 The Data Center VRF leaked routes in the Service VRF are advertised to Service VRF CE peers.

Sample Router Configuration

The following sample configuration specifies how EVPN Service VRF Route Leaking feature is configured on a DCI router providing access to data center hosts to Services in the Service VRF.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !

vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !

route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then
    set extcommunity rt (4:1) additive
  if destination in (200.168.0.0/16)
    set extcommunity rt (3:1) additive
  endif
end-policy
!
```

```
route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then
    set extcommunity rt (4:1) additive
  elseif destination in (200:168::0/64)
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    remote-as 100
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt
    !
  neighbor 60.0.0.1
    remote-as 200
    address-family vpnv4 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
    address-family vpnv6 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
```

Sample Router Configuration: with default-originate

The following sample configuration specifies how EVPN Service VRF Route Leaking feature is configured along with default-originate on a DCI router to provide data center hosts access to Services in the Service VRF..

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
```

```
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !

vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !

route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then
    set extcommunity rt (4:1) additive
  if destination in (200.168.0.0/16) then
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then
    set extcommunity rt (4:1) additive
  elseif destination in (200:168::0/64) then
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive
  endif
end-policy
```

```
!

route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    remote-as 100
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv4 unicast imported-from-vrf disable
      advertise vpnv6 unicast re-originated stitching-rt
      advertise vpnv6 unicast imported-from-vrf disable
      default-originate <= Added=>
    !
  neighbor 60.0.0.1
    remote-as 200
    address-family vpnv4 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
      default-originate <= Added=>
    address-family vpnv6 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
      default-originate <= Added=>

  vrf data-center-vrf
    rd auto
    address-family ipv4 unicast
      allow vpn default-originate <= Added=>
    !
    address-family ipv6 unicast
      allow vpn default-originate <= Added=>
```

User Defined Martian Check

The Cisco IOS XR Software Release 5.1.0 allows disabling the Martian check for these IP address prefixes:

- IPv4 address prefixes
  - 0.0.0.0/8
  - 127.0.0.0/8
  - 224.0.0.0/4

- IPv6 address prefixes
  - ::
  - ::0002 - ::ffff
  - ::ffff:a.b.c.d
  - fe80:xxxx
  - ffxx:xxxx

Resilient Per-CE Label Mode

The Resilient Per-CE Label is an extension of the Per-CE label mode to support Prefix Independent Convergence (PIC) and load balancing.

There are three label modes that are supported. They are:

- Per-Prefix
  - Label consumption: Large number of labels are required for the MPLS forwarding table on core routers. You need to ensure that the VRF table size is within the MPLS label table limits.
  - Forwarding performance of ASR 9000 Provider Edge router on MPLS to IP path: Optimal. The MPLS label is directly associated with an output interface, hence packets are forwarded by a single pass through the NP microcode.
- Per-VRF
  - Label consumption: Less labels are required for the MPLS forwarding table on core routers. This is because the Provider Edge router advertises one label per entire VRF.
  - Forwarding performance of ASR 9000 Provider Edge router on MPLS to IP path: Sub-optimal. Aggregate label requires two passes through the NP microcode: 1st pass for MPLS lookup, 2nd pass for IP lookup.
- Per-CE
  - Label consumption: Only moderate number of labels are required. This is because the Provider Edge router only advertises one label per CE per VRF.
  - Forwarding performance of ASR 9000 Provider Edge router on MPLS to IP path: Optimal. The MPLS label is directly associated with an output interface, hence packets are forwarded by a single pass through the NP microcode.

At present, the three label modes, Per-Prefix, Per-CE, and Per-VRF have these restrictions:

- No support for PIC
- No support for load balancing across CEs
- Temporary forwarding loop during local traffic diversion to support PIC
- No support for EIBGP multipath load balancing
- Forwarding performance impact

- Per-prefix label mode causes scale issues on another vendor router in a network

In the Resilient Per-CE label scheme, BGP installs a unique rewrite label in LSD for every unique set of CE paths or next hops. There may be one or more prefixes in BGP table that points to this label. BGP also installs the CE paths (primary) and optionally a backup PE path into RIB. FIB learns about the label rewrite information from LSD and the IP paths from RIB.

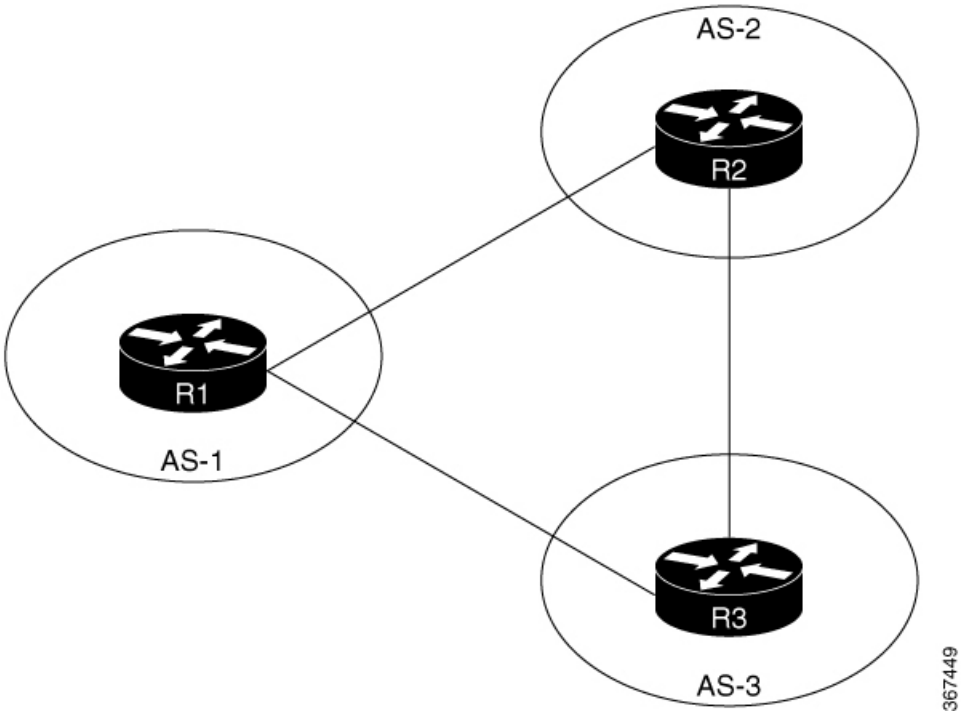
In steady state, labeled traffic destined to the resilient per-CE label is load balanced across all the CE next hops. When all the CE paths fail, any traffic destined to that label will result in an IP lookup and will be forwarded towards the backup PE path, if available. This action is performed on the label independently of the number of prefixes that may point to the label, resulting in the PIC behavior during primary paths failure.

BGP Multipath Enhancements

- Overwriting of next-hop calculation for multipath prefixes is not allowed. The **next-hop-unchanged multipath** command disables overwriting of next-hop calculation for multipath prefixes.
- The ability to ignore as-path onwards while computing multipath is added. The **bgp multipath as-path ignore onwards** command ignores as-path onwards while computing multipath.

When multiple connected routers start ignoring as-path onwards while computing multipath, it causes routing loops. Therefore, you should not configure the **bgp multipath as-path ignore onwards** command on routers that can form a loop.

Figure 16: Topology to illustrate formation of loops



Consider three routers R1, R2 and R3 in different autonomous systems (AS-1, AS-2, and AS-3). The routers are connected with each other. R1 announces a prefix to R2 and R3. Both R2 and R3 are configured with multipath and also with bgp multipath as-path ignore onwards command. Since R3 is configured as multipath,

R2 will send part of its traffic to R3. Similarly, R3 will send part of its traffic to R2. This creates a forwarding loop between R3 and R2. Therefore, to avoid such forwarding loops you should not configure the **bgp multipath as-path ignore onwards** command on connected routers.

MVPN with BGP SAFI-2 and SAFI-129

BGP supports Subsequent Address Family Identifier (SAFI)-2 and SAFI-129 for multicast VPNs (MVPNs).

SAFI-129 provides the capability to support multicast routing in the core IPv4 network. SAFI-129 supports BGP-based MVPNs. The addition of SAFI-129 allows multicast to select an upstream multicast hop that may be independent of the unicast topology. Multicast routes learned from the customer edge (CE) router or multicast VPN routes learned from remote provider edge (PE) routers are installed into the multicast Routing Information Base (MuRIB). This MuRIB will be populated with routes that are specific to multicast, and are not used by unicast forwarding. The PE-CE BGP prefixes are advertised using SAFI-2, the PE-PE routes are advertised using SAFI-129.

Overview of BGP Monitoring Protocol

The BGP Monitoring Protocol (BMP) feature enables monitoring of BGP speakers (called BMP clients). You can configure a device to function as a BMP server, which monitors either one or several BMP clients, which in turn, has several active peer sessions configured. You can also configure a BMP client to connect to one or more BMP servers. The BMP feature enables configuration of multiple BMP servers (configured as primary servers) to function actively and independent of each other, simultaneously to monitor BMP clients.

The BMP Protocol provides access to the Adjacent Routing Information Base, Incoming (Adj-RIB-In) table of a peer on an ongoing basis and a periodic dump of certain statistics that the monitoring station can use for further analysis. The BMP provides pre-policy view of the Adj-RIB-In table of a peer.

There can be several BMP servers configured globally across all the BGP instances. The BMP severs configured are common across multiple speaker instances and each BGP peer in an instance can be configured for monitoring by all or a subset of the BMP servers, giving a 'any-to-any' map between BGP peers and BMP servers from the point of view of a BGP speaker. If a BMP server is configured before any of the BGP peers come up, then the monitoring will start as soon as the BGP peers come up. A BMP server configuration can be removed only when there are no BGP peers configured to be monitored by that particular BMP server.

Sessions between BMP clients and BMP servers operate over plain TCP (no encryption/encapsulation). If a TCP session with the BMP server is not established, the client retries to connect every 7 seconds.

The BMP server does not send any messages to its clients (BGP speakers). The message flow is in one direction only—from BGP speakers to the BMP servers

A maximum of eight BMP servers can be configured on the Cisco NCS 5500 Series Routers. Each BMP server is specified by a server ID and certain parameters such as IP address, port number, etc are configurable. Upon successful configuration of a BMP server with host and port details, the BGP speaker attempts to connect to BMP Server. Once the TCP connection is setup, an Initiation message is sent as first message.

The **bmp server** command enables the user to configure multiple—independent and asynchronous—BMP server connections.

All neighbors for a BGP speaker need not necessarily be BMP clients. BMP clients are the ones that have direct TCP connection with a BMP server. Each of these BGP speakers can have many BGP neighbors or peers. Under a BGP speaker, if any of its neighbors are configured for BMP monitoring, only that particular peer router's messages are sent to BMP servers.

The session connection to BMP server is attempted after an initial-delay at the BMP client. This initial-delay can be configured. If the initial-delay is not configured, then the default connection delay of 7 seconds is used. Configuring the initial delay becomes significant under certain circumstances where, if multiple BMP servers' states toggle closely and refresh delay is so small, then this might result in redundant route-refreshes being generated. This causes considerable network traffic and load on the device. Having different initial delays can reduce the load spike on the network and router.

After the initial delay, TCP connection to BMP servers are attempted. Once the server connections are up, it is checked if there are any peers enabled for monitoring. Once a BGP peer that is already being monitored is in the “ESTAB” state, speaker sends a “peer-up” message for that peer to the BMP server. After the BGP peer receives a route-refresh request, neighbor sends the updates. This route refresh is initiated based on a delay configured for each BMP server. This is called route refresh delay. When there are multiple neighbors to be monitored, each neighbor is set a refresh delay based upon the BMP server they are enabled for. Once all the BGP neighbors have sent the updates in response to the refresh requests, the tables will be up to date in the BMP Server. If a neighbor establishes connection after BMP monitoring has begun, it does not require a route-refresh request. All received routes from that neighbor is sent to BMP servers.

Note In the case of BMP Pre Inbound Policy Route monitoring, when a new BMP server comes up, route refresh requests are sent to the peer router by the BGP speaker. However, in the case of BMP Post Inbound Policy Route Monitoring route refresh request are not sent to the peer routers when the new BMP server comes up because the BMP table is used for update generation.

It is advantageous to batch up refresh requests to BGP peers, if several BMP servers are activated in quick succession. Use the **bmp server initial-refresh-delay** command to configure a delay in triggering the refresh mechanism when the first BMP server comes up. If other BMP servers come online within this time-frame, only one set of refresh requests is sent to the BGP peers. You can also configure the **bmp server initial-refresh-delay skip** command to skip all refresh requests from BGP speakers and just monitor all incoming messages from the peers.

In a client-server configuration, it is recommended that the resource load of the devices be kept minimal and adding excessive network traffic must be avoided. In the BMP configuration, you can configure various delay timers on the BMP server to avoid flapping during connection between the server and client.

Recent Prefixes Events and Trace Support

The Recent Prefixes Events and Trace Support feature enables you to obtain per prefix level churning information without the use of debug commands. The show commands associated with this feature provide you a recent history of major events at the prefix level. They display the last eight events for the last 100 churning number of prefixes across an address family.

The following address families support this feature:

- IPv4 Unicast
- IPv6 Unicast
- IPv4 Multicast
- IPv6 Multicast
- VPNv4 Unicast

- VPNv6 Unicast
- BGP Link-State
- L2VPN EVPN
- IPv4 FlowSpec

Retrictions

The following restrictions apply to recent prefixes only. They do not apply to trace support.

- You can only track remote prefixes and path updates. You cannot track internal event trigger or local prefixes updates.
- You cannot track the events when the neighbor session goes down

Verification

Use the following command to check the events for a specific prefix.

```
Router# show bgp ipv4 unicast recent-prefixes 192.168.112.0/24 priv$

P/0/RP0/CPU0:root#
Tue Jan 21 10:30:44.488 UTC

Address-Family: IPv4 Unicast Route-Distinguisher: 0:0:0
192.168.112.0/24
Event History [Total events: 8]
------------------------------------
Time                 Event     Context1  Context2  Context3
====                 =====     =====     =====     =====
Dec 19 16:39:53.329  Withdraw  0x3010101  0x0     0x4000000000020004
Dec 19 16:39:53.330  Create    0x3010101  0x0     0x4000000000020005
Dec 19 16:39:53.330  Modify    0x3010101  0x0     0x4000000000020005
Dec 19 16:40:42.717  Create    0x3010101  0x0     0x4000000000020005
Dec 19 18:16:33.318  Create    0x3010101  0x0     0x4000000000020005
Jan 2 13:36:18.595  Modify     0x3010101  0x0     0x4000000000020005
Jan 2 15:16:00.344  Duplicate  0x3010101  0x0     0x4000000000020005
Jan 14 15:56:28.561 Duplicate 0x3010101   0x0     0x4000000000020005
------------------------------------
```

Verify the route distinguishers and corresponding prefix.

```
Router# show bgp l2vpn recent-prefixes

Address-Family     Route-Distinguisher          Prefix
==============     ===================          ==============
L2VPN EVPN         0:0:0 [5][0][32]            [198.51.100.22]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]     [192.0.2.1]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]     [192.0.2.2]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]     [192.0.2.3]/24
L2VPN EVPN         10.5.0.1:100 [5][0][32]     [192.0.2.4]/24
```

Verify recently updated or deleted prefixes.

```
Router# show bgp ipv4 unicast recent-prefixes

Address-Family   Route-Distinguisher Prefix
================ =================== ==============
IPv4 Unicast     0:0:0               10.1.1.1/32
```

```
IPv4 Unicast     0:0:0              10.1.1.101/32
IPv4 Unicast     0:0:0              10.1.1.100/32
IPv4 Unicast     0:0:0              10.1.1.99/32
IPv4 Unicast     0:0:0              10.1.1.98/32
IPv4 Unicast     0:0:0              10.1.1.93/32
```

Verify recently updated or deleted prefixes with timestamps and related contexts.

```
Router# show bgp ipv4 unicast recent-prefixes private

Address-Family: IPv4 Unicast Route-Distinguisher: 0:0:0
10.1.1.10/32
Event History [Total events: 4]
-------------------------------------
Time                Event     Context1    Context2  Context3
====                =====     =====       =====     =====
Jul 24 17:03:58.357 Create   0x13000001  0x0       0x4000000000000007
Jul 24 17:04:12.365 Withdraw 0x13000001  0x0       0x4000000001040006
Jul 24 17:04:31.625 Create   0x13000001  0x0       0x4000000000000007
Jul 24 17:04:39.880 Duplicate 0x13000001 0x0       0x4000000000000007
```

Verify recent history of major events in the link-state database of a network advertised through BGP.

```
Router# show bgp link-state link-state recent-prefixes

Address-Family: Link-state Link-state Route-Distinguisher: 0:0:0
[E][B][I0x0][N[c1][b19.0.0.1][q19.0.0.1]][R[c200][q19.0.0.2]][L[i26.0.101.100][n29.0.1.30]]/600
Event History [Total events: 4]
-------------------------------------
Time Event Context1 Context2 Context3
==== ===== ===== ===== =====
Aug 1 15:45:25.171 Create 0x13000001 0x0 0x4000000000020005
```

Reasons for not Advertising BGP Prefix to a Peer

The following are the categories of reasons for which a BGP prefix may not be advertised to a peer or a set of peers. The exact reason for which the BGP prefix is not advertised is displayed in the output of the show bgp ipv4 unicast update-group performance-statistics command.

- Path element not applicable
- Path not available
- Block stitching route targer (RT) constraint
- Block RT constraint network layer reachability information (NLRI)
- Imported path to non-customer edge (CE) neighbor
- VPN only path to CE neighbor
- External peer with no export
- Encapsulation mismatch (VxLAN)
- Sender Autonomous System (AS)
- Non-client to non-client
- Cluster identifier not set
- Client to non-client for cluster

- No PIM feedback for eBGP neighbor
- No PIM feedback
- PIM withdraw Feedback
- Wait for PIM feedback
- Prefix-based outbound route filter (ORF)
- RT type mismatch
- No out-policy for eBGP neighbor
- Out-policy
- Nexthop and label select fail
- V6 nexthop for V4 NLRI non-extended encoding capable
- No label
- Net suppressed
- No second label
- Dropped by RT filter
- Dropped by MVPN neighbor filter
- Oversized
- Split horizon update

Verification

The below example shows how to display performance statisticsfor a unadvertized prefix without enabling debug commands and checking the logs.

BGP prefix may not be advertized to a peer or a set of peers. The below example shows how to display the total numbers of prefixes not advertising in any AFI or SAFI, including repeating counts on 1 or more prefixes

```
Router# show bgp update-group performance-statistics

Update group for IPv4 Unicast, index 0.1:
..
Update timer last processed: Sep 23 00:10:15.350
  Not-Advertised Stats:
    Non-Client to Non-Client       : 105        Sep 23 00:10:15.350
    Path Not Available       : 132        Sep 23 00:10:15.350
```

BGP Slow Peer Automatic Isolation from Update Group

Table 8: Feature History Table

Feature Name	Release Information	Feature Description

BGP Slow Peer Automatic Isolation from Update Group	Release 7.3.1	A slow peer cannot keep up with the rate at which the router generates BGP update messages over a period of time, in an update group. This feature automatically detects a slow peer in an update group and moves it to a new update group. The feature is enabled on the router, by default. New commands introduced in this release: • **slow-peer detection enable** • **clear bgp slow-peers** Updated commands in this release: • **slow-peer detection disable**

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
BGP Slow Peer Automatic Isolation from Update Group	Release 7.3.1	A slow peer cannot keep up with the rate at which the router generates BGP update messages over a period of time, in an update group. This feature automatically detects a slow peer in an update group and moves it to a new update group. The feature is enabled on the router, by default. New commands introduced in this release: • **slow-peer detection enable** • **clear bgp slow-peers** Updated commands in this release: • **slow-peer detection disable**

The BGP Slow Peer Automatic Isolation from Update Group feature enables you to detect a slow peer in an update group and moving it to its own update group.

When a peer is slow in an BGP update group it cannot keep up with the rate at which update messages are generated over a prolonged time causing formatted messages to build up. The rest of the members of the group that are faster than the slow peer and have completed transmission of the formatted messages will not have anything new to send even though there may be newly modified BGP nets waiting to be advertised or withdrawn.

This feature enables you to detect a slow peer in an update group and moves it to its own update group. This feature is enabled by default.

When a slow peer is detected it is automatically moved to a new update group. Hence if there are slow peers then there will be an update group containing one or more slow peers corresponding to the original update group. There will be only one update group containing slow peers corresponding to the original update group. Hence, if multiple peers are slow, they will be in different sub-groups within the new slow update group. On recovery of the slow peer the peer is moved back to the original update group.

The presence of a slow peer in an update group, increases the number of formatted updates that are pending transmission. Events causing large churn in the BGP table, such as connection resets can result in a short-lived spike in the rate of update generation. A peer that temporarily falls behind during such events but quickly recovers after the event is not considered a slow peer.

This feature enables moving all the slow peers out of their original group, and into a new group dedicated to slow peers. After the slow peers are moved out, the non-slow members in the original group progress at their regular pace and catch up with the BGP table changes. The slow members consume updates at the slower pace and lag in their new dedicated group. One group for slow peers is required for each original group containing a slow peer. It is not possible to group together slow peers from different original groups as they will have a different outbound policy configuration.

Both the feature and splitting of update groups is enabled by default.

Configuration Examples

Detect Dynamic Slow Peers at the Global Configuration Level

Perform the following steps to disable slow peer detection globally and override all configuration under the neighbor. Any slow peers that are detected are marked as normal peers. They are moved back to their original update groups. No more slow peers are detected.

```
Router# configure
Router(config)# slow-peer-detection disable
```

Manually Configure Static Slow Peers at the Neighbor Configuration Level

Perform the following steps to control the behavior of the slow-peer detection and mitigation at neighbor configuration level. The configuration manually marks a neighbor as slow peer. Also, the peer will be part of slow update group.

```
Router(config)# router bgp 5
Router(config-bgp)# address-family ipv4
Router(config-bgp-af)# neighbor 172.60.2.3
Router(config-bgp-nbr-af)# slow-peer detection disable split-updategroup static
```

Configure Dynamic Slow Peers at the Neighbor Configuration Level

Use the split-update-group dynamic command to dynamically detect the slow peer and move it to a slow update group.

Note When the split-update-group dynamic command alone is configured, the dynamically detected slow peer is moved to a slow update group. If there already exists a slow peer update group, the dynamic slow peer is moved to slow peer update group, otherwise a new slow peer update group is created and the peer is moved to the new slow peer update group. This option is enabled by default.

Note If the permanent keyword is not configured, the slow peer is moved to its regular original update group, after it becomes regular peer. If the permanent keyword is configured, the peer will not be moved to its original update group automatically. The administrator can use clear command to move it to original update group. Use this option if a peer keeps becoming a slow peer and recovering.

```
Router(config)# router bgp 5
Router(config-bgp)# address-family ipv4
Router(config-bgp-af)# neighbor 172.60.2.3
Router(config-bgp-nbr-af)# slow-peer detection enable split-update-group
dynamic permanent
```

Clear Dynamically Detected Slow Peers

Perform the following task to clears all slow peers part of a specific address family identifiers (AFI) or subsequent address family identifiers (SAFI):

```
Router# clear bgp slow-peers <afi> <safi>
```

Perform the following task to clear all slow peers for all AFI or SAFI of the neighbor:

```
Router# clear bgp slow-peers <neighbor-address>
```

Perform the following task to clear the specified combination:

```
Router# clear bgp slow-peers <afi> <safi> <neighbor-address>
```

Running Configuration

This section shows the BGP Slow Peer Automatic Isolation from Update Group running configuration.

```
slow-peer-detection disable
router bgp 5
address-family ipv4
 neighbor 172.60.2.3
 slow-peer detection disable split-update-group static
router bgp 5
address-family ipv4
 neighbor 172.60.2.3
 slow-peer detection enable split-update-group dynamic permanent
```

Verification

```
Router# show bgp vrf <vrf-name> <afi> <safi> update out neighbor slowpeers <brief>
<SME to provide show output.>
```

```
Router# show bgp all all update out neighbor slow-peers
Fri Sep 13 13:57:48.503 PDT
Address Family: IPv4 Unicast
----------------------------
++++++++++++++++++AFTER 5 MINUTES ++++++++++++++++++++++
```

```
Router# show bgp all all update out neighbor slow-peers
Fri Sep 13 14:02:23.097 PDT
Address Family: IPv4 Unicast
----------------------------
VRF "default", Address-family "IPv4 Unicast"
Main routing table version: 3329832
RIB version: 3329832
Neighbor 11.11.11.21
Filter-group 0.3, Refresh filter-group ---
Sub-group 0.2, Refresh sub-group ---
Update-group 0.3
Update OutQ: 20447800 bytes (7680 messages) Refresh
update OutQ: 0 bytes (0 messages) Filter-group pending:
7680 messages
```

How to Implement BGP

Enabling BGP Routing

Perform this task to enable BGP routing and establish a BGP routing process. Configuring BGP neighbors is included as part of enabling BGP routing.

Note At least one neighbor and at least one address family must be configured to enable BGP routing. At least one neighbor with both a remote AS and an address family must be configured globally using the **address family** and **remote as** commands.

Before you begin

BGP must be able to obtain a router identifier (for example, a configured loopback address). At least, one address family must be configured in the BGP router configuration and the same address family must also be configured under the neighbor.

Note If the neighbor is configured as an external BGP (eBGP) peer, you must configure an inbound and outbound route policy on the neighbor using the **route-policy** command.

Note While establishing eBGP neighborship between two peers, BGP checks if the two peers are directly connected. If the peers are not directly connected, BGP does not try to establish a relationship by default. If two BGP peers are not directly connected and peering is required between the loop backs of the routers, you can use the **ignore-connected-check** command. This command overrides the default check that BGP performs which is to verify if source IP in BGP control packets is in same network as that of destination. In this scenario, a TTL value of 1 is sufficient if **ignore-connected-check** is used.

Configuring **egp-multihop** *ttl* is needed when the peers are not directly connected and there are more routers in between. If the **egp-multihop** *ttl* command is not configured, eBGP sets the TTL of packets carrying BGP messages to 1 by default. When eBGP needs to be setup between routers which are more than one hop away, you need to configure a TTL value which is at least equal to the number of hops between them. For example, if there are 2 hops (R2, R3) between two BGP peering routers R1 and R4, you need to set a TTL value of 3.

SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **end-policy**
4. Use the **commit** or **end** command.
5. **configure**
6. **router bgp** *as-number*
7. **bgp router-id** *ip-address*

8. **address-family** { **ipv4** | **ipv6** } **unicast**
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **address-family** { **ipv4** | **ipv6** } **unicast**
13. **route-policy** *route-policy-name* { **in** | **out** }
14. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**route-policy** *route-policy-name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# route-policy drop-as-1234`
	`RP/0/RSP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then`
	`RP/0/RSP0/CPU0:router(config-rpl)# apply check-communities`
	`RP/0/RSP0/CPU0:router(config-rpl)# else`
	`RP/0/RSP0/CPU0:router(config-rpl)# pass`
	`RP/0/RSP0/CPU0:router(config-rpl)# endif`
		(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 3	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 6	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 7	**bgp router-id** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24`	Configures the local router with a specified router ID.
Step 8	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 10	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 11	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 12	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 13	**route-policy** *route-policy-name* { **in** | **out** } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in`	(Optional) Applies the specified policy to inbound IPv4 unicast routes.

	Command or Action	Purpose
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Multiple BGP Instances for a Specific Autonomous System

Perform this task to configure multiple BGP instances for a specific autonomous system.

All configuration changes for a single BGP instance can be committed together. However, configuration changes for multiple instances cannot be committed together.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number* [**instance** *instance name*]
3. **bgp router-id** *ip-address*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* [**instance** *instance name*] **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100 instance inst1`	Enters BGP configuration mode for the user specified BGP instance.
Step 3	**bgp router-id** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.0`	Configures a fixed router ID for the BGP-speaking router (BGP instance). **Note** You must manually configure unique router ID for each BGP instance.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring a Routing Domain Confederation for BGP

Perform this task to configure the routing domain confederation for BGP. This includes specifying a confederation identifier and autonomous systems that belong to the confederation.

Configuring a routing domain confederation reduces the internal BGP (iBGP) mesh by dividing an autonomous system into multiple autonomous systems and grouping them into a single confederation. Each autonomous system is fully meshed within itself and has a few connections to another autonomous system in the same confederation. The confederation maintains the next hop and local preference information, and that allows you to retain a single Interior Gateway Protocol (IGP) for all autonomous systems. To the outside world, the confederation looks like a single autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp confederation identifier** *as-number*
4. **bgp confederation peers** *as-number*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp confederation identifier** *as-number* **Example:**	Specifies a BGP confederation identifier.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation identifier 5`	

Step 4	**bgp confederation peers** *as-number*
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1091`
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1092`
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1093`
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1094`
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1095`
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 1096`
		Specifies that the BGP autonomous systems belong to a specified BGP confederation identifier. You can associate multiple AS numbers to the same confederation identifier, as shown in the example.

Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Resetting an eBGP Session Immediately Upon Link Failure

By default, if a link goes down, all BGP sessions of any directly adjacent external peers are immediately reset. Use the **bgp fast-external-fallover disable** command to disable automatic resetting. Turn the automatic reset back on using the **no bgp fast-external-fallover disable** command.

eBGP sessions flap when the node reaches 3500 eBGP sessions with BGP timer values set as 10 and 30. To support more than 3500 eBGP sessions, increase the packet rate by using the **lpts pifib hardware police location** *location-id* command. Following is a sample configuration to increase the eBGP sessions:

```
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#lpts pifib hardware police location 0/2/CPU0
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#flow bgp configured rate 4000
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#flow bgp known rate 4000
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#flow bgp default rate 4000
RP/0/RSP0/CPU0:router(config-pifib-policer-per-node)#commit
```

Logging Neighbor Changes

Logging neighbor changes is enabled by default. Use the **log neighbor changes disable** command to turn off logging. The **no log neighbor changes disable** command can also be used to turn logging back on if it has been disabled.

Adjusting BGP Timers

Perform this task to set the timers for BGP neighbors.

BGP uses certain timers to control periodic activities, such as the sending of keepalive messages and the interval after which a neighbor is assumed to be down if no messages are received from the neighbor during the interval. The values set using the **timers bgp** command in router configuration mode can be overridden on particular neighbors using the **timers** command in the neighbor configuration mode.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **timers bgp** *keepalive hold-time*
4. **neighbor** *ip-address*
5. **timers** *keepalive hold-time*
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 123`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**timers bgp** *keepalive hold-time* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# timers bgp 30 90`	Sets a default keepalive time and a default hold time for all neighbors.
Step 4	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

	Command or Action	Purpose
Step 5	**timers** *keepalive hold-time* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# timers 60 220`	(Optional) Sets the keepalive timer and the hold-time timer for the BGP neighbor.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Changing the BGP Default Local Preference Value

Perform this task to set the default local preference value for BGP paths.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp default local-preference** *value*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	**bgp default local-preference** *value* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp default local-preference 200`	Sets the default local preference value from the default of 100, making it either a more preferable path (over 100) or less preferable path (under 100).
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring the MED Metric for BGP

Perform this task to set the multi exit discriminator (MED) to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **default-metric** *value*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	**default-metric** *value* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# default metric 10`	Sets the default metric, which is used to set the MED to advertise to peers for routes that do not already have a metric set (routes that were received with no MED attribute).
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Weights

Perform this task to assign a weight to routes received from a neighbor. A weight is a number that you can assign to a path so that you can control the best-path selection process. If you have particular neighbors that you want to prefer for most of your traffic, you can use the **weight** command to assign a higher weight to all routes learned from that neighbor.

Before you begin

Note The **clear bgp** command must be used for the newly configured weight to take effect.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family** { **ipv4** | **ipv6** } **unicast**
6. **weight** *weight-value*
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 6	**weight** *weight-value* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# weight 41150`	Assigns a weight to all routes learned through the neighbor.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Tuning the BGP Best-Path Calculation

Perform this task to change the default BGP best-path calculation behavior.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp bestpath med missing-as-worst**
4. **bgp bestpath med always**
5. **bgp bestpath med confed**
6. **bgp bestpath as-path ignore**
7. **bgp bestpath compare-routerid**
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 126`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp bestpath med missing-as-worst** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath med missing-as-worst`	Directs the BGP software to consider a missing MED attribute in a path as having a value of infinity, making this path the least desirable path.
Step 4	**bgp bestpath med always** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath med always`	Configures the BGP speaker in the specified autonomous system to compare MEDs among all the paths for the prefix, regardless of the autonomous system from which the paths are received.
Step 5	**bgp bestpath med confed** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath med confed`	Enables BGP software to compare MED values for paths learned from confederation peers.
Step 6	**bgp bestpath as-path ignore** **Example:**	Configures the BGP software to ignore the autonomous system length when performing best-path selection.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath as-path ignore`	
Step 7	**bgp bestpath compare-routerid** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp bestpath compare-routerid`	Configure the BGP speaker in the autonomous system to compare the router IDs of similar paths.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Indicating BGP Back-door Routes

Perform this task to set the administrative distance on an external Border Gateway Protocol (eBGP) route to that of a locally sourced BGP route, causing it to be less preferred than an Interior Gateway Protocol (IGP) route.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **network** { *ip-address* / *prefix-length* | *ip-address mask* } **backdoor**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**network** { *ip-address* / *prefix-length* | *ip-address mask* } **backdoor** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# network 172.20.0.0/16`	Configures the local router to originate and advertise the specified network.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Aggregate Addresses

Perform this task to create aggregate entries in a BGP routing table.

Note For optimal CPU utilization when deploying BGP aggregate for supernet addresses with a higher scale such as internet bgp table, it is recommended to:

- Use aggregate subnet of size not exceeding /24.
- Tune the subnet mask size based on network scale and churn.
- Use the **default-originate** or **network** 0.0.0.0 CLI instead of 0.0.0.0 as aggregate, when advertising the default route 0.0.0.0.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# aggregate-address 10.0.0.0/8 as-set`	Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized. • The **as-set** keyword generates autonomous system set path information and community information from contributing paths. • The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths. • The **summary-only** keyword filters all more specific routes from updates. • The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Redistributing iBGP Routes into IGP

Perform this task to redistribute iBGP routes into an Interior Gateway Protocol (IGP), such as Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF).

Note Use of the **bgp redistribute-internal** command requires the **clear route *** command to be issued to reinstall all BGP routes into the IP routing table.

Caution Redistributing iBGP routes into IGPs may cause routing loops to form within an autonomous system. Use this command with caution.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp redistribute-internal**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp redistribute-internal** **Example:**	Allows the redistribution of iBGP routes into an IGP, such as IS-IS or OSPF.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# bgp redistribute-internal`	
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Discard Extra Paths

Perform this task to configure BGP maximum-prefix discard extra paths.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **maximum-prefix** *maximum* **discard-extra-paths**
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters Global Configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 10`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

	Command or Action	Purpose
Step 4	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
Step 5	**maximum-prefix** *maximum* **discard-extra-paths** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths`	Configures a limit to the number of prefixes allowed. Configures discard extra paths to discard extra paths when the maximum prefix limit is exceeded.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Per Neighbor TCP MSS

Perform this task to configure TCP MSS under neighbor group, which is inherited by a neighbor.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv4 unicast**
4. **exit**
5. **neighbor-group** *name*
6. **tcp mss** *segment-size*
7. **address-family ipv4 unicast**
8. **exit**
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **use neighbor-group** *group-name*
13. **address-family ipv4 unicast**
14. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters Global Configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 10`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# exit`	Exits router address family configuration mode, and returns to BGP configuration mode.
Step 5	**neighbor-group** *name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group n1`	Enters neighbor group configuration mode.
Step 6	**tcp mss** *segment-size* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# tcp mss 500`	Configures TCP maximum segment size. The range is from 68 to 10000.
Step 7	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast`	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit`	Exits router address family configuration mode.
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit`	Exits the neighbor group configuration mode.

	Command or Action	Purpose
Step 10	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.2`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 11	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1`	Creates a neighbor and assigns a remote autonomous system (AS) number to it. • Range for 2-byte autonomous system numbers (ASNs) is 1 to 65535. • Range for 4-byte autonomous system numbers (ASNs) in asplain format is 1 to 4294967295. • Range for 4-byte autonomous system numbers (ASNs) is asdot format is 1.0 to 65535.65535.
Step 12	**use neighbor-group** *group-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group n1`	Specifies that the BGP neighbor inherit configuration from the specified neighbor group.

Step 13	**address-family ipv4 unicast**
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
	`RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#`
		Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling Per Neighbor TCP MSS

Perform this task to disable TCP MSS for a particular neighbor under neighbor group.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv4 unicast**
4. **exit**
5. **neighbor-group** *name*
6. **tcp mss** *segment-size*
7. **address-family ipv4 unicast**
8. **exit**
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **use neighbor-group** *group-name*
13. **tcp mss inheritance-disable**
14. **address-family ipv4 unicast**
15. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters Global Configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 10`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# exit`	Exits router address family configuration mode, and returns to BGP configuration mode.
Step 5	**neighbor-group** *name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group n1`	Enters neighbor group configuration mode.
Step 6	**tcp mss** *segment-size* **Example:**	Configures TCP maximum segment size. The range is from 68 to 10000.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# tcp mss 500`	
Step 7	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast`	Specifies the IPv4 address family unicast and enters address family configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit`	Exits router address family configuration mode.
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit`	Exits the neighbor group configuration mode.
Step 10	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.2`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 11	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1`	Creates a neighbor and assigns a remote autonomous system (AS) number to it. • Range for 2-byte autonomous system numbers (ASNs) is 1 to 65535. • Range for 4-byte autonomous system numbers (ASNs) in asplain format is 1 to 4294967295. • Range for 4-byte autonomous system numbers (ASNs) is asdot format is 1.0 to 65535.65535.
Step 12	**use neighbor-group** *group-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group n1`	Specifies that the BGP neighbor inherit configuration from the specified neighbor group.
Step 13	**tcp mss inheritance-disable** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# tcp mss inheritance-disable`	Disables TCP MSS for the neighbor.
Step 14	**address-family ipv4 unicast** **Example:**	Specifies the IPv4 address family unicast and enters address family configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast` `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#`	
Step 15	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Redistributing Prefixes into Multiprotocol BGP

Perform this task to redistribute prefixes from another protocol into multiprotocol BGP.

Redistribution is the process of injecting prefixes from one routing protocol into another routing protocol. This task shows how to inject prefixes from another routing protocol into multiprotocol BGP. Specifically, prefixes that are redistributed into multiprotocol BGP using the **redistribute** command are injected into the unicast database, the multicast database, or both.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. Do one of the following:
   - **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	Do one of the following: • **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# redistribute ospf 110`	Causes routes from the specified instance to be redistributed into BGP.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Route Dampening

Perform this task to configure and monitor BGP route dampening.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]
5. Use the **commit** or **end** command.
6. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics**
7. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics regexp** *regular-expression*
8. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **route-policy** *route-policy-name*
9. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] { *mask* | */prefix-length* }}
10. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** { *ip-address* [{ *mask* | */prefix-length* } [**longer-prefixes**
11. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics**
12. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics regexp** *regular-expression*
13. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **route-policy** *route-policy-name*
14. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** *network* / *mask-length*
15. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** *ip-address* / *mask-length*

16. **show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **dampened-paths**
17. **clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **dampening** *ip-address* / *mask-length*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120`	Configures BGP dampening for the specified address family. • *half-life*—(Optional) Time (in minutes) after which a penalty is decreased. Once the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default). Penalty reduction happens every 5 seconds. Range of the half-life period is from 1 to 45 minutes. • *reuse*—(Optional) Value for route reuse if the flapping route penalty decreases and falls below the reuse value. When this happens, the route is unsuppressed. The process of unsuppressing routes occurs at 10-second increments. Range is 1 to 20000. • *suppress*—(Optional) Maximum penalty value. Suppress a route when its penalty exceeds the value specified. When this happens, the route is suppressed. Range is 1 to 20000. • *max-suppress-time*—(Optional) Maximum time (in minutes) a route can be suppressed. Range is 1 to 255. If the *half-life* value is allowed to default, the maximum suppress time defaults to 60 minutes.

	Command or Action	Purpose
		• **route-policy** *route-policy-name* —(Optional) Specifies the route policy to use to set dampening parameters.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 6	**show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** **Example:** `RP/0/RSP0/CPU0:router# show bgp flap statistics`	Displays BGP flap statistics.
Step 7	**show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** **regexp** *regular-expression* **Example:** `RP/0/RSP0/CPU0:router# show bgp flap-statistics regexp _1$`	Displays BGP flap statistics for all paths that match the regular expression.
Step 8	**show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **route-policy** *route-policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# show bgp flap-statistics route-policy policy_A`	Displays BGP flap statistics for the specified route policy.

	Command or Action	Purpose
Step 9	**show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] { *mask* | */prefix-length* }} **Example:** `RP/0/RSP0/CPU0:router# show bgp flap-statistics 172.20.1.1`	Displays BGP flap for the specified prefix.
Step 10	**show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** { *ip-address* [{ *mask* | */prefix-length* } [**longer-prefixes** **Example:** `RP/0/RSP0/CPU0:router# show bgp flap-statistics 172.20.1.1 longer-prefixes`	Displays BGP flap statistics for more specific entries for the specified IP address.
Step 11	**clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** **Example:** `RP/0/RSP0/CPU0:router# clear bgp all all flap-statistics`	Clears BGP flap statistics for all routes.
Step 12	**clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics regexp** *regular-expression* **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics regexp _1$`	Clears BGP flap statistics for all paths that match the specified regular expression.
Step 13	**clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **route-policy** *route-policy-name*	Clears BGP flap statistics for the specified route policy.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics route-policy policy_A`	
Step 14	**clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** *network* / *mask-length* **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 192.168.40.0/24`	Clears BGP flap statistics for the specified network.
Step 15	**clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **flap-statistics** *ip-address* / *mask-length* **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast flap-statistics 172.20.1.1`	Clears BGP flap statistics for routes received from the specified neighbor.
Step 16	**show bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **dampened-paths** **Example:** `RP/0/RSP0/CPU0:router# show bgp dampened-paths`	Displays the dampened routes, including the time remaining before they are unsuppressed.
Step 17	**clear bgp** [**ipv4** { **unicast** | **multicast** | **labeled-unicast** | **all** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **labeled-unicast** } | **vpnv4 unicast** [**rd** *rd-address*] | **vrf** { *vrf-name* | **all** } [**ipv4** { **unicast** | **labeled-unicast** } | **ipv6 unicast**]] **dampening** *ip-address* / *mask-length* **Example:** `RP/0/RSP0/CPU0:router# clear bgp dampening`	Clears route dampening information and unsuppresses the suppressed routes. **Caution** Always use the **clear bgp dampening** command for an individual address-family. The **all** option for address-families with **clear bgp dampening** should never be used during normal functioning of the system. For example, use `clear bgp ipv4 unicast dampening prefix x.x.x./y`

Applying Policy When Updating the Routing Table

Perform this task to apply a routing policy to routes being installed into the routing table.

Before you begin

See the *Implementing Routing Policy on* Cisco ASR 9000 Series Router module of *Routing Configuration Guide for Cisco ASR 9000 Series Routers* (this publication) for a list of the supported attributes and operations that are valid for table policy filtering.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **table-policy** *policy-name*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120.6`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4**	**ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**table-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# table-policy tbl-plcy-A`	Applies the specified policy to routes being installed into the routing table.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Setting BGP Administrative Distance

Perform this task to specify the use of administrative distances that can be used to prefer one class of route over another.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **distance bgp** *external-distance internal-distance local-distance*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**distance bgp** *external-distance internal-distance local-distance* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# distance bgp 20 20 200`	Sets the external, internal, and local administrative distances to prefer one class of routes over another. The higher the value, the lower the trust rating.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring a BGP Neighbor Group and Neighbors

Perform this task to configure BGP neighbor groups and apply the neighbor group configuration to a neighbor. A neighbor group is a template that holds address family-independent and address family-dependent configurations associated with the neighbor.

After a neighbor group is configured, each neighbor can inherit the configuration through the **use** command. If a neighbor is configured to use a neighbor group, the neighbor (by default) inherits the entire configuration of the neighbor group, which includes the address family-independent and address family-dependent configurations. The inherited configuration can be overridden if you directly configure commands for the neighbor or configure session groups or address family groups through the **use** command.

You can configure an address family-independent configuration under the neighbor group. An address family-dependent configuration requires you to configure the address family under the neighbor group to enter address family submode.

From neighbor group configuration mode, you can configure address family-independent parameters for the neighbor group. Use the **address-family** command when in the neighbor group configuration mode.

After specifying the neighbor group name using the **neighbor group** command, you can assign options to the neighbor group.

Note All commands that can be configured under a specified neighbor group can be configured under a neighbor.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. exit
5. **neighbor-group** *name*
6. **remote-as** *as-number*
7. **address-family** { **ipv4** | **ipv6** } **unicast**
8. **route-policy** *route-policy-name* { **in** | **out** }
9. **exit**
10. **exit**
11. **neighbor** *ip-address*

12. **use neighbor-group** *group-name*
13. **remote-as** *as-number*
14. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	exit **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 5	**neighbor-group** *name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group nbr-grp-A`	Places the router in neighbor group configuration mode.
Step 6	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 7	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 8	**route-policy** *route-policy-name* { **in** | **out** } **Example:**	(Optional) Applies the specified policy to inbound IPv4 unicast routes.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy drop-as-1234 in`	
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp-af)# exit`	Exits the current configuration mode.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# exit`	Exits the current configuration mode.
Step 11	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 12	**use neighbor-group** *group-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# use neighbor-group nbr-grp-A`	(Optional) Specifies that the BGP neighbor inherit configuration from the specified neighbor group.
Step 13	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring a Route Reflector for BGP

Perform this task to configure a route reflector for BGP.

All the neighbors configured with the **route-reflector-client**command are members of the client group, and the remaining iBGP peers are members of the nonclient group for the local route reflector.

Together, a route reflector and its clients form a *cluster*. A cluster of clients usually has a single route reflector. In such instances, the cluster is identified by the software as the router ID of the route reflector. To increase redundancy and avoid a single point of failure in the network, a cluster can have more than one route reflector. If it does, all route reflectors in the cluster must be configured with the same 4-byte cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. The **bgp cluster-id** command is used to configure the cluster ID when the cluster has more than one route reflector.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp cluster-id** *cluster-id*
4. **neighbor** *ip-address*
5. **remote-as** *as-number*
6. **address-family** { **ipv4** | **ipv6** } **unicast**
7. **route-reflector-client**
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**bgp cluster-id** *cluster-id* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp cluster-id 192.168.70.1`	Configures the local router as one of the route reflectors serving the cluster. It is configured with a specified cluster ID to identify the cluster.
Step 4	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 5	**remote-as** *as-number* **Example:**	Creates a neighbor and assigns a remote autonomous system number to it.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2003`	
Step 6	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-nbr)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 7	**route-reflector-client** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures the router as a BGP route reflector and configures the neighbor as its client.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Route Filtering by Route Policy

Perform this task to configure BGP routing filtering by route policy.

Before you begin

See the *Implementing Routing Policy on* Cisco ASR 9000 Series Router module of *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide* (this publication) for a list of the supported attributes and operations that are valid for inbound and outbound neighbor policy filtering.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **end-policy**
4. **router bgp** *as-number*
5. **neighbor** *ip-address*
6. **address-family** { **ipv4** | **ipv6** } **unicast**
7. **route-policy** *route-policy-name* { **in** | **out** }
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**route-policy** *name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# route-policy drop-as-1234`
	`  RP/0/RSP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then`
	`  RP/0/RSP0/CPU0:router(config-rpl)# apply check-communities`
	`  RP/0/RSP0/CPU0:router(config-rpl)# else`
	`  RP/0/RSP0/CPU0:router(config-rpl)# pass`
	`  RP/0/RSP0/CPU0:router(config-rpl)# endif`
		(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 3	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 5	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 6	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 7	**route-policy** *route-policy-name* { **in** | **out** } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in`	Applies the specified policy to inbound routes.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Attribute Filtering

Perform the following tasks to configure BGP attribute filtering:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **attribute-filter group** *attribute-filter group name*
4. **attribute** *attribute code* { **discard** | **treat-as-withdraw** }

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**attribute-filter group** *attribute-filter group name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# attribute-filter group ag_discard_med`	Specifies the attribute-filter group name and enters the attribute-filter group configuration mode, allowing you to configure a specific attribute filter group for a BGP neighbor.
Step 4	**attribute** *attribute code* { **discard**	**treat-as-withdraw** } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-attrfg)# attribute 24 discard`	Specifies a single or a range of attribute codes and an associated action. The allowed actions are: • Treat-as-withdraw— Considers the update message for withdrawal. The associated IPv4-unicast or MP_REACH NLRIs, if present, are withdrawn from the neighbor's Adj-RIB-In.

	Command or Action	Purpose
		• Discard Attribute— Discards this attribute. The matching attributes alone are discarded and the rest of the Update message is processed normally.

Configuring BGP Next-Hop Trigger Delay

Perform this task to configure BGP next-hop trigger delay. The Routing Information Base (RIB) classifies the dampening notifications based on the severity of the changes. Event notifications are classified as critical and noncritical. This task allows you to specify the minimum batching interval for the critical and noncritical events.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **nexthop trigger-delay** { **critical** *delay* | **non-critical** *delay* }
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4**	**ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**nexthop trigger-delay** { **critical** *delay*	**non-critical** *delay* } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# nexthop trigger-delay critical 15000`	Sets the critical next-hop trigger delay.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling Next-Hop Processing on BGP Updates

Perform this task to disable next-hop calculation for a neighbor and insert your own address in the next-hop field of BGP updates. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the network device as the next hop.

Note Next-hop processing can be disabled for address family group, neighbor group, or neighbor address family.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family** { **ipv4** | **ipv6** } **unicast**
6. **next-hop-self**
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 206`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 6	**next-hop-self** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	Sets the next-hop attribute for all routes advertised to the specified neighbor to the address of the local router. Disabling the calculation of the best next hop to use when advertising a route causes all routes to be advertised with the local network device as the next hop.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Community and Extended-Community Advertisements

Perform this task to specify that community/extended-community attributes should be sent to an eBGP neighbor. These attributes are not sent to an eBGP neighbor by default. By contrast, they are always sent to iBGP neighbors. This section provides examples on how to enable sending community attributes. The **send-community-ebgp** keyword can be replaced by the **send-extended-community-ebgp** keyword to enable sending extended-communities.

If the **send-community-ebgp** command is configured for a neighbor group or address family group, all neighbors using the group inherit the configuration. Configuring the command specifically for a neighbor overrides inherited values.

Note BGP community and extended-community filtering cannot be configured for iBGP neighbors. Communities and extended-communities are always sent to iBGP neighbors under VPNv4, MDT, IPv4, and IPv6 address families.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family**{**ipv4** {**labeled-unicast** | **unicast** | **mdt** | **multicast** | **mvpn** | **tunnel**} | **ipv6** {**labeled-unicast** | **mvpn** | **unicast**}}
6. Use one of these commands:
   - **send-community-ebgp**
   - **send-extended-community-ebgp**
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	**address-family**{**ipv4** {**labeled-unicast**	**unicast**	**mdt**	**multicast**	**mvpn**	**tunnel**}	**ipv6** {**labeled-unicast**	**mvpn**	**unicast**}} **Example:**	Enters neighbor address family configuration mode for the specified address family. Use either **ipv4** or **ipv6** address family keyword with one of the specified address family sub mode identifiers. IPv6 address family mode supports these sub modes:

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast`	• **labeled-unicast** • **mvpn** • **unicast** IPv4 address family mode supports these sub modes: • **labeled-unicast** • **mdt** • **multicast** • **mvpn** • **rt-filter** • **tunnel** • **unicast** Refer the **address-family (BGP)** command in *BGP Commands* module of *Routing Command Reference for Cisco ASR 9000 Series Routers* for more information on the Address Family Submode support.

Step 6	Use one of these commands:
	• **send-community-ebgp**
	• **send-extended-community-ebgp**
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# send-community-ebgp`
	or
	`RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# send-extended-community-ebgp`
		Specifies that the router send community attributes or extended community attributes (which are disabled by default for eBGP neighbors) to a specified eBGP neighbor.

Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring the BGP Cost Community

Perform this task to configure the BGP cost community.

BGP receives multiple paths to the same destination and it uses the best-path algorithm to decide which is the best path to install in RIB. To enable users to determine an exit point after partial comparison, the cost community is defined to tie-break equal paths during the best-path selection process.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set extcommunity cost** { *cost-extcommunity-set-name* | *cost-inline-extcommunity-set* } [**additive**]
4. **end-policy**
5. **router bgp** *as-number*
6. Do one of the following:
   - **default-information originate**
   - **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
7. Do one of the following:
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
   - **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** }
   - **route-policy** *route-policy-name* { **in** | **out** }

8. Use the **commit** or **end** command.
9. **show bgp** [**vrf** *vrf-name*] *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy costA`	Enters route policy configuration mode and specifies the name of the route policy to be configured.
Step 3	**set extcommunity cost** { *cost-extcommunity-set-name* | *cost-inline-extcommunity-set* } [**additive**] **Example:** `RP/0/RSP0/CPU0:router(config)# set extcommunity cost cost_A`	Specifies the BGP extended community attribute for cost.
Step 4	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config)# end-policy`	Ends the definition of a route policy and exits route policy configuration mode.
Step 5	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 6	Do one of the following: • **default-information originate** • **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** }	Applies the cost community to the attach point (route policy).

	Command or Action	Purpose
	redistribute isis *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv6 unicast** | **vpnv4 unicast** } **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]	
Step 7	Do one of the following: • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } **network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*] • **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4 unicast** | **ipv4 multicast** | **ipv4 tunnel** | **ipv4 mdt** | **ipv6 unicast** | **ipv6 multicast** | **vpnv4 unicast** | **vpnv6 unicast** } • **route-policy** *route-policy-name* { **in** | **out** }	
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 9	**show bgp** [**vrf** *vrf-name*] *ip-address* **Example:** `RP/0/RSP0/CPU0:router# show bgp 172.168.40.24`	Displays the cost community in the following format: Cost: *POI* : *cost-community-ID* : *cost-number*

Configuring Software to Store Updates from a Neighbor

Perform this task to configure the software to store updates received from a neighbor.

The **soft-reconfiguration inbound** command causes a route refresh request to be sent to the neighbor if the neighbor is route refresh capable. If the neighbor is not route refresh capable, the neighbor must be reset to relearn received routes using the **clear bgp soft** command. See the Resetting Neighbors Using BGP Inbound Soft Reset, on page 208.

Note Storing updates from a neighbor works only if either the neighbor is route refresh capable or the **soft-reconfiguration inbound** command is configured. Even if the neighbor is route refresh capable and the **soft-reconfiguration inbound** command is configured, the original routes are not stored unless the **always** option is used with the command. The original routes can be easily retrieved with a route refresh request. Route refresh sends a request to the peer to resend its routing information. The **soft-reconfiguration inbound** command stores all paths received from the peer in an unmodified form and refers to these stored paths during the clear. Soft reconfiguration is memory intensive.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. **soft-reconfiguration inbound** [**always**]
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 5	**soft-reconfiguration inbound** [**always]** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# soft-reconfiguration inbound always`	Configures the software to store updates received from a specified neighbor. Soft reconfiguration inbound causes the software to store the original unmodified route in addition to a route that is modified or filtered. This allows a "soft clear" to be performed after the inbound policy is changed. Soft reconfiguration enables the software to store the incoming updates before apply policy if route refresh is not supported by the peer (otherwise a copy of the update is not stored). The **always** keyword forces the software to store a copy even when route refresh is supported by the peer.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Persistence

BGP persistence enables the local router to retain routes that it has learnt from the configured neighbor even after the neighbor session is down. BGP persistence is also referred as Long Lived Graceful Restart (LLGR). LLGR takes effect after graceful restart (GR) ends or immediately if GR is not enabled. LLGR ends either when the LLGR stale timer expires or when the neighbor sends the end-of-RIB marker after it has revised its routes. When LLGR for a neighbor ends, all routes from that neighbor that are still stale will be deleted. The

LLGR capability is signaled to a neighbor in the BGP OPEN message if it has been configured for that neighbor. LLGR differs from graceful restart in the following ways.

- It can be in effect for a much longer time than GR
- LLGR stale routes are least preferred during route selection (bestpath computation).
- An LLGR stale route will be advertised with the LLGR_STALE community attached if it is selected as best path. It will not be advertised at all to routers that are not LLGR capable.
- LLGR stale routes will not be deleted when the forwarding path to the neighbor is detected to be down
- An LLGR stale route will not be deleted if the BGP session to the neighbor goes down multiple times even if that neighbor does not re-advertise the route.
- Any route that has the NO_LLGR community will not be retained.

Note You can disable GR helper-only for peer-group and neighbor, when there is no global GR helper-only configured.

BGP will not pass the updates containing communities 65535:6, 65535:7 to its neighbors until the neighbors negotiate BGP persistence capabilities. The communities 65535:6 and 65535:7 are reserved for LLGR_STALE and NO_LLGR respectively, BGP behavior maybe unpredictable if you have configured these communities prior to release 5.2.2. We recommend not to configure the communities 65535:6 and 65535:7.

The BGP persistence feature is supported only on the following AFIs:

- VPNv4 and VPNv6
- RT constraint
- Flow spec (IPv4, IPv6, VPNv4 and VPNv6)
- Private IPv4 and IPv6 (IPv4/v6 address family inside VRF)

BGP Persistence Configuration: Example

This example sets long lived graceful restart (LLGR) stale-time of 16777215 on BGP neighbor 3.3.3.3.

```
router bgp 100
 neighbor 3.3.3.3
  remote-as 30813
  update-source Loopback0
  graceful-restart stalepath-time 150
  address-family vpnv4 unicast
   long-lived-graceful-restart capable
   long-lived-graceful-restart stale-time send 16777215 accept 16777215
  !
  address-family vpnv6 unicast
   long-lived-graceful-restart capable
   long-lived-graceful-restart stale-time send 16777215 accept 16777215
```

BGP Graceful Maintenance

When a BGP link or router is taken down, other routers in the network find alternative paths for the traffic that was flowing through the failed router or link, if such alternative paths exist. The time required before all routers involved can reach a consensus about an alternate path is called convergence time. During convergence time, traffic that is directed to the router or link that is down is dropped. The BGP Graceful Maintenance feature allows the network to perform convergence before the router or link is taken out of service. The router or link remains in service while the network reroutes traffic to alternative paths. Any traffic that is yet on its way to the affected router or link is still delivered as before. After all traffic has been rerouted, the router or link can safely be taken out of service.

The Graceful Maintenance feature is helpful when alternate paths exist and these alternate paths are not known to routers at the time that the primary paths are withdrawn. The feature provides these alternate paths before the primary paths are withdrawn. The feature is most helpful in networks where convergence time is long. Several factors, such as large routing tables and presence of route reflectors, can result in longer convergence time.

When a BGP router or link is brought into service, the possibility of traffic loss during convergence also exists, although it is less than when a router or link is taken out of service. The BGP Graceful Maintenance feature can also be used in this scenario.

Restrictions for BGP Graceful Maintenance

The following restrictions apply for BGP Graceful Maintenance:

- If the affected router is configured to send the GSHUT community attribute, then other routers in the network that receive it must be configured to interpret it. You must match the community with a routing policy and set a lower preference.
- The LOCAL_PREF attribute is not sent to another AS. Therefore, the LOCAL_PREF option cannot be used on an eBGP link.

Note This restriction does not apply to eBGP links between member-ASs of an AS confederation.

- Alternative routes must exist in the network, otherwise advertising a lower preference has no effect. For example, there is no advantage in configuring Graceful Maintenance for a singly-homed customer router which does not have alternate routes.
- If time consuming policies exist, either at the output of the sending router or at the input of the receiving router, the Graceful Maintenance operation can take a long time.
- Configuring an eBGP ASBR neighbor results in advertising an implicit null label for directly connected routes via BGP. If a user shuts down an eBGP neighbor, the label is not reprogrammed as the system withdraws rewrites on any neighbor state changes. Implicit null label feature support helps avoid churn in terms of adding or removing rewrites for neighbor flaps.

Graceful Maintenance Operation

When Graceful Maintenance is activated, the affected routes are advertised again with a reduced preference. This causes neighboring routers to choose alternative routes. You can use any of the following methods to a signal reduced route preference:

- **Add GSHUT community:** Use this method to allow remote routers the freedom to set a preference. Receiving routers must match this community in a policy and set their own preference.
- **Reduce LOCAL_PREF value:** This works for internal BGP neighbors. Use this method if remote routers do not match the GSHUT community.
- **Prepend AS Path:** This works for both internal and external BGP neighbors. Use this method if remote routers do not match the GSHUT community.

When Graceful Maintenance is activated on a BGP connection, the following two operations happen:

1. All routes received from the connection are re-advertised to other neighbors with a lower preference. Note, this happens to only those routes that have actually been advertised to other neighbors. It is possible that a received route was not selected as the best path and therefore not advertised. In that case, it will not be re-advertised.
2. All routes that were advertised to the connection is re-advertised with a lower preference.

In order for the first operation to happen, all routes received from the connection are tagged with an internal attribute called graceful-shut. This attribute is stored internal to only the router; it is not advertised by BGP. This attribute can be seen when the route is displayed with the **show bgp** command. It is different from the GSHUT community. The GSHUT community is advertised by BGP and can be seen in the community list when the route is displayed with the **show bgp** command.

All routes that have the graceful-shut attribute are given the lowest preference during route-selection. Any new route updates that are sent or received on a BGP session under Graceful Maintenance are also treated as described above.

Inter Autonomous System

Advertising a lower preference to another AS in the public Internet may cause unnecessary routing advertisements in distant networks, which may not be desirable. An additional configuration under the neighbor address family, **send-community-gshut-ebgp**, is necessary for the router to originate the GSHUT community to the eBGP neighbor.

Note This does not affect the GSHUT community on a route that already had this community when it was received; it only affects the GSHUT community when this router adds it.

No Automatic Shutdown

The Graceful Maintenance feature does not perform any shutdown. When Graceful Maintenance is configured, it remains configured, even through system restarts. It is intended to be used in conjunction with a shutdown of a router or a BGP neighbor. The operator must explicitly shut down whenever it is needed. After Graceful Maintenance is no longer required, the operator must explicitly deactivate it. Graceful Maintenance may be deactivated either after the shutdown is completed, or after the deactivated facilities are again brought up. Whether to leave Graceful Maintenance activated through a bring-up operation depends on whether the transient routing during the bring-up operation is considered a problem.

When to Shut Down After Graceful Maintenance

The router or link can be shut down after the network has converged as a result of a graceful-maintenance activation. Convergence can take from less than a second to more than an hour. Unfortunately, a single router cannot know when a whole network has converged. After a graceful-maintenance activation, it can take a few seconds to start sending updates. Then, the “InQ” and “OutQ” of neighbors in the **show bgp <vrf> <afi> <safi> summary** command's output indicates the level of BGP messaging. Both InQ and OutQ should be 0 after convergence. Neighbors should stop sending traffic. However, they won't stop sending traffic if they do not have alternate paths; and in that case traffic loss cannot be prevented.

Activate Graceful Maintenance under BGP Router (All Neighbors)

Activating Graceful Maintenance under a BGP router results in **activate**being configured under **graceful-maintenance** for all neighbors. With just this one configuration, you get the same result if you were to go to every neighbor that has **graceful-maintenance** configured, and added **activate** under it. If you add the keyword **all-neighbors**, thus, **graceful-maintenance activate all-neighbors**, then the router acts as if you configured **graceful-maintenance activate** under every neighbor.

Note We suggest that you activate Graceful Maintenance under a BGP router instance only if it is acceptable to send the GSHUT community for all routes on every neighbor. Re-sending all routes to every neighbor can take significant amount of time on a large router. Sending GSHUT to a neighbor that does not have alternative routes is pointless. If a router has many of such neighbors then a significant amount of time can be saved by not activating Graceful Maintenance on them.

The BGP Graceful Maintenance feature allows you to enable Graceful Maintenance either on a single neighbor, on a group of neighbors across BGP sessions, or on all neighbors. Enabling Graceful Maintenance under a neighbor sub-mode, does two things:

1. All routes that are advertised to this neighbor that has the graceful-shut attribute are advertised to that neighbor with the GSHUT community.
2. Enters graceful-maintenance configuration mode to allow further configuration.

Using the **activate** keyword under graceful-maintenance, causes the following:

1. All routes that are received from this neighbor acquire the graceful-shut attribute.
2. All routes that are advertised to this neighbor are re-advertised to that neighbor with the GSHUT community.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **graceful-maintenance activate** [**all-neighbors** | **retain-routes**]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**graceful-maintenance activate** [**all-neighbors** | **retain-routes**] **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# graceful-maintenance activate all-neighhbors`	Announces routes with the g-shut community and other attributes as configured under the neighbors. This causes neighbors to reject routes from this router and choose alternates. This allows the router to be gracefully brought in or out of service. If you use the **all-neighbors** keyword, Graceful Maintenance is activated even for those neighbors that do not have it activated. Choosing **retain-routes** causes RIB to retain BGP routes when the BGP process is stopped. Use the **retain-routes** option when only BGP must be brought down instead of the entire router, and when it is known that neighboring routers are kept in operation during the maintenance of the local BGP. If RIB has alternative routes provided by another protocol or a default route, then it is recommended that you do not to retain BGP routes after the BGP process stops.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

After activating Graceful Maintenance, you must wait for all the routes to be sent and for the neighboring routers to redirect their traffic away from the router or link under maintenance. After the traffic is redirected, then it is safe to take the router or link out of service. While there is no definitive way to know when all the routes have been sent, you can use the **show bgp summary** command to check the OutQ of the neighbors. When OutQ reaches a value 0, there are no more updates to be sent.

Activate Graceful Maintenance on a Single Neighbor

Use the following steps to activate Graceful Maintenance for a single neighbor:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **graceful-maintenance activate**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**graceful-maintenance activate** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# graceful-maintenance activate`	Announces routes with Graceful Maintenance attributes.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Activate Graceful Maintenance on a Group of Neighbors

Use the following steps to activate Graceful Maintenance on a group of neighbors:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor-group** *Neighbor-group name*
4. **graceful-maintenance activate**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor-group** *Neighbor-group name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor-group AS_1`	Places the router in neighbor group configuration mode.
Step 4	**graceful-maintenance activate** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbrgrp)# graceful-maintenance activate`	Announces routes with Graceful Maintenance attributes.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

You must configure the **send-community-gshut-ebgp** command under the neighbor address family of an eBGP neighbor for this router to add the GSHUT community.

Note Sending GSHUT community may not be desirable under every address family of an eBGP neighbor. To allow you to target GSHUT community to a specific set of address families, use the **send-community-gshut-ebgp** command.

Direct Router to Reduce Route Preference

The BGP Graceful Maintenance feature works only with the availability of alternate paths. You must advertise routes with a lower preference to allow alternate routes to take over before taking down a link or router. Use the following steps to modify the route preference:

Note Attributes for graceful maintenance are added to a route update message after an outbound policy has been applied to it.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **graceful-maintenance** **as-prepends** *value*| **local-preference** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

	Command or Action	Purpose
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	**graceful-maintenance as-prepends** *value*| **local-preference** *value* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# graceful-maintenance local-preference 4`	Specifies the number of times the local AS number is to be to prepended to the AS path of routes and advertises the GSHUT community with the local preference value specified for the routes. When the router adds the GSHUT community to a route as it advertises it, it also changes the LOCAL_PREF attribute and prepends the local AS number as specified in the commands. Sending GSHUT provides flexibility in the manner in which neighboring routers handle the lower preference: they can match it in a route policy and do the most appropriate thing with it. On the other hand, in simple networks, it is easier to set local-preference to 0, than to create route policies everywhere else. **Note** LOCAL_PREF is not sent to real eBGP neighbors, but sent to confederation member AS eBGP neighbors. To lower the preference to eBGP neighbors, as-prepends value is required.

Example: Configure route policy matching GSHUT community to lower route preference

```
route-policy gshut
  if community matches-any gshut then
    set local-preference 0
  endif
  pass
end-policy


neighbor 666.0.0.3
    address-family ipv4 unicast
      route-policy gshut in
```

Note Routes received from a GSHUT neighbor are marked with a GSHUT attribute to distinguish them from routes received with the GSHUT community. When a neighbor is taken out of maintenance, the attribute on its paths is removed, but not the community. The attribute is internal and not sent in BGP messages. It is used to reject routes during path selection.

Bring Router or Link Back into Service

Before you bring the router or link back into service, you must first activate graceful maintenance and then remove the **activate** configuration.

Show Command Outputs to Verify BGP Graceful Maintenance

This section lists the show commands you can use to verify that BGP Graceful Maintenance is activated and check related attributes:

Use the **show bgp <IP address>** command to display graceful-shutdown community and the graceful-shut path attribute with BGP graceful maintenance activated:

```
RP/0/0/CPU0:R4#show bgp 5.5.5.5
...
10.10.10.1 from 10.10.10.1 (192.168.0.5)
Received Label 24000
Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best,
import-candidate
Received Path ID 0, Local Path ID 1, version 4
Community: graceful-shutdown
Originator: 192.168.0.5, Cluster list: 192.168.0.1
```

The following is sample output from the **show bgp community graceful-shutdown** command displaying the graceful maintenance feature information:

```
RP/0/0/CPU0:R4#show bgp community graceful-shutdown
BGP router identifier 192.168.0.4, local AS number 4
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0xe0000000 RD version: 18
BGP main routing table version 18
BGP scan interval 60 secs
Status codes: s suppressed, d damped, h history, * valid, > best
i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Network Next Hop Metric LocPrf Weight Path
* 5.5.5.5/32 10.10.10.1 88 0 1 ?
Processed 1 prefixes, 1 paths
```

The following is the sample output from the **show bgp neighbors** command with the ip-address and configuration argument and keyword to display graceful maintenance feature attributes:

```
RP/0/0/CPU0:R1#show bgp neighbor 12.12.12.5
...
Graceful Maintenance locally active, Local Pref=45, AS prepends=3
...
For Address Family: IPv4 Unicast
...
GSHUT Community attribute sent to this neighbor
...
************************************************************************
RP/0/0/CPU0:R1#show bgp neighbor 12.12.12.5 configuration
neighbor 12.12.12.5
remote-as 1 []
graceful-maintenance 1 []
gr-maint local-preference 45 []
gr-maint as-prepends 3 []
gr-maint activate []
```

The following is the sample output of the **show rpl community-set** command with graceful maintenance feature attributes displayed:

```
RP/0/0/CPU0:R5#show rpl community-set
Listing for all Community Set objects
community-set gshut
graceful-shutdown
end-set
```

The following is the sample of the syslog that is issued when a BGP neighbor that has graceful maintenance activated, comes up. It is a warning text that reminds you to deactivate graceful maintenance after convergence.

```
RP/0/0/CPU0:Jan 28 22:01:36.356 : bgp[1056]: %ROUTING-BGP-5-ADJCHANGE : neighbor 10.10.10.4
 Up (VRF: default) (AS: 4)
WARNING: Graceful Maintenance is Active
```

Flow-tag propagation

The flow-tag propagation feature enables you to establish a co-relation between route-policies and user-policies. Flow-tag propagation using BGP allows user-side traffic-steering based on routing attributes such as, AS number, prefix lists, community strings and extended communities. Flow-tag is a logical numeric identifier that is distributed through RIB as one of the routing attribute of FIB entry in the FIB lookup table. A flow-tag is instantiated using the 'set' operation from RPL and is referenced in the C3PL PBR policy, where it is associated with actions (policy-rules) against the flow-tag value.

You can use flow-tag propagation to:

- Classify traffic based on destination IP addresses (using the Community number) or based on prefixes (using Community number or AS number).
- Select a TE-group that matches the cost of the path to reach a service-edge based on customer site service level agreements (SLA).
- Apply traffic policy (TE-group selection) for specific customers based on SLA with its clients.
- Divert traffic to application or cache server.

For more information on the commands for flow-tag propagation see the BGP Commands module in the *Routing Command Reference for Cisco ASR 9000 Series Routers*.

Restrictions for flow-tag propagation

Some restrictions are placed with regard to using Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) and flow-tag feature together in a ASR9K platform. These include:

- A route-policy can have either 'set qos-group' or 'set flow-tag,' but not both for a prefix-set.
- Route policy for qos-group and route policy flow-tag cannot have overlapping routes. The QPPB and flow tag features can coexist (on same as well as on different interfaces) as long as the route policy used by them do not have any overlapping route.
- Mixing usage of qos-group and flow-tag in route-policy and policy-map is not recommended.

Source and destination-based flow tag

The source-based flow tag feature allows you to match packets based on the flow-tag assigned to the source address of the incoming packets. Once matched, you can then apply any supported PBR action on this policy.

Configure Source and Destination-based Flow Tag

This task applies flow-tag to a specified interface. The packets are matched based on the flow-tag assigned to the source address of the incoming packets.

Note You will not be able to enable both QPPB and flow tag feature simultaneously on an interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 | ipv6 bgp policy propagation input flow-tag**{**destination** | **source**}
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-if)# interface GigabitEthernet 0/0/0/0`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 | ipv6 bgp policy propagation input flow-tag**{**destination** | **source**} **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 bgp policy propagation input flow-tag source`	Enables flow-tag policy propagation on source or destination IP address on an interface.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Example

The following show commands display outputs with PBR policy applied on the router:

```
show running-config interface gigabitEthernet 0/0/0/12
Thu Feb 12 01:51:37.820 UTC
interface GigabitEthernet0/0/0/12
 service-policy type pbr input flowMatchPolicy
 ipv4 bgp policy propagation input flow-tag source
 ipv4 address 192.5.1.2 255.255.255.0
!

RP/0/RSP0/CPU0:ASR9K-0#show running-config policy-map type pbr flowMatchPolicy
Thu Feb 12 01:51:45.776 UTC
policy-map type pbr flowMatchPolicy
 class type traffic flowMatch36
  transmit
 !
 class type traffic flowMatch38
  transmit
 !
 class type traffic class-default
 !
 end-policy-map
!

RP/0/RSP0/CPU0:ASR9K-0#show running-config class-map type traffic flowMatch36
Thu Feb 12 01:52:04.838 UTC
class-map type traffic match-any flowMatch36
 match flow-tag 36
 end-class-map
!
```

Configuring a VPN Routing and Forwarding Instance in BGP

Layer 3 (virtual private network) VPN can be configured only if there is an available Layer 3 VPN license for the line card slot on which the feature is being configured. If advanced IP license is enabled, 4096 Layer 3 VPN routing and forwarding instances (VRFs) can be configured on an interface. If the infrastructure VRF license is enabled, eight Layer 3 VRFs can be configured on the line card.

See the Software Entitlement on Cisco ASR 9000 Series Router module in *System Management Configuration Guide for Cisco ASR 9000 Series Routers* for more information on advanced IP licencing.

The following error message appears if the appropriate licence is not enabled:

```
RP/0/RSP0/CPU0:router#LC/0/0/CPU0:Dec 15 17:57:53.653 : rsi_agent[247]:
%LICENSE-ASR9K_LICENSE-2-INFRA_VRF_NEEDED : 5 VRF(s) are configured without license
A9K-iVRF-LIC in violation of the Software Right To Use Agreement.
This feature may be disabled by the system without the appropriate license.
Contact Cisco to purchase the license immediately to avoid potential service interruption.
```

Note An AIP license is not required for configuring L2VPN services.

The following tasks are used to configure a VPN routing and forwarding (VRF) instance in BGP:

Defining Virtual Routing and Forwarding Tables in Provider Edge Routers

Perform this task to define the VPN routing and forwarding (VRF) tables in the provider edge (PE) routers.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **maximum prefix** *maximum* [*threshold*]
5. **import route-policy** *policy-name*
6. **import route-target** [*as-number* **:** *nn* | *ip-address* **:** *nn*]
7. **export route-policy** *policy-name*
8. **export route-target** [*as-number* **:** *nn* | *ip-address* **:** *nn*]
9. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config)# vrf vrf_pe`	Configures a VRF instance.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**maximum prefix** *maximum* [*threshold*] **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# maximum prefix 2300`	Configures a limit to the number of prefixes allowed in a VRF table. A maximum number of routes is applicable to dynamic routing protocols as well as static or connected routes. You can specify a threshold percentage of the prefix limit using the *mid-threshold* argument.
Step 5	**import route-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# import route-policy policy_a`	(Optional) Provides finer control over what gets imported into a VRF. This import filter discards prefixes that do not match the specified *policy-name* argument.
Step 6	**import route-target** [*as-number* **:** *nn* | *ip-address* **:** *nn*] **Example:**	Specifies a list of route target (RT) extended communities. Only prefixes that are associated with the specified import route target extended communities are imported into the VRF.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 234:222`	
Step 7	**export route-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy policy_b`	(Optional) Provides finer control over what gets exported into a VRF. This export filter discards prefixes that do not match the specified *policy-name* argument.
Step 8	**export route-target** [*as-number* **:** *nn* | *ip-address* **:** *nn*] **Example:** `RP/0/RSP0/CPU0:routerr(config-vrf-af)# export route-target 123;234`	Specifies a list of route target extended communities. Export route target communities are associated with prefixes when they are advertised to remote PEs. The remote PEs import them into VRFs which have import RTs that match these exported route target communities.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring the Route Distinguisher

The route distinguisher (RD) makes prefixes unique across multiple VPN routing and forwarding (VRF) instances.

In the L3VPN multipath same route distinguisher (RD)environment, the determination of whether to install a prefix in RIB or not is based on the prefix's bestpath. In a rare misconfiguration situation, where the best pah is not a valid path to be installed in RIB, BGP drops the prefix and does not consider the other paths. The behavior is different for different RD setup, where the non-best multipath will be installed if the best multipath is invalid to be installed in RIB.

Perform this task to configure the RD.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp router-id** *ip-address*
4. **vrf** *vrf-name*
5. **rd** { *as-number* **:** *nn* | *ip-address* **:** *nn* | **auto** }
6. Do one of the following:

- **end**
- **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode allowing you to configure the BGP routing process.
Step 3	**bgp router-id** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.0`	Configures a fixed router ID for the BGP-speaking router.
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_pe`	Configures a VRF instance.
Step 5	**rd** { *as-number* **:** *nn* | *ip-address* **:** *nn* | **auto** } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd 345:567`	Configures the route distinguisher. Use the **auto** keyword if you want the router to automatically assign a unique RD to the VRF. Automatic assignment of RDs is possible only if a router ID is configured using the **bgp router-id** command in router configuration mode. This allows you to configure a globally unique router ID that can be used for automatic RD generation. The router ID for the VRF does not need to be globally unique, and using the VRF router ID would be incorrect for automatic RD generation. Having a single router ID also helps in checkpointing RD information for BGP graceful restart, because it is expected to be stable across reboots.
Step 6	Do one of the following: • **end** • **commit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# end`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:`

	Command or Action	Purpose
	or `RP/0/RSP0/CPU0:router(config-bgp-vrf)# commit`	• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC configuration mode. • Entering **no** exits the configuration session and returns the router to EXEC configuration mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring PE-PE or PE-RR Interior BGP Sessions

To enable BGP to carry VPN reachability information between provider edge (PE) routers you must configure the PE-PE interior BGP (iBGP) sessions. A PE uses VPN information carried from the remote PE router to determine VPN connectivity and the label value to be used so the remote (egress) router can demultiplex the packet to the correct VPN during packet forwarding.

The PE-PE, PE-route reflector (RR) iBGP sessions are defined to all PE and RR routers that participate in the VPNs configured in the PE router.

Perform this task to configure PE-PE iBGP sessions and to configure global VPN options on a PE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family vpnv4 unicast**
4. **exit**
5. **neighbor** *ip-address*
6. **remote-as** *as-number*
7. **description** *text*
8. **password** { **clear** | **encrypted** } *password*
9. **shutdown**
10. **timers** *keepalive hold-time*
11. **update-source** *type interface-id*
12. **address-family vpnv4 unicast**
13. **route-policy** *route-policy-name* **in**
14. **route-policy** *route-policy-name* **out**
15. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpvn4 unicast`	Enters VPN address family configuration mode.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 5	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.16.1.1`	Configures a PE iBGP neighbor.
Step 6	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1`	Assigns the neighbor a remote autonomous system number.
Step 7	**description** *text* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# description neighbor 172.16.1.1`	(Optional) Provides a description of the neighbor. The description is used to save comments and does not affect software function.
Step 8	**password** { **clear** | **encrypted** } *password* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# password encrypted 123abc`	Enables Message Digest 5 (MD5) authentication on the TCP connection between the two BGP neighbors.
Step 9	**shutdown** **Example:**	Terminates any active sessions for the specified neighbor and removes all associated routing information.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp-nbr)# shutdown`	
Step 10	**timers** *keepalive hold-time* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# timers 12000 200`	Set the timers for the BGP neighbor.
Step 11	**update-source** *type interface-id* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source gigabitEthernet 0/1/5/0`	Allows iBGP sessions to use the primary IP address from a specific interface as the local address when forming an iBGP session with a neighbor.
Step 12	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpvn4 unicast`	Enters VPN neighbor address family configuration mode.
Step 13	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pe-pe-vpn-in in`	Specifies a routing policy for an inbound route. The policy can be used to filter routes or modify route attributes.
Step 14	**route-policy** *route-policy-name* **out** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pe-pe-vpn-out out`	Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.
Step 15	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Route Reflector to Hold Routes That Have a Defined Set of RT Communities

A provider edge (PE) needs to hold the routes that match the import route targets (RTs) of the VPNs configured on it. The PE router can discard all other VPNv4 routes. But, a route reflector (RR) must retain all VPNv4

routes, because it might peer with PE routers and different PEs might require different RT-tagged VPNv4 (making RRs non-scalable). You can configure an RR to only hold routes that have a defined set of RT communities. Also, a number of the RRs can be configured to service a different set of VPNs (thereby achieving some scalability). A PE is then made to peer with all RRs that service the VRFs configured on the PE. When a new VRF is configured with an RT for which the PE does not already hold routes, the PE issues route refreshes to the RRs and retrieves the relevant VPN routes.

Note Note that this process can be more efficient if the PE-RR session supports extended community outbound route filter (ORF).

Perform this task to configure a reflector to retain routes tagged with specific RTs.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family vpnv4 unicast**
4. **retain route-target** { **all** | **route-policy** *route-policy-name* }
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family vpnv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpvn4 unicast`	Enters VPN address family configuration mode.
Step 4	**retain route-target** { **all** | **route-policy** *route-policy-name* } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# retain route-target route-policy rr_ext-comm`	Configures a reflector to retain routes tagged with particular RTs. Use the *route-policy-name* argument for the policy name that lists the extended communities that a path should have in order for the RR to retain that path. **Note** The **all** keyword is not required, because this is the default behavior of a route reflector.

	Command or Action	Purpose
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP as a PE-CE Protocol

Perform this task to configure BGP on the PE and establish PE-CE communication using BGP. This task can be performed in both VRF and non-VRF configuration.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **bgp router-id** *ip-address*
5. **label mode per-ce**
6. **address-family** { **ipv4** | **ipv6** } **unicast**
7. **network** { *ip-address* / *prefix-length* | *ip-address mask* }
8. **aggregate-address** *address* / *mask-length*
9. **exit**
10. **neighbor** *ip-address*
11. **remote-as** *as-number*
12. **password** { **clear** | **encrypted** } *password*
13. **ebgp-multihop** [*ttl-value*]
14. Do one of the following:
    - **address-family** { **ipv4** | **ipv6** } **unicast**
    - **address-family {ipv4 {unicast | labeled-unicast} | ipv6 unicast}**
15. **site-of-origin** [*as-number* **:** *nn* | *ip-address* **:** *nn*]
16. **as-override**
17. **allowas-in** [*as-occurrence-number*]
18. **route-policy** *route-policy-name* **in**
19. **route-policy** *route-policy-name* **out**
20. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_pe_2`	Enables BGP routing for a particular VRF on the PE router.
Step 4	**bgp router-id** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# bgp router-id 172.16.9.9`	Configures a fixed router ID for a BGP-speaking router.
Step 5	**label mode per-ce** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# label mode per-ce`	• Configures the per-CE label mode to avoid an extra lookup on the PE router and conserve label space (per-prefix is the default label mode). In this mode, the PE router allocates one label for every immediate next-hop (in most cases, this would be a CE router). This label is directly mapped to the next hop, so there is no VRF route lookup performed during data forwarding. However, the number of labels allocated would be one for each CE rather than one for each VRF. Because BGP knows all the next hops, it assigns a label for each next hop (not for each PE-CE interface). When the outgoing interface is a multiaccess interface and the media access control (MAC) address of the neighbor is not known, Address Resolution Protocol (ARP) is triggered during packet forwarding. • The **per-vrf** keyword configures the same label to be used for all the routes advertised from a unique VRF.
Step 6	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

	Command or Action	Purpose
Step 7	**network** { *ip-address* / *prefix-length* | *ip-address mask* } **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# network 172.16.5.5/24`	Originates a network prefix in the address family table in the VRF context.
Step 8	**aggregate-address** *address* / *mask-length* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/24`	Configures aggregation in the VRF address family context to summarize routing information to reduce the state maintained in the core. This summarization introduces some inefficiency in the PE edge, because an additional lookup is required to determine the ultimate next hop for a packet. When configured, a summary prefix is advertised instead of a set of component prefixes, which are more specifics of the aggregate. The PE advertises only one label for the aggregate. Because component prefixes could have different next hops to CEs, an additional lookup has to be performed during data forwarding.
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# exit`	Exits the current configuration mode.
Step 10	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.0.0`	Configures a CE neighbor. The *ip-address* argument must be a private address.
Step 11	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2`	Configures the remote AS for the CE neighbor.
Step 12	**password** { **clear** | **encrypted** } *password* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# password encrypted 234xyz`	Enable Message Digest 5 (MD5) authentication on a TCP connection between two BGP neighbors.
Step 13	**ebgp-multihop** [*ttl-value*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop 55`	Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected.

	Command or Action	Purpose
Step 14	Do one of the following: • **address-family** { **ipv4** | **ipv6** } **unicast** • **address-family {ipv4 {unicast | labeled-unicast} | ipv6 unicast}** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 15	**site-of-origin** [*as-number* **:** *nn* | *ip-address* **:** *nn*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# site-of-origin 234:111`	Configures the site-of-origin (SoO) extended community. Routes that are learned from this CE neighbor are tagged with the SoO extended community before being advertised to the rest of the PEs. SoO is frequently used to detect loops when as-override is configured on the PE router. If the prefix is looped back to the same site, the PE detects this and does not send the update to the CE.
Step 16	**as-override** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# as-override`	Configures AS override on the PE router. This causes the PE router to replace the CE's ASN with its own (PE) ASN. **Note** This loss of information could lead to routing loops; to avoid loops caused by as-override, use it in conjunction with site-of-origin.
Step 17	**allowas-in** [*as-occurrence-number*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 5`	Allows an AS path with the PE autonomous system number (ASN) a specified number of times. Hub and spoke VPN networks need the looping back of routing information to the HUB PE through the HUB CE. When this happens, due to the presence of the PE ASN, the looped-back information is dropped by the HUB PE. To avoid this, use the **allowas-in** command to allow prefixes even if they have the PEs ASN up to the specified number of times.
Step 18	**route-policy** *route-policy-name* **in** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pe_ce_in_policy in`	Specifies a routing policy for an inbound route. The policy can be used to filter routes or modify route attributes.
Step 19	**route-policy** *route-policy-name* **out** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pe_ce_out_policy out`	Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.
Step 20	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Redistribution of IGPs to BGP

Perform this task to configure redistribution of a protocol into the VRF address family.

Even if Interior Gateway Protocols (IGPs) are used as the PE-CE protocol, the import logic happens through BGP. Therefore, all IGP routes have to be imported into the BGP VRF table.

Note Starting with Cisco IOS XR 7.4.1, redistribution of ISIS routes is supported on the VRF address family.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **address-family** { **ipv4** | **ipv6** } **unicast**
5. Do one of the following:
   - **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_a`	Enables BGP routing for a particular VRF on the PE router.
Step 4	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 5	Do one of the following: • **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute eigrp 23`	Configures redistribution of a protocol into the VRF address family context. The **redistribute** command is used if BGP is not used between the PE-CE routers. If BGP is used between PE-CE routers, the IGP that is used has to be redistributed into BGP to establish VPN connectivity with other PE sites. Redistribution is also required for inter-table import and export.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Keychains for BGP

Keychains provide secure authentication by supporting different MAC authentication algorithms and provide graceful key rollover. Perform this task to configure keychains for BGP. This task is optional.

Note If a keychain is configured for a neighbor group or a session group, a neighbor using the group inherits the keychain. Values of commands configured specifically for a neighbor override inherited values.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **keychain** *name*
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

	Command or Action	Purpose
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 5	**keychain** *name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# keychain kych_a`	Configures keychain-based authentication.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Disabling a BGP Neighbor

Perform this task to administratively shut down a neighbor session without removing the configuration.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **shutdown**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 127`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**shutdown** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# shutdown`	Disables all active sessions for the specified neighbor.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Neighbor Capability Suppression

A BGP speaker can learn about BGP extensions that are supported by a peer by using the capabilities negotiation feature. Capabilities negotiation allows BGP to use only the set of features supported by both BGP peers on a link. The neighbor capability suppression feature will turn off neighbor capabilities negotiation during Open message exchange. This is required for interoperability with very old customer premises equipment devices that do not understand Capabilities option.

Configuration:

Command introduced in neighbor, session-group and neighbor-group modes.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **capability suppress all**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 4`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**capability suppress all** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# capability suppress all`	Turn off neighbor capabilities.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

BGP Dynamic Neighbors

Earlier IOS-XR supported explicitly configured or static neighbor configuration. BGP dynamic neighbor support allows BGP peering to a group of remote neighbors that are defined by a range of IP addresses. Each range can be configured as a subnet IP address.

In larger BGP networks, implementing BGP dynamic neighbors can reduce the amount and complexity of CLI configuration and save CPU and memory usage. Both IPv4 and IPv6 peering are supported. Both IPv4 and IPv6 peering are supported.

Note The maximum number of remote neighbors for a single BGP dynamic neighbor subnet range is 4096. This is equivalent to a /20 subnet. You can configure multiple dynamic neighbor subnet ranges.

Configuring BGP Dynamic Neighbors using Address Range

The existing neighbor command is extended to accept a prefix instead of an address.

In the following task, Router B is configured as a remote BGP peer. After a subnet range is configured, a TCP session is initiated by Router B which has an IP address in the subnet range and a new BGP neighbor is dynamically established.

After the initial configuration of subnet ranges and activation of the peer neighbor, dynamic BGP neighbor creation does not require any further CLI configuration on the Router A.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *address prefix*
4. **remote-as** *as-number*
5. **update-source** *type interface-id*
6. **address-family ipv4 unicast**
7. Use the **commit** or **end** command.

DETAILED STEPS

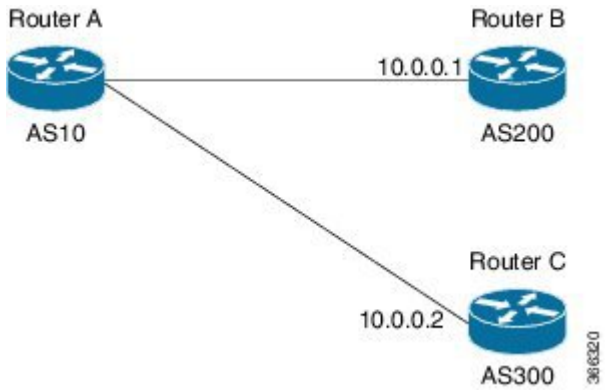
	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters the global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *address prefix* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.0/16`	Places the router in neighbor configuration mode for BGP routing and configures the BGP dynamic neighbor within the subnet range.

	Command or Action	Purpose
		Note All commands currently supported under a static neighbor, including address-family and inheritance using neighbor-group, session-group and af-group, will be supported for dynamic neighbor ranges with the exception of the following commands: • session-open-mode • local address
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1`	Creates a neighbor and assigns a remote autonomous system (AS) number to it.
Step 5	**update-source** *type interface-id* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source TenGige 0/0/05`	Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor. The type and interface-id arguments specify the type and ID number of the interface. Use the CLI help (?) to see a list of all the possible interface types and their ID numbers.
Step 6	**address-family ipv4 unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies the IPv4 unicast address family unicast and enters address family configuration mode.
Step 7	Use the **commit** or **end** command.	**commit** - Saves the configuration changes and remains within the configuration session. **end** - Prompts user to take one of these actions: • **Yes** - Saves configuration changes and exits the configuration session. • **No** - Exits the configuration session without committing the configuration changes. • **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Remote AS

In the following task, Router B and Router C are configured as a remote BGP peers. Both Router B and Router C are in different autonomous systems.

A list is created with the autonomous system of the remote routers and the list is then configured under neighbor mode using remote-as-list command.



SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **as-list** *name*
4. **neighbor** *address prefix*
5. **remote-as-list** *name*
6. **address-family ipv4 unicast**
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters the global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 10`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3	**as-list** *name*
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp)# as-list LIST-1`
	`RP/0/RSP0/CPU0:router(config-bgp-as-list)#200`
	`RP/0/RSP0/CPU0:router(config-bgp-as-list)#300`
	`RP/0/RSP0/CPU0:router(config-bgp-as-list)#end`
		Specifies a list of remote autonomous systems under BGP mode.
		Note The autonomous system numbers configured under as-list must be different from the router autonomous system number.

Step 4	**neighbor** *address prefix* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.0/16`	Places the router in neighbor configuration mode for BGP routing and configures the BGP dynamic neighbor within the subnet range.

	Command or Action	Purpose
Step 5	**remote-as-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as-list LIST-1`	Applies the configured as-list to the neighbor range. **Note** The remote-as and remote-as-list commands cannot be configured under a given mode at the same time. **Note** This command is applicable only to dynamic neighbor ranges and will be accepted under the following modes – neighbor, neighbor-group and session-group.

Step 6	**address-family ipv4 unicast**
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`
	`RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#`
		Specifies the IPv4 address family unicast and enters address family configuration mode.

Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Maximum-peers and Idle-watch timeout

In the below task, maximum-peers and idle-watch timeout commands are configured for a remote BGP peer.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *address prefix*
4. **maximum-peers** *number*
5. **idle-watch-time** *number*
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters the global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 10`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *address prefix* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.0/16`	Places the router in neighbor configuration mode for BGP routing and configures the BGP dynamic neighbor within the subnet range.
Step 4	**maximum-peers** *number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# maximum-peers 16`	This is used to configure an upper limit on the number of dynamic neighbor instances allowed under a range. Range for the maximum number of peers is 1 to 4095.
Step 5	**idle-watch-time** *number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# idle-watch-time 120`	Configures the time to wait before deleting an idle TCP instance.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Resetting Neighbors Using BGP Inbound Soft Reset

Perform this task to trigger an inbound soft reset of the specified address families for the specified group or neighbors. The group is specified by the ***** , *ip-address* , *as-number* , or **external** keywords and arguments.

Resetting neighbors is useful if you change the inbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates. If an inbound soft reset is triggered, BGP sends a REFRESH request to the neighbor if the neighbor has advertised the ROUTE_REFRESH capability. To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

SUMMARY STEPS

1. **show bgp neighbors**

2. **clear bgp** { **ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **ipv6 unicast** } { ***** | *ip-address* | **as** *as-number* | **external** } **soft** [**in** [**prefix-filter**] | **out**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp neighbors** **Example:** `RP/0/RSP0/CPU0:router# show bgp neighbors`	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	**clear bgp** { **ipv4** { **unicast**	**multicast**	**all**	**tunnel** }	**ipv6 unicast**	**all** { **unicast**	**multicast**	**all**	**tunnel** }	**vpnv4 unicast**	**vrf** { *vrf-name*	**all** } { **ipv4 unicast**	**ipv6 unicast** } { *****	*ip-address*	**as** *as-number*	**external** } **soft** [**in** [**prefix-filter**]	**out**] **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.1 soft in`	Soft resets a BGP neighbor. • The ***** keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset.

Resetting Neighbors Using BGP Outbound Soft Reset

Perform this task to trigger an outbound soft reset of the specified address families for the specified group or neighbors. The group is specified by the ***** , *ip-address* , *as-number* , or **external** keywords and arguments.

Resetting neighbors is useful if you change the outbound policy for the neighbors or any other configuration that affects the sending or receiving of routing updates.

If an outbound soft reset is triggered, BGP resends all routes for the address family to the given neighbors.

To determine whether the neighbor has advertised the ROUTE_REFRESH capability, use the **show bgp neighbors** command.

SUMMARY STEPS

1. **show bgp neighbors**
2. **clear bgp** { **ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **ipv6 unicast** } { ***** | *ip-address* | **as** *as-number* | **external** } **clear bgp** { **ipv4 | ipv6**} { **unicast** | **labeled-unicast** } **soft** [**in** [**prefix-filter**] |]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp neighbors** **Example:**	Verifies that received route refresh capability from the neighbor is enabled.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# show bgp neighbors`	
Step 2	**clear bgp** { **ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **ipv6 unicast** } { ***** | *ip-address* | **as** *as-number* | **external** } **clear bgp** { **ipv4 | ipv6**} { **unicast** | **labeled-unicast** } **soft** [**in** [**prefix-filter**] |] **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.2 soft out`	Soft resets a BGP neighbor. • The ***** keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset.

Resetting Neighbors Using BGP Hard Reset

Perform this task to reset neighbors using a hard reset. A hard reset removes the TCP connection to the neighbor, removes all routes received from the neighbor from the BGP table, and then re-establishes the session with the neighbor. If the **graceful** keyword is specified, the routes from the neighbor are not removed from the BGP table immediately, but are marked as stale. After the session is re-established, any stale route that has not been received again from the neighbor is removed.

SUMMARY STEPS

1. **clear bgp** { **ipv4** { **unicast** \| **multicast** \| **all** \| **tunnel** } \| **ipv6 unicast** \| **all** { **unicast** \| **multicast** \| **all** \| **tunnel** } \| **vpnv4 unicast** \| **vrf** { *vrf-name* \| **all** } { **ipv4 unicast** \| **ipv6 unicast** } \| { ***** \| *ip-address* \| **as** *as-number* \| **external** } [**graceful**] **soft** [**in** [**prefix-filter**] \| **out**] **clear bgp** { **ipv4 \| ipv6**} { **unicast** \| **labeled-unicast** }

DETAILED STEPS

	Command or Action	Purpose
Step 1	**clear bgp** { **ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **ipv6 unicast** } | { ***** | *ip-address* | **as** *as-number* | **external** } [**graceful**] **soft** [**in** [**prefix-filter**] | **out**] **clear bgp** { **ipv4 | ipv6**} { **unicast** | **labeled-unicast** } **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.3 graceful soft out`	Clears a BGP neighbor. • The ***** keyword resets all BGP neighbors. • The *ip-address* argument specifies the address of the neighbor to be reset. • The *as-number* argument specifies that all neighbors that match the autonomous system number be reset. • The **external** keyword specifies that all external neighbors are reset. The **graceful** keyword specifies a graceful restart.

Clearing Caches, Tables, and Databases

Perform this task to remove all contents of a particular cache, table, or database. The **clear bgp** command resets the sessions of the specified group of neighbors (hard reset); it removes the TCP connection to the neighbor, removes all routes received from the neighbor from the BGP table, and then re-establishes the session with the neighbor. Clearing a cache, table, or database can become necessary when the contents of the particular structure have become, or are suspected to be, invalid.

SUMMARY STEPS

1. **clear bgp** { **ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **ipv6 unicast** } *ip-address*
2. **clear bgp external**
3. **clear bgp ***

DETAILED STEPS

	Command or Action	Purpose
Step 1	**clear bgp** { **ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6 unicast** | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } { **ipv4 unicast** | **ipv6 unicast** } *ip-address* **Example:** `RP/0/RSP0/CPU0:router# clear bgp ipv4 172.20.1.1`	Clears a specified neighbor.
Step 2	**clear bgp external** **Example:** `RP/0/RSP0/CPU0:router# clear bgp external`	Clears all external peers.
Step 3	**clear bgp *** **Example:** `RP/0/RSP0/CPU0:router# clear bgp *`	Clears all BGP neighbors.

Displaying System and Network Statistics

Perform this task to display specific statistics, such as the contents of BGP routing tables, caches, and databases. Information provided can be used to determine resource usage and solve network problems. You can also display information about node reachability and discover the routing path that the packets of your device are taking through the network.

SUMMARY STEPS

1. **show bgp cidr-only**
2. **show bgp community** *community-list* [**exact-match**]
3. **show bgp regexp** *regular-expression*

4. **show bgp**
5. **show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**]
6. **show bgp paths**
7. **show bgp neighbor-group** *group-name* **configuration**
8. **show bgp summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp cidr-only** **Example:** `RP/0/RSP0/CPU0:router# show bgp cidr-only`	Displays routes with nonnatural network masks (classless interdomain routing [CIDR]) routes.
Step 2	**show bgp community** *community-list* [**exact-match**] **Example:** `RP/0/RSP0/CPU0:router# show bgp community 1081:5 exact-match`	Displays routes that match the specified BGP community.
Step 3	**show bgp regexp** *regular-expression* **Example:** `RP/0/RSP0/CPU0:router# show bgp regexp "^3 "`	Displays routes that match the specified autonomous system path regular expression.
Step 4	**show bgp** **Example:** `RP/0/RSP0/CPU0:router# show bgp`	Displays entries in the BGP routing table.
Step 5	**show bgp neighbors** *ip-address* [**advertised-routes** | **dampened-routes** | **flap-statistics** | **performance-statistics** | **received** *prefix-filter* | **routes**] **Example:** `RP/0/RSP0/CPU0:router# show bgp neighbors 10.0.101.1`	Displays information about the BGP connection to the specified neighbor. • The **advertised-routes** keyword displays all routes the router advertised to the neighbor. • The **dampened-routes** keyword displays the dampened routes that are learned from the neighbor. • The **flap-statistics** keyword displays flap statistics of the routes learned from the neighbor. • The **performance-statistics** keyword displays performance statistics relating to work done by the BGP process for this neighbor. • The **received** *prefix-filter* keyword and argument display the received prefix list filter. • The **routes** keyword displays routes learned from the neighbor.

	Command or Action	Purpose
Step 6	**show bgp paths** **Example:** `RP/0/RSP0/CPU0:router# show bgp paths`	Displays all BGP paths in the database.
Step 7	**show bgp neighbor-group** *group-name* **configuration** **Example:** `RP/0/RSP0/CPU0:router# show bgp neighbor-group group_1 configuration`	Displays the effective configuration for a specified neighbor group, including any configuration inherited by this neighbor group.
Step 8	**show bgp summary** **Example:** `RP/0/RSP0/CPU0:router# show bgp summary`	Displays the status of all BGP connections.

Displaying BGP Process Information

Perform this task to display specific BGP process information.

SUMMARY STEPS

1. **show bgp process**
2. **show bgp ipv4 unicast summary**
3. **show bgp vpnv4 unicast summary**
4. **show bgp vrf** { *vrf-name* | **all** }
5. **show bgp process detail**
6. **show bgp summary**
7. **show placement program bgp**
8. **show placement program brib**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp process** **Example:** `RP/0/RSP0/CPU0:router# show bgp process`	Displays status and summary information for the BGP process. The output shows various global and address family-specific BGP configurations. A summary of the number of neighbors, update messages, and notification messages sent and received by the process is also displayed.
Step 2	**show bgp ipv4 unicast summary** **Example:** `RP/0/RSP0/CPU0:router# show bgp ipv4 unicast summary`	Displays a summary of the neighbors for the IPv4 unicast address family.

	Command or Action	Purpose
Step 3	**show bgp vpnv4 unicast summary** **Example:** `RP/0/RSP0/CPU0:router# show bgp vpnv4 unicast summary`	Displays a summary of the neighbors for the VPNv4 unicast address family.
Step 4	**show bgp vrf** { *vrf-name* | **all** } **Example:** `RP/0/RSP0/CPU0:router# show bgp vrf vrf_A`	Displays BGP VPN virtual routing and forwarding (VRF) information.
Step 5	**show bgp process detail** **Example:** `RP/0/RSP0/CPU0:router# show bgp processes detail`	Displays detailed process information including the memory used by each of various internal structure types.
Step 6	**show bgp summary** **Example:** `RP/0/RSP0/CPU0:router# show bgp summary`	Displays the status of all BGP connections.
Step 7	**show placement program bgp** **Example:** `RP/0/RSP0/CPU0:router# show placement program bgp`	Displays BGP program information. • If a program is shown as having 'rejected locations' (for example, locations where program cannot be placed), the locations in question can be viewed using the **show placement program bgp** command. • If a program has been placed but not started, the amount of elapsed time since the program was placed is displayed in the Waiting to start column.
Step 8	**show placement program brib** **Example:** `RP/0/RSP0/CPU0:router# show placement program brib`	Displays bRIB program information. • If a program is shown as having 'rejected locations' (for example, locations where program cannot be placed), the locations in question can be viewed using the **show placement program bgp** command. • If a program has been placed but not started, the amount of elapsed time since the program was placed is displayed in the Waiting to start column.

Monitoring BGP Update Groups

This task displays information related to the processing of BGP update groups.

SUMMARY STEPS

1. **show bgp** [**ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6** { **unicast** | **all** } | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } [**ipv4 unicast**] **update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp** [**ipv4** { **unicast** | **multicast** | **all** | **tunnel** } | **ipv6** { **unicast** | **all** } | **all** { **unicast** | **multicast** | **all** | **tunnel** } | **vpnv4 unicast** | **vrf** { *vrf-name* | **all** } [**ipv4 unicast**] **update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]] **Example:** `RP/0/RSP0/CPU0:router# show bgp update-group 0.0`	Displays information about BGP update groups. • The *ip-address* argument displays the update groups to which that neighbor belongs. • The *process-id.index* argument selects a particular update group to display and is specified as follows: process ID (dot) index. Process ID range is from 0 to 254. Index range is from 0 to 4294967295. • The **summary** keyword displays summary information for neighbors in a particular update group. • If no argument is specified, this command displays information for all update groups (for the specified address family). • The **performance-statistics** keyword displays performance statistics for an update group.

Configuring BGP Nonstop Routing

BGP Nonstop Routing (BGP NSR) is enabled by default. The **no nsr disable** command can also be used to turn BGP NSR back on if it has been disabled.

Disable BGP Nonstop Routing

Perform this task to disable BGP Nonstop Routing (NSR):

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **nsr disable**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number, and enters the BGP configuration mode, for configuring BGP routing processes.
Step 3	**nsr disable** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# nsr disable`	Disables BGP Nonstop routing.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Re-enable BGP Nonstop Routing

If BGP Nonstop Routing (NSR) is disabled, use the following steps to turn BGP NSR back on using the following steps:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **no nsr disable**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the BGP AS number, and enters the BGP configuration mode, for configuring BGP routing processes.
Step 3	**no nsr disable** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# nsr disable`	Enables BGP Nonstop routing.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Installing Primary Backup Path for Prefix Independent Convergence (PIC)

Perform the following tasks to install a backup path into the forwarding table and provide prefix independent convergence (PIC) in case of a PE-CE link failure:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. Do one of the following
   - **address-family** {**vpnv4 unicast** | **vpnv6 unicast**}
   - **vrf** *vrf-name* {**ipv4 unicast** | **ipv6 unicast**}
4. **additional-paths selection route-policy** *route-policy-name*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	Do one of the following • **address-family** {**vpnv4 unicast** | **vpnv6 unicast**} • **vrf** *vrf-name* {**ipv4 unicast** | **ipv6 unicast**} **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Specifies the address family or VRF address family and enters the address family or VRF address family configuration submode.

Step 4	**additional-paths selection route-policy** *route-policy-name*
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp-af)# additional-paths selection route-policy ap1`
		Configures additional paths selection mode for a prefix.
		Note Use the **additional-paths selection** command with an appropriate route-policy to calculate backup paths and to enable Prefix Independent Convergence (PIC) functionality.
		The route-policy configuration is a pre-requisite for configuring the additional-paths selection mode for a prefix . This is an example route-policy configuration to use with additional-selection command:
		`route-policy ap1`
		`  set path-selection backup 1 install`
		`end-policy`

Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Retaining Allocated Local Label for Primary Path

Perform the following tasks to retain the previously allocated local label for the primary path on the primary PE for some configurable time after reconvergence:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family { vpnv4 unicast | vpnv6 unicast }**
4. **retain local-label** *minutes*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family { vpnv4 unicast | vpnv6 unicast }** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast`	Specifies the address family and enters the address family configuration submode.
Step 4	**retain local-label** *minutes* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# retain local-label 10`	Retains the previously allocated local label for the primary path on the primary PE for 10 minutes after reconvergence.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Additional Paths

Perform these tasks to configure BGP Additional Paths capability:

SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **if** *conditional-expression* **then** *action-statement* **else**
4. **pass endif**
5. **end-policy**
6. **router bgp** *as-number*
7. **as-league peers** *as-number*
8. **address-family** {**ipv4** {**unicast** | **multicast**} | **ipv6** {**unicast** | **multicast** | **l2vpn vpls-vpws** | **vpnv4 unicast** | **vpnv6 unicast** }
9. **additional-paths receive**
10. **additional-paths send**
11. **additional-paths selection route-policy** *route-policy-name*
12. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy** *route-policy-name* **Example:** `RP/0/RSP0/CPU0:router (config)#route-policy add_path_policy`	Defines the route policy and enters route-policy configuration mode.

Step 3	**if** *conditional-expression* **then** *action-statement* **else**
	Example:
	`RP/0/RSP0/CPU0:router (config-rpl)#if community matches-any (*) then`
	`  set path-selection all advertise`
	`else`
		Decides the actions and dispositions for the given route.

Step 4	**pass endif**
	Example:
	`RP/0/RSP0/CPU0:router(config-rpl-else)#pass`
	`RP/0/RSP0/CPU0:router(config-rpl-else)#endif`
		Passes the route for processing and ends the if statement.

Step 5	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)#end-policy`	Ends the route policy definition of the route policy and exits route-policy configuration mode.

	Command or Action	Purpose
Step 6	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)#router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 7	**as-league peers** *as-number*
	Example:
	`RP/0/RSP0/CPU0:router(config-bgp)#as-league peers`
	`RP/0/RSP0/CPU0:router(config-bgp-as-league-peers)# 6200`
		Configures a group of peer autonomous systems (AS) that will be used under common administration or a trusted relationship.

Step 8	**address-family** {**ipv4** {**unicast** | **multicast**} | **ipv6** {**unicast** | **multicast** | **l2vpn vpls-vpws** | **vpnv4 unicast** | **vpnv6 unicast** } **Example:** `RP/0/RSP0/CPU0:router(config-bgp)#address-family ipv4 unicast`	Specifies the address family and enters address family configuration submode.
Step 9	**additional-paths receive** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)#additional-paths receive`	Configures receive capability of multiple paths for a prefix to the capable peers.
Step 10	**additional-paths send** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)#additional-paths send`	Configures send capability of multiple paths for a prefix to the capable peers .
Step 11	**additional-paths selection route-policy** *route-policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)#additional-paths selection route-policy add_path_policy`	Configures additional paths selection capability for a prefix.
Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring iBGP Multipath Load Sharing

Perform this task to configure the iBGP Multipath Load Sharing:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** {**ipv4**|**ipv6**} {**unicast**|**multicast**}
4. **maximum-paths ibgp** *number*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** {**ipv4**|**ipv6**} {**unicast**|**multicast**} **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 multicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
Step 4	**maximum-paths ibgp** *number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths ibgp 30`	Configures the maximum number of iBGP paths for load sharing.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Originating Prefixes with AiGP

Perform this task to configure origination of routes with the AiGP metric:

Before you begin

Origination of routes with the accumulated interior gateway protocol (AiGP) metric is controlled by configuration. AiGP attributes are attached to redistributed routes that satisfy following conditions:

- The protocol redistributing the route is enabled for AiGP.
- The route is an interior gateway protocol (iGP) route redistributed into border gateway protocol (BGP). The value assigned to the AiGP attribute is the value of iGP next hop to the route or as set by a route-policy.
- The route is a static route redistributed into BGP. The value assigned is the value of next hop to the route or as set by a route-policy.
- The route is imported into BGP through network statement. The value assigned is the value of next hop to the route or as set by a route-policy.

SUMMARY STEPS

1. **configure**
2. **route-policy** *aigp_policy*
3. **set aigp-metricigp-cost**
4. **exit**
5. **router bgp** *as-number*
6. **address-family** {**ipv4** | **ipv6**} **unicast**
7. **redistribute ospf** *osp* **route-policy** *plcy_name***metric** *value*
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy** *aigp_policy* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy aip_policy`	Enters route-policy configuration mode and sets the route-policy
Step 3	**set aigp-metricigp-cost** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# set aigp-metric igp-cost`	Sets the internal routing protocol cost as the aigp metric.

	Command or Action	Purpose
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# exit`	Exits route-policy configuration mode.
Step 5	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 6	**address-family** {**ipv4** | **ipv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.
Step 7	**redistribute ospf** *osp* **route-policy** *plcy_name***metric** *value* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)#redistribute ospf osp route-policy aigp_policy metric 1`	Allows the redistribution of AiBGP metric into OSPF.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring BGP Accept Own

Perform this task to configure BGP Accept Own:

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **update-source** *type interface-path-id*
6. **address-family** {*vpnv4 unicast* | *vpnv6 unicast*}
7. **accept-own** [**inheritance-disable**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `Router(config)#router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**neighbor** *ip-address* **Example:** `Router(config-bgp)#neighbor 10.1.2.3`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**remote-as** *as-number* **Example:** `Router(config-bgp-nbr)#remote-as 100`	Assigns a remote autonomous system number to the neighbor.
Step 5	**update-source** *type interface-path-id* **Example:** `Router(config-bgp-nbr)#update-source Loopback0`	Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.
Step 6	**address-family** {*vpnv4 unicast* | *vpnv6 unicast*} **Example:** `Router(config-bgp-nbr)#address-family vpnv6 unicast`	Specifies the address family as VPNv4 or VPNv6 and enters neighbor address family configuration mode.
Step 7	**accept-own** [**inheritance-disable**] **Example:** `Router(config-bgp-nbr-af)#accept-own`	Enables handling of self-originated VPN routes containing Accept_Own community. Use the **inheritance-disable** keyword to disable the "accept own" configuration and to prevent inheritance of "acceptown" from a parent configuration.

Configuring BGP Permanent Network

Configuring BGP Permanent Network

Perform this task to configure BGP permanent network. You must configure at least one route-policy to identify the set of prefixes (networks) for which the permanent network (path) is to be configured.

SUMMARY STEPS

1. **configure**
2. **prefix-set** *prefix-set-name*
3. **exit**

4. **route-policy** *route-policy-name*
5. **end-policy**
6. **router bgp** *as-number*
7. **address-family { ipv4 | ipv6 } unicast**
8. **permanent-network route-policy** *route-policy-name*
9. Use the **commit** or **end** command.
10. **show bgp {ipv4 | ipv6} unicast** *prefix-set*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**prefix-set** *prefix-set-name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# prefix-set PERMANENT-NETWORK-IPv4`
	`RP/0/RSP0/CPU0:router(config-pfx)# 1.1.1.1/32,`
	`RP/0/RSP0/CPU0:router(config-pfx)# 2.2.2.2/32,`
	`RP/0/RSP0/CPU0:router(config-pfx)# 3.3.3.3/32`
	`RP/0/RSP0/CPU0:router(config-pfx)# end-set`
		Enters prefix set configuration mode and defines a prefix set for contiguous and non-contiguous set of bits.

Step 3	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pfx)# exit`	Exits prefix set configuration mode and enters global configuration mode.

Step 4	**route-policy** *route-policy-name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# route-policy POLICY-PERMANENT-NETWORK-IPv4`
	`RP/0/RSP0/CPU0:router(config-rpl)# if destination in PERMANENT-NETWORK-IPv4 then`
	`RP/0/RSP0/CPU0:router(config-rpl)# pass`
	`RP/0/RSP0/CPU0:router(config-rpl)# endif`
		Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 5	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	Ends the definition of a route policy and exits route policy configuration mode.

	Command or Action	Purpose
Step 6	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode.
Step 7	**address-family { ipv4 | ipv6 } unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.
Step 8	**permanent-network route-policy** *route-policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# permanent-network route-policy POLICY-PERMANENT-NETWORK-IPv4`	Configures the permanent network (path) for the set of prefixes as defined in the route-policy.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show bgp {ipv4 | ipv6} unicast** *prefix-set* **Example:** `RP/0/RSP0/CPU0:routershow bgp ipv4 unicast`	(Optional) Displays whether the prefix-set is a permanent network in BGP.

How to Advertise Permanent Network

Perform this task to identify the peers to whom the permanent paths must be advertised.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*

5. **address-family { ipv4 | ipv6 } unicast**
6. **advertise permanent-network**
7. Use the **commit** or **end** command.
8. **show bgp {ipv4 | ipv6} unicast neighbor** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.255.255.254`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 4713`	Assigns the neighbor a remote autonomous system number.
Step 5	**address-family { ipv4	ipv6 } unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.
Step 6	**advertise permanent-network** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise permanent-network`	Specifies the peers to whom the permanent network (path) is advertised.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**show bgp {ipv4 | ipv6} unicast neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:routershow bgp ipv4 unicast neighbor 10.255.255.254`	(Optional) Displays whether the neighbor is capable of receiving BGP permanent networks.

Enabling BGP Unequal Cost Recursive Load Balancing

Perform this task to enable unequal cost recursive load balancing for external BGP (eBGP), interior BGP (iBGP), and eiBGP and to enable BGP to carry link bandwidth attribute of the demilitarized zone (DMZ) link.

When the PE router includes the link bandwidth extended community in its updates to the remote PE through the Multiprotocol Interior BGP (MP-iBGP) session (either IPv4 or VPNv4), the remote PE automatically does load balancing if the **maximum-paths** command is enabled.

Unequal cost recursive load balancing happens across maximum eight paths only.

Note Enabling BGP unequal cost recursive load balancing feature is not supported on CPP based cards.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } **unicast**
4. **maximum-paths** { **ebgp** | **ibgp** | **eibgp** } *maximum* [**unequal-cost**]
5. **exit**
6. **neighbor** *ip-address*
7. **dmz-link-bandwidth**
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.
Step 3	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode. To see a list of all the possible keywords and arguments for this command, use the CLI help (?).
Step 4	**maximum-paths** { **ebgp** | **ibgp** | **eibgp** } *maximum* [**unequal-cost**] **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# maximum-paths ebgp 3`	Configures the maximum number of parallel routes that BGP installs in the routing table. **Note** • Valid values for maximum-paths are 8 for ASR 9000 Ethernet Line Card and 32 for ASR 9000 Enhanced Ethernet Line Card. • The ASR 9000 Ethernet Line Card limits the number of routes to be installed to 8 in the forwarding hardware even though the maximum-path value configured is more than 8. • **ebgp** *maximum* : Consider only eBGP paths for multipath. • **ibgp** *maximum* [**unequal-cost**]: Consider load balancing between iBGP learned paths. • **eibgp** *maximum* : Consider both eBGP and iBGP learned paths for load balancing. eiBGP load balancing always does unequal-cost load balancing. When eiBGP is applied, eBGP or iBGP load balancing cannot be configured; however, eBGP and iBGP load balancing can coexist.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 6	**neighbor** *ip-address* **Example:**	Configures a CE neighbor. The *ip-address* argument must be a private address.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.0`	
Step 7	**dmz-link-bandwidth** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# dmz-link-bandwidth`	Originates a demilitarized-zone (DMZ) link-bandwidth extended community for the link to an eBGP/iBGP neighbor.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring VRF Dynamic Route Leaking

Perform these steps to import routes from default-VRF to non-default VRF or to import routes from non-default VRF to default VRF.

Before you begin

A route-policy is mandatory for configuring dynamic route leaking. Use the **route-policy** *route-policy-name* command in global configuration mode to configure a route-policy.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf_name*
3. **address-family** {**ipv4** | **ipv6**} **unicast**
4. Use one of these options:
   - **import from default-vrf route-policy** *route-policy-name* [**advertise-as-vpn**]
   - **export to default-vrf route-policy** *route-policy-name*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**vrf** *vrf_name* **Example:** `RP/0/RSP0/CPU0:PE51_ASR-9010(config)#vrf vrf_1`	Enters VRF configuration mode.
Step 3	**address-family** {**ipv4** | **ipv6**} **unicast** **Example:** `RP/0/RSP0/CPU0:router(config-vrf)#address-family ipv6 unicast`	Enters VRF address-family configuration mode.

Step 4	Use one of these options:
	• **import from default-vrf route-policy** *route-policy-name* [**advertise-as-vpn**]
	• **export to default-vrf route-policy** *route-policy-name*
	Example:
	`RP/0/RSP0/CPU0:router(config-vrf-af)#import from default-vrf route-policy rpl_dynamic_route_import`
	or
	`RP/0/RSP0/CPU0:router(config-vrf-af)#export to default-vrf route-policy rpl_dynamic_route_export`
		Imports routes from default-VRF to non-default VRF or from non-default VRF to default-VRF.
		• **import from default-vrf**—configures import from default-VRF to non-default-VRF.
		If the **advertise-as-vpn** option is configured, the paths imported from the default-VRF to the non-default-VRF are advertised to the PEs as well as to the CEs. If the **advertise-as-vpn** option is not configured, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PE. However, the paths are still advertised to the CEs.
		• **export to default-vrf**—configures import from non-default-VRF to default VRF. The paths imported from the default-VRF are advertised to other PEs.

Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

These **show bgp** command output displays information from the dynamic route leaking configuration:

- Use the **show bgp prefix** command to display the source-RD and the source-VRF for imported paths, including the cases when IPv4 or IPv6 unicast prefixes have imported paths.
- Use the **show bgp imported-routes** command to display IPv4 unicast and IPv6 unicast address-families under the default-VRF.

Enabling Selective VRF Download

To enable selective VRF download, configure the **svd platform enable** command followed by router reload.

Note Selective VRF download is disabled by default.

SUMMARY STEPS

1. **admin**
2. **configure**
3. **svd platform enable**
4. Use the **commit** or **end** command.
5. **show svd state**
6. **admin**
7. **reload location all**
8. **exit**
9. **show svd role**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**admin** **Example:** `RP/0/RSP0/CPU0:router# admin`	Enters administration EXEC mode.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router(admin)#configure`	Enters Administrative configuration mode.
Step 3	**svd platform enable** **Example:** `RP/0/RSP0/CPU0:router(admin-config)#svd platform enable`	Enables selective VRF download.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5	**show svd state**
	Example:
	`RP/0/RSP0/CPU0:router#show svd state`
	`Selective VRF Download (SVD) Feature State:`
	`SVD Configuration State Enabled`
	`SVD Operational State Enabled`
		Displays Selective VRF download feature state information.

Step 6	**admin** **Example:** `RP/0/RSP0/CPU0:router#admin`	Enters administrator mode.

Step 7	**reload location all**
	Example:
	`RP/0/RSP0/CPU0:router(admin)#reload loc all`
	`Tue Feb 12 07:51:25.279 UTC`
	
	`Preparing system for backup. This may take a few minutes especially for large configurations.`
	`Status report: node0_RSP0_CPU0: START TO BACKUP`
	`Status report: node0_RSP0_CPU0: BACKUP HAS COMPLETED SUCCESSFULLY`
	`[Done]`
	`Proceed with reload? [confirm]RP/0/RSP0/CPU0::This node received reload`
		Reloads the chassis.

Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(admin)#exit`	Exits administrator mode and enters EXEC mode.

Step 9	**show svd role**
	Example:
	`RP/0/RSP0/CPU0:router#show svd role`
	`Tue Feb 12 07:50:26.908 UTC`
	
	`Codes: (C) : user Configured role`
	`Node Name IPv4 Role IPv6 Role`
	`---`
	`0/RSP0/CPU0 Standard Standard`
	`0/0/CPU0 Customer Facing Not Interested`
	`0/1/CPU0 Customer Facing Not Interested`
		Verifies if selective VRF download is active by confirming that svd roles are "customer facing" for the line cards that have VRF interfaces on them.

What to do next

After enabling SVD using the **svd platform enable** command, do not use the **selective-vrf-download disable** to turn off SVD.

Disabling Selective VRF Download

Selective VRF Download is disabled by default. However, if the SVD is enabled, perform these tasks to disable the functionality.

SUMMARY STEPS

1. **admin**
2. **configure**
3. **no svd platform enable**
4. Use the **commit** or **end** command.
5. **show svd state**
6. **admin**
7. **reload location all**
8. **exit**
9. **show svd role**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**admin** **Example:** `RP/0/RSP0/CPU0:router# admin`	Enters administration EXEC mode.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router(admin)#configure`	Enters Administrative configuration mode.
Step 3	**no svd platform enable** **Example:** `RP/0/RSP0/CPU0:PE51_ASR-9010(admin-config)#no svd platform enable`	Disables Selective VRF Download.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 5	**show svd state**
	Example:
	`RP/0/RSP0/CPU0:router#show svd state`
	`Selective VRF Download (SVD) Feature State:`
	`SVD Configuration State Unsupported`
	`SVD Operational State Unsupported`
		Displays Selective VRF download feature state information.

Step 6	**admin** **Example:** `RP/0/RSP0/CPU0:router#admin`	Enters administrator mode.

Step 7	**reload location all**
	Example:
	`RP/0/RSP0/CPU0:router(admin)#reload loc all`
	`Tue Feb 12 07:51:25.279 UTC`
	
	`Preparing system for backup. This may take a few minutes especially for large configurations.`
	`Status report: node0_RSP0_CPU0: START TO BACKUP`
	`Status report: node0_RSP0_CPU0: BACKUP HAS COMPLETED SUCCESSFULLY`
	`[Done]`
	`Proceed with reload? [confirm]RP/0/RSP0/CPU0::This node received reload`
		Reloads the chassis.

Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(admin)#exit`	Exits administrator mode and enters EXEC mode.

Step 9	**show svd role**
	Example:
	`RP/0/RSP0/CPU0:router#show svd role`
	
	`Codes: (C) : user Configured role`
	`Node Name IPv4 Role IPv6 Role`
	`0/RSP0/CPU0 Standard Standard`
	`0/0/CPU0 Standard Standard`
	`0/1/CPU0 Standard Standard`
		Verifies if selective VRF download is inactive by confirming that svd roles are "standard" for the line cards that have VRF interfaces on them.

Configuring Resilient Per-CE Label Mode

Configuring Resilient Per-CE Label Mode Under VRF Address Family

Perform this task to configure resilient per-ce label mode under VRF address family.

Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**
2. **router bgp***as-number*
3. **vrf***vrf-instance*
4. **address-family {ipv4 | ipv6} unicast**
5. **label mode per-ce**
6. Do one of the following:
   - **end**
   - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)#
```

Enters global configuration mode.

Step 2 **router bgp***as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 666
RP/0/RSP0/CPU0:router(config-bgp)#
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **vrf***vrf-instance*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf-pe
RP/0/RSP0/CPU0:router(config-bgp-vrf)#
```

Configures a VRF instance.

Step 4 **address-family {ipv4 | ipv6} unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#
```

Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

Step 5 **label mode per-ce**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# label mode per-ce
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)#
```

Configures resilient per-ce label mode.

Step 6 Do one of the following:

- **end**
- **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# end
```

or

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Resilient Per-CE Label Mode Using a Route-Policy

Perform this task to configure resilient per-ce label mode using a route-policy.

Note Resilient per-CE 6PE label allocation is not supported on CRS-1 and CRS-3 routers, but supported only on ASR 9000 routers.

SUMMARY STEPS

1. **configure**

2. **route-policy***policy-name*
3. **set label mode per-ce**
4. Do one of the following:
   - **end**
   - **commit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)#
```

Enters global configuration mode.

Step 2 **route-policy***policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# route-policy route1
RP/0/RSP0/CPU0:router(config-rpl)#
```

Creates a route policy and enters route policy configuration mode.

Step 3 **set label mode per-ce**

Example:

```
RP/0/RSP0/CPU0:router(config-rpl)# set label mode per-ce
RP/0/RSP0/CPU0:router(config-rpl)#
```

Configures resilient per-ce label mode.

Step 4 Do one of the following:

- **end**
- **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-rpl)# end
```

or

```
RP/0/RSP0/CPU0:router(config-rpl)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Exclusion of Label Allocation for Non-Advertised Routes

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
Exclusion of Label Allocation for Non-Advertised Routes	Release 7.10.1	We have enabled better label space management and hardware resource utilization by making MPLS label allocation more flexible. This flexibility means you can now assign these labels to only those routes that are advertised to their peer routes, ensuring better label space management and hardware resource utilization. Prior to this release, label allocation was done regardless of whether the routes being advertised. This resulted in inefficient use of label space.

The functionality to control label allocation to the routes which are not advertised to peers is introduced. You can now choose to assign labels to the routes which are advertised to the peers.

Provider Edge (PE) routers works as autonomous systems border routers (ASBRs) where this feature is configured.

You can set the **community** attribute to either **no-advertise** or **no-export** in route-policy configuration mode to the routes which are not going to be advertised to peers. Once the **community** attribute in the route-policy is updated, the router doesn't allocate any label to those routes.

Note **no-export** is only for eBGP and **no-advertise** can be used for both eBGP and iBGP.

How to exclude label allocation for non-advertised routes

Configuration Example

This example shows how to set the *community* parameter to **no-advertise** for the routes which are not going to be advertised to any peer routes.

```
/*Configure the community set*/
Router(config)#community-set no-advertise
Router(config-comm)#no-advertise
Router(config-comm)#end-set

/*Configure the route policy*/
Router(config)#route-policy set-no-advertise
Router(config-rpl)#set community no-advertise additive
Router(config-rpl)#end-policy
Router(config-bgp-af)#route-policy pass_all
Router(config-rpl)#  pass
Router(config-rpl)#end-policy
Router(config)#route-policy pass_all
Router(config-rpl)#  pass
Router(config-rpl)#end-policy

/*Apply the route policy as inbound route policy*/
Router(config)#router bgp 1
Router(config-bgp)# neighbor 192.0.2.1
Router(config-bgp-nbr)#  remote-as 1
Router(config-bgp-nbr)#  update-source Loopback0
Router(config-bgp-nbr)#  address-family ipv4 unicast
Router(config-bgp-nbr-af)#   route-policy set-no-advertise in
Router(config-bgp-nbr-af)#   route-policy pass_all out
Router(config-bgp-nbr-af)#commit
```

Running Configuration

```
community-set no-advertise
  no-advertise
end-set
  !
!
route-policy set-no-advertise
  set community no-advertise additive
end-policy
  !
!
route-policy pass_all
  pass
end-policy
!
```

Verification

Use **show bgp vpnv6 unicast rd** command to verify the **community** parameter is set to **no-advertised**.

```
Router(config)# show bgp vpnv6 unicast rd 2001:DB8:0:ABCD::1

BGP routing table entry for 0:ABCD::1 Route Distinguisher: 2001:DB8
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               19207        19207
Paths: (1 available, best #1, not advertised to any peer)
  Not advertised to any peer
```

```
 Path #1: Received by speaker 0
 Not advertised to any peer
 Local, (Received from a RR-client)
   192.0.2.254 from 192.0.2.1 (192.0.2.1)
     Received Label 16
     Origin IGP, metric 3, localpref 3, aigp metric 3, valid, internal, best, group-best,
import-candidate, not-in-vrf
     Received Path ID 0, Local Path ID 1, version 19207
     Community: 1:1 no-advertise
     Extended community: Color:3333 RT:2001:DB8
     AIGP set by inbound policy metric
     Total AIGP metric 3
```

Configuring BGP Large Communities

BGP communities provide a way to group destinations and apply routing decisions such as acceptance, rejection, preference, or redistribution on a group of destinations using community attributes. BGP community attributes are variable length attributes consisting of a set of one or more 4-byte values which are split into two parts of 16 bits. The higher-order 16 bits represents the AS number and the lower order bits represents a locally defined value assigned by the operator of the AS.

Since the adoption of 4-byte ASNs (RFC6793), the BGP communities attribute can no longer accommodate the 4 byte ASNs as you need more than 4 bytes to encode the 4-byte ASN and an AS specific value that you want to tag with the route. Although BGP extended community permits a 4-byte AS to be encoded as the global administrator field, the local administrator field has only 2-byte of available space. So, 6-byte extended community attribute is also unsuitable. To overcome this limitation, you can configure a 12-byte BGP large community which is an optional attribute that provides the most significant 4-byte value to encode autonomous system number as the global administrator and the remaining two 4-byte assigned numbers to encode the local values.

Similar to BGP communities, routers can apply BGP large communities to BGP routes by using route policy languages (RPL) and other routers can then perform actions based on the community that is attached to the route. The policy language provides sets as a container for groups of values for matching purposes.

When large communities are specified in other commands, they are specified as three non negative decimal integers separated by colons. For example, 1:2:3. Each integer is stored in 32 bits. The possible range for each integer is 0 to 4294967295.

In route-policy statements, each integer in the BGP large community can be replaced by any of the following expressions :

- [x..y] — This expression specifies a range between x and y, inclusive.
- * —This expression stands for any number.
- peeras — This expression is replaced by the AS number of the neigbhor from which the community is received or to which the community is sent, as appropriate.
- not-peeras —This expression matches any number other than the peeras.
- private-as — This expression specifies any number in the private ASN range: [64512..65534] and [4200000000..4294967294].

These expressions can be also used in policy-match statements.

IOS regular expression (ios-regex) and DFA style regular expression (dfa-regex) can be used in any of the large-community policy match and delete statements. For example, the IOS regular expression ios-regex '^5:.*:7$' is equivalent to the expression 5:*:7.

The **send-community-ebgp** command is extended to include BGP large communities. This command is required for the BGP speaker to send large communities to ebgp neighbors.

For more information about BGP communities, extended communities, and route policy language, see the following link: https://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k-r6-2/routing/configuration/guide/b-routing-cg-asr9000-62x/b-routing-cg-asr9000-62x_chapter_01011.html

Restrictions and Guidelines

The following restrictions and guidelines apply for BGP large communities:

- All functionalities of the BGP community attribute is available for the BGP large-community attribute.
- The **send-community-ebgp** command is required for the BGP speaker to send large communities to ebgp neighbors.
- There are no well-known large-communities.
- The peeras expression cannot be used in a large-community-set.
- The peeras expression can only be used in large-community match or delete statements that appear in route policies that are applied at the neighbor-in or neighbor-out attach points.
- The not-peeras expression cannot be used in a large-community-set or in policy set statements.

Configuration Example: Large Community Set

A large-community set defines a set of large communities. Named large-community sets are used in route-policy match and set statements.

This example shows how to create a named large-community set.

```
RP/0/RP0/CPU0:router(config)# large-community-set catbert
RP/0/RP0/CPU0:router(config-largecomm)#  1: 2: 3,
RP/0/RP0/CPU0:router(config-largecomm)#  peeras:2:3
RP/0/RP0/CPU0:router(config-largecomm)# end-set
```

Configuration Example: Set Large Community

The following example shows how to set the BGP large community attribute in a route, using the **set large-community** {*large-community-set-name* | *inline-large-community-set* | *parameter* } [**additive**] command. You can specify a named large-community-set or an inline set. The **additive** keyword retains the large communities already present in the route and adds the new set of large communities. However the **additive** keyword does not result in duplicate entries.

If a particular large community is attached to a route and you specify the same large community again with the additive keyword in the set statement, then the specified large community is not added again. The merging operation removes duplicate entries. This also applies to the peeras keyword.

The peeras expression in the example is replaced by the AS number of the neighbor from which the BGP large community is received or to which the community is sent, as appropriate.

```
RP/0/RP0/CPU0:router(config)# route-policy mordac
RP/0/RP0/CPU0:router(config-rpl)# set large-community (1:2:3, peeras:2:3)
RP/0/RP0/CPU0:router(config-rpl)# end-set
```

```
RP/0/RP0/CPU0:router(config)# large-community-set catbert
RP/0/RP0/CPU0:router(config-largecomm)#  1: 2: 3,
RP/0/RP0/CPU0:router(config-largecomm)#  peeras:2:3
RP/0/RP0/CPU0:router(config-largecomm)# end-set
RP/0/RP0/CPU0:router(config)# route-policy wally
RP/0/RP0/CPU0:router(config-rpl)# set large-community catbert additive
RP/0/RP0/CPU0:router(config-rpl)# end-set
```

In this example, if the route-policy mordac is applied to a neighbor, the ASN of which is 1, then the large community (1:2:3) is set only once.

Note You should configure the **send-community-ebgp** command to send large communities to ebgp neighbors.

Configuration Example: Large Community Matches-any

The following example shows how to configure a route policy to match any element of a large -community set. This is a boolean condition and returns true if any of the large communities in the route match any of the large communities in the match condition.

```
RP/0/RP0/CPU0:router(config)# route-policy elbonia
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-any (1:2:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

Configuration Example: Large Community Matches-every

The following example shows how to configure a route policy where every match specification in the statement must be matched by at least one large community in the route.

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-every (*:*:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

In this example, routes with these sets of large communities return TRUE:

- (1:1:3, 4:5:10)
- (4:5:3) —This single large community matches both specifications.
- (1:1:3, 4:5:10, 7:6:5)

Routes with the following set of large communities return FALSE:

(1:1:3, 5:5:10)—The specification (4:5:*) is not matched.

Configuration Example: Large Community Matches-within

The following example shows how to configure a route policy to match within a large community set. This is similar to the **large-community matches-any** command but every large community in the route must match at least one match specification. Note that if the route has no large communities, then it matches.

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if large-community matches-within (*:*:3, 4:5:*) then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 103
```

```
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

For example, routes with these sets of large communities return TRUE:

- (1:1:3, 4:5:10)
- (4:5:3)
- (1:2:3, 6:6:3, 9:4:3)

Routes with this set of large communities return FALSE:

(1:1:3, 4:5:10, 7:6:5) —The large community (7:6:5) does not match

Configuration Example: Community Matches-within

The following example shows how to configure a route policy to match within the elements of a community set. This command is similar to the **community matches-any** command, but every community in the route must match at least one match specification. If the route has no communities, then it matches.

```
RP/0/RP0/CPU0:router(config)# route-policy bob
RP/0/RP0/CPU0:router(config-rpl)# if community matches-within (*:3, 5:*)  then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 94
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

For example, routes with these sets of communities return TRUE:

- (1:3, 5:10)
- (5:3)
- (2:3, 6:3, 4:3)

Routes with this set of communities return FALSE:

(1:3, 5:10, 6:5) —The community (6:5) does not match.

Configuration Example: Large Community Is-empty

The following example shows using the **large-community is-empty** clause to filter routes that do not have the large-community attribute set.

```
RP/0/RP0/CPU0:router(config)# route-policy lrg_comm_rp4
RP/0/RP0/CPU0:router(config-rpl)# if large-community is-empty then
RP/0/RP0/CPU0:router(config-rpl)#   set local-preference 104
RP/0/RP0/CPU0:router(config-rpl)#  endif
RP/0/RP0/CPU0:router(config-rpl)# end-policy
```

Configuration Example: Attribute Filter Group

The following example shows how to configure and apply the attribute-filter group with large-community attributes for a BGP neighbor. The filter specifies the BGP path attributes and an action to take when BGP update message is received. If an update message is received from the BGP neighbor that contains any of the specified attributes, then the specified action is taken. In this example, the attribute filter named dogbert is created and applied to the BGP neighbor 10.0.1.101. It specifies the large community attribute and the action

of discard. That means, if the large community BGP path attribute is received in a BGP UPDATE message from the neighbor 10.0.1.101 then the attribute will be discarded before further processing of the message.

```
RP/0/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)# attribute-filter group dogbert
RP/0/RP0/CPU0:router(config-bgp-attrfg)# attribute LARGE-COMMUNITY discard
RP/0/RP0/CPU0:router(config-bgp-attrfg)# neighbor 10.0.1.101
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 6461
RP/0/RP0/CPU0:router(config-bgp-nbr)# update in filtering
RP/0/RP0/CPU0:router(config-nbr-upd-filter)# attribute-filter group dogbert
```

Configuration Example: Deleting Large Community

The following example shows how to delete specified BGP large-communities from a route policy using the **delete large-community** command.

```
RP/0/RP0/CPU0:router(config)# route-policy lrg_comm_rp2
RP/0/RP0/CPU0:router(config-rpl)# delete large-community in (ios-regex '^100000:')
RP/0/RP0/CPU0:router(config-rpl)# delete large-community all
RP/0/RP0/CPU0:router(config-rpl)# delete large-community not in (peeras:*:*, 41289:*:*)
```

Verification

This example displays the routes with large-communities given in the **show bgp large-community** *list-of-large-communities* [**exact-match**] command. If the optional keyword exact-match is used, then the listed routes will contain only the specified large communities. Otherwise, the displayed routes may contain additional large communities.

```
RP/0/0/CPU0:R1# show bgp large-community 1:2:3 5:6:7
Thu Mar 23 14:40:33.597 PDT
BGP router identifier 4.4.4.4, local AS number 3
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 66
BGP main routing table version 66
BGP NSR Initial initsync version 3 (Reached)
BGP NSR/ISSU Sync-Group versions 66/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
* 10.0.0.3/32         10.10.10.3               0     94      0 ?
* 10.0.0.5/32         10.11.11.5               0             0 5 ?
```

This example displays the large community attached to a network using the **show bgp** *ip-address/ prefix-length* command.

```
RP/0/0/CPU0:R4# show bgp 10.3.3.3/32
Thu Mar 23 14:36:15.301 PDT
BGP routing table entry for 10.3.3.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 42          42
Last Modified: Mar 22 20:04:46.000 for 18:31:30
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
```

```
  10.11.11.5
Path #1: Received by speaker 0
Advertised to peers (in unique update groups):
  10.11.11.5
Local
  10.10.10.3 from 10.10.10.3 (10.3.3.3)
    Origin incomplete, metric 0, localpref 94, valid, internal, best, group-best
    Received Path ID 0, Local Path ID 0, version 42
    Community: 258:259 260:261 262:263 264:265
    Large Community: 1:2:3 5:6:7 4123456789:4123456780:4123456788
```

Adj-RIB-In Post-Policy View for L3VPN Address Families

Table 11: Feature History Table

Feature Name	Release	Description

Adj-RIB-In Post-Policy View for L3VPN Address Families	Release 7.5.4	After applying policy filters, you can now monitor BGP events and collect BGP route information and statistics for L3VPN address families for unprocessed routing information.
		
		This is made possible because this feature enables the BGP Monitoring Protocol (BMP) to allow a BGP router to advertise the BGP Adj-RIB-In post-policy for L3VPN address families.
		
		This feature introduces these changes:
		
		• **CLI**: This feature modifies the following commands:
		  • **show bgp bmp**
		  • **route-monitoring inbound post-policy**
		
		• **YANG Data Model**: New XPaths for
		  • Cisco-IOS-XR-um-router-bgp-cfg.yang
		  • Cisco-IOS-XR-ipv4-bgp-oper.yang
		  (see GitHub, YANG Data Models Navigator)

The BGP Monitoring Protocol (BMP), defined in RFC 7854, is a protocol to monitor BGP events as well as BGP route information and statistics. Using this protocol, a BMP collector can monitor various routing

information bases within a BGP speaker such as Adj-RIB-In (Pre-Policy and Post-Policy), Local RIB and Adj-RIB-Out (Pre-Policy and Post-Policy). This provides comprehensive insights into real-time and historical operation of a BGP network which can be used for route monitoring, routing analytics, and traffic engineering analytics. BMP can additionally send information on peer state change events, including why a peer went down in the case of a BGP event.

The Adj-RIB-In pre-policy (also referred to as Inbound pre-policy) conveys to a BMP receiver all unprocessed routing information that has been advertised to the local BGP speaker by its peers before any inbound policy has been applied. The Adj-RIB-In post-policy (also referred to as Inbound post-policy) conveys to a BMP receiver all routing information after policy filters and/or modifications (such as addition or deletion of BGP attributes) have been applied.

BMP provides access to the Adjacent Routing Information Base - Inbound (Adj-RIB-In) table of a peer on an ongoing basis and statistics that the monitoring station can use for further analysis. BMP allows a BGP router to advertise the pre-policy or post-policy BGP Adj-RIB-In from the specific BGP peers to a monitoring station.

BGP Adj-RIB-In post-policy (inbound post-policy) view for L3VPN traffic shows the routing information that a BGP peer gets from another peer BGP speaker after applying a BGP input policy and exports the route information to BMP server. The policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

To enable the Adj-RIB-In post-policy (inbound post-policy) for L3VPN address families, you must run configure the **route-monitoring inbound post-policy** command.

In addition to the existing RIB views available for monitoring, Cisco IOS XR Release 7.5.4 adds the following address families in the Adj-RIB-In Post-Policy view for monitoring L3VPN BGP network:

- Default VRF
  - VPNv4 Unicast
  - VPNv6 Unicast
- Non-Default VRF
  - IPv4 Unicast
  - IPv6 Unicast

Configuration

Configure the **route-monitoring inbound post-policy** command to enable the Adj-RIB-In post-policy (inbound post-policy) view by performing the following actions:

```
Router# config
Router(config)#bmp server all
Router(config-bgp-bmp)#route-monitoring inbound post-policy
Router(config-bgp-bmp-rmon)#commit
```

Running Configuration

```
bmp server all
 route-monitoring inbound post-policy
 !
!
```

Verification

Verify whether the Adj-RIB-In post-policy (inbound post-policy) configuration is done by running the **show bgp bmp server** <*server ID*> command.

```
Router# show bgp bmp server 1
Tue Nov 29 19:02:27.837 IST
BMP server 1
Host 12.1.2.1 Port 16001
Connected for 05:51:09
Last Disconnect event received : 00:00:00
Precedence:  internet
BGP neighbors: 7
VRF: - (0x60000000)
Update Source: - (-)
Update Source Vrf ID: 0x0
Update Mode                       : In-Post-Policy
  In-Post-Policy
   Advertisement interval         : 15 secs
   Scanner interval               : 60 secs
Flapping Delay                    : 300 secs
Initial Delay                     : 0 secs
Initial Refresh Delay             : 1 secs
Initial Refresh Spread            : 1 secs
Stats Reporting Period            : 0 secs
Queue Route Mon Msg buffer limit  : 133693 KB (Current Server Up Count: 2)
Queue Route Mon Msg buffer usage  : 0 B
Queue write pulse sent            : Nov 29 13:13:15.484, Nov 29 13:11:53.478 (all)
Queue write pulse received        : Nov 29 13:13:15.484
Update Generation in Progress     : No
Reset Walk in Progress            : No
------More----
```

You can then configure the following commands:

- **bmp advertisement-interval** to set the minimum interval between the sending of BMP routing updates.
- **bmp scan-time** to configure scanning intervals of BMP-speaking networking devices.

Local-RIB view for IP and L3VPN Address Families

Table 12: Feature History Table

Feature Name	Release	Description
Local-RIB view for IP and L3VPN Address Families	Release 7.5.4	After applying policy filters, you can now monitor BGP events and collect BGP best path information and statistics for IP and L3VPN address families for unprocessed routing information. This is made possible because this feature enables BMP to allow a BGP router to advertise the BGP Local-RIB for IP and L3VPN address families. Operators may wish to validate the impact of policies applied to the Adj-RIB-In by analysing the final decision made by the router when installing into the Loc-RIB. This feature introduces these changes: • **CLI**: Modifies the **show bgp bmp** command. • **YANG Data Model**: New XPaths for • Cisco-IOS-XR-um-router-bgp-cfg.yang • Cisco-IOS-XR-ipv4-bgp-oper.yang (see GitHub, YANG Data Models Navigator)

The Local RIB (Loc-RIB) contains the routes that are received from the BGP peers and selected by the local BGP speaker's decision process. The Adj-RIB-In may contain hundreds of thousands of routes for per peer. But only a few of routes are selected and installed in the Loc-RIB after the best-path selection.

The Loc-RIB contains the routes selected by the local BGP speaker's Decision Process and are considered valid to it.

For example, the Adj-RIB-In for a given peer post-policy (inbound post-policy) may contain thousands of routes per peer. But only a few of routes are selected and installed in the Loc-RIB after the best-path selection.

The monitoring application that need only to correlate flow records to Loc-RIB entries, only need to collect and monitor the routes that are actually selected and used. The Loc-RIB includes all selected received routes from BGP peers in addition to locally originated routes. It also contains the address family, the prefixes, attributes, prefixes for address families.

Starting from Release 7.5.4, the Loc-RIB view (best-path only) is available for monitoring for the following address families:

- Default VRF
  - IPv4 Unicast
  - IPv4 Labeled Unicast
  - IPv6 Unicast
  - IPv6 Labeled Unicast
  - VPNv4 Unicast
  - VPNv6 Unicast
- Non-Default VRF
  - IPv4 Unicast
  - IPv6 Unicast

The Route Monitoring messages are used for initial synchronization of the Loc-RIB. They are also used to convey incremental Loc-RIB changes.

This feature complies with RFC 9069.

Configuration

Configure the **route-monitoring local-rib** command to enable the local-RIB view by performing the following actions:

```
Router# config
Router(config)#bmp server all
Router(config-bgp-bmp)#route-monitoring local-rib
Router(config-bgp-bmp-rmon)#commit
```

Running Configuration

```
bmp server all
 route-monitoring local-rib
 !
!
```

Verification

Verify whether the Local RIB (Loc-RIB) configuration is done by running the **show bgp bmp server** <*server ID*> command.

```
Router#show bgp bmp server 1
BMP server 1
Host 12.1.2.1 Port 16001
Connected for 06:00:39
Last Disconnect event received : 00:00:00
Precedence:  internet
BGP neighbors: 10
VRF: - (0x60000000)
Update Source: - (-)
```

```
Update Source Vrf ID: 0x0
Update Mode                        : In-Post-Policy, Local-RIB
  In-Post-Policy
   Advertisement interval          : 15 secs
   Scanner interval                : 60 secs
  Local-RIB
   Advertisement interval          : 15 secs
   Scanner interval
     Global                        : 60 secs
     IPv4 Unicast                  : 60 secs
     VPNv4 Unicast                 : 60 secs
     IPv6 Unicast                  : 60 secs
     VPNv6 Unicast                 : 60 secs
Flapping Delay                     : 300 secs
Initial Delay                      : 0 secs
Initial Refresh Delay              : 1 secs
Initial Refresh Spread             : 1 secs
Stats Reporting Period             : 0 secs
Queue Route Mon Msg buffer limit   : 133693 KB (Current Server Up Count: 2)
Queue Route Mon Msg buffer usage   : 0 B
Queue write pulse sent             : Nov 29 19:08:32.826, Nov 29 13:11:53.478 (all)
Queue write pulse received         : Nov 29 19:08:32.826
Update Generation in Progress      : No
Reset Walk in Progress             : No
----More-----
```

You can then configure **bmp advertisement-interval** command to set the minimum interval between the sending of BMP routing updates.

EVPN Default VRF Route Leaking on the DCI for Internet Connectivity

The EVPN Default VRF Route Leaking feature leak routes between the Default-VRF and Data Center-VRF on the DCI to provide Internet access to data center hosts.

This feature is enabled by:

- Leaking routes from Default-VRF to Data Center-VRF
- Leaking routes to Default-VRF from Data Center-VRF

Protection of Directly connected eBGP neighbors through Interface-based LPTS Identifier

Table 13: Feature History Table

Feature Name	Release Name	Description
Protection of Directly connected eBGP neighbors through Interface-based LPTS Identifier	Release 7.10.1	We have enhanced the network security for directly connected eBGP neighbors by ensuring that only packets originating from designated eBGP neighbors can traverse through a single interface, thus preventing IP spoofing. This is made possible because we've now added an interface identifier for Local Packet Transport Services (LPTS). LPTS filters and polices the packets based on the type of flow rate you configure. The feature introduces these changes: **CLI:** • **bgp lpts-secure-binding** **YANG Data Model:** • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Overview

Local Packet Transport Services (LPTS) maintains tables describing all packet flows destined for the secure domain router (SDR), ensuring that packets are delivered to their intended destinations.

For BGP sessions, LPTS bindings are categorized as follows:

- BGP-Known Entries: These LPTS entries correspond to BGP sessions with established neighbors.
- BGP Configured Peer: LPTS entries in this category are designated to receive the initial packets (TCP SYN, SYN-ACK and ACK) from specifically configured BGP neighbors.
- BGP Default Entries: This category encompasses LPTS entries that capture all packets originating from un-configured BGP neighbors.

IP Spoofing

This section explains how IP spoofing occurs:

1. An attacker spoofing a packet using an exact combination of source IP, destination IP, source port, and destination port floods these packets from another interface within the same VRF.
2. The spoofed packets match the BGP-known LPTS entry, causing them to reach the TCP layer.
3. Packets arriving through any of these entries are policed at the specified rate as all BGP-known LPTS entries share a common LPTS policer.
4. If the packets exceed the rate defined by the LPTS policer, congestion occurs, impacting BGP-established peers and potentially causing instability in BGP sessions. This could lead to flapping.

Prevention of IP Spoofing

This section explains how the Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier feature prevents IP spoofing:

1. The Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier feature prevents IP spoofing by adding an interface identifier for LPTS in directly connected eBGP neighbors.

   You must configure the **bgp lpts-secure-binding** command to enable this feature.
2. When this feature is enabled, LPTS filters and polices packets based on the configured flow rate, allowing only designated eBGP neighbor packets to traverse through a specified interface.
3. In the LPTS binding process through the LPTS socket option, BGP generates a tuple for the interface identifier for every directly configured eBGP neighbor.
4. The configured BGP LPTS entry only matches an incoming connection (TCP SYN packet) if it is received from the specified. interface.
5. Upon receiving a passive connection from the specified interface, and establishing ar the TCP level, TCP inherits the same interface for BGP-known LPTS entry, which is created for this specific connection.
6. Packets that match the source IP, destination IP, source port, destination port, and VRF information of an established connection, but are received from a different interface, are not matched to the LPTS entry. As a result, these packets are directed to the BGP default entry. These packets are subjected to rigorous policing and forwarded to TCP for reset generation. This mechanism ensures that spoofed packets from non-desired interfaces are not affect the BGP-known peer LPTS entries.
7. During the bind process for an active connection, BGP also furnishes the interface identifier. TCP incorporates this interface information into the LPTS entry corresponding to the active connection, effectively safeguarding BGP-known LPTS entries against spoofed packets that might match this connection but originate from a different interface.

The interface identifier is added to the LPTS and TCP only when the following criteria are met:

- The BGP peer is configured to be external.
- The Fast External Failover (FEF) is not disabled.
- The BGP peer is directly connected.
- The BGP peer is not a dynamic peer.

- eBGP multihop is not enabled.
- The default eBGP TTL is used.
- The "ignore connected" option is not configured.
- A non-link local IPv6 neighbor address is configured.

Configure Protection of Directly Connected EBGP Neighbors through Interface-Based LPTS Identifier

To enable Local Packet Transport Services (LPTS) secure binding, perform the following steps:

```
Router#(config)router bgp 100
Router#(config-bgp) bgp lpts-secure-binding
```

Running Configuration

```
router bgp 100
 bgp lpts-secure-binding
```

Verification

Verify the LPTS bindings along with the connected interface identifier:

```
Router# show lpts pifib entry brief

 IPv4    default  TCP    any          [0x00000003]      10.10.10.1,23756 10.10.10.2,179
 IPv4    default  TCP    any          0/0/CPU0           10.10.10.1,179 10.10.10.2
 IPv4    default  TCP    Gi0/2/0/1    [0x00000003]       192.0.2.1,57342 192.0.2.3,179
 IPv4    default  TCP    Gi0/2/0/1    0/0/CPU0           192.0.2.1,179 192.0.2.3
 IPv4   default  TCP    any         [0x00000003]      209.165.201.1,179 209.165.201.4,52798

 IPv4    default  TCP    any          0/0/CPU0          209.165.201.1,179 209.165.201.0/24

 IPv4    default  TCP    Gi0/2/0/3    [0x00000003]       172.16.0.1,179 172.16.0.5,49505
 IPv4    default  TCP    Gi0/2/0/3    0/0/CPU0           172.16.0.1,179 172.16.0.5
 IPv4    default  TCP    any          [0x00000003]       192.168.0.1,179 192.168.0.6,32909
 IPv4    default  TCP    any          0/0/CPU0           192.168.0.1,179 192.168.0.6
```

Verify that the LPTS secure binding is enabled:

```
Router# show bgp process | in LPTS

Wed Dec 14 14:28:33.779 PST
LPTS secure binding is enabled
```

Verify that the status of the connected interface identifier in LPTS is active:

```
Router# show bgp neighbor 192.0.2.3, detail | in Connected

Wed Dec 14 14:28:51.814 PST
  Connected IFH: 0x1000080, IFH in LPTS 0x1000080
```

EIBGP Policy-Based Multipath with Equal Cost Multipath and Default VRF

Table 14: Feature History Table

Feature Name	Release Name	Description
EIBGP Policy-Based Multipath with Equal Cost Multipath and Default VRF	Release 7.10.1	Now, with the inclusion of the default VRF in policy-based multipath selection, you gain control over traffic distribution and load-balancing capabilities across various BGP variations, including iBGP, eBGP, and eiBGP. This is achieved through the utilization of BGP communities, nexthops, and path types. Additionally, by employing the equal cost multipath (ECMP) option in eiBGP, this feature provides the capability to select ECMP across the iBGP paths chosen for eiBGP. The feature introduces these changes: **CLI:** The keywords **route-policy** and **equal-cost** are added to the command: **maximum-paths** **YANG Data Model:** • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Overview

The existing policy-based multipath selection in BGP operates at the Virtual Routing and Forwarding (VRF) address family level, particulary on the neighbor configuration. This process lacks the capability to include the default VRF for variations of BGP, such as iBGP, eBGP and eiBGP. To improve this functionality the policy-based multipath selection is now extended to include iBGP, eBP and eiBGP by utilizing communities as the underlying mechanism. By utilizing communities, the selection of multiple paths based on specific policy criteria becomes more elaborate enabling better flexibility and control over the routing decisions within the BGP network.

In BGP, communities serve as a mechanism for grouping destinations and implementing routing decisions, such as acceptance, rejection, preference, or redistribution, across a set of destinations using community attributes.

This feature ensures that the default VRF is included in the policy-based multipath selection process for the iBGP, eBGP, and eiBGP. Including the default VRF in the policy-based multipath selection process for iBGP, eBGP, and eiBGP provides the following benefits.

- It enhances load balancing by enabling traffic distribution across multiple paths within the default VRF, resulting in optimized resource utilization and improved network performance
- It improves network resiliency as the inclusion of the default VRF provides alternate paths for traffic in the event of failures or disruptions, ensuring uninterrupted connectivity.
- It enables traffic engineering capabilities within the default VRF, thus allowing for more precise control and fine-tuning of traffic flows.

eiBGP traditionally implements the unequal-cost mutipath (UCMP) capability to enable the use of both iBGP and eBGP paths. This feature, utilizing the equal-cost multipath option, ensures that the nexthop iBGP metric remains consistent across the chosen iBGP paths. This is achieved because eBGP and iBGP have distinct path types. Hence the metric evaluation is not performed between eBGP and iBGP paths.

Peering Between BGP Routers Within a Confederation

Table 15: Feature History Table

Feature Name	Release Name	Description
Peering Between BGP Routers Within the Same Confederation	Release 7.11.1	You can now enable BGP peering between routers in the sub-autonomous system (AS) within a confederation to advertise specific router updates using iBGP. This capability ensures that the mesh of routers between sub-ASes in a confederation maintains consistent routing tables, ensuring proper network reachability. Enabling this feature helps improve preventing performance reduction and traffic management challenges. The feature introduces these changes: **CLI:** New Command: • `allowconfedas-in` **YANG Data Models** • New XPaths for `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Overview

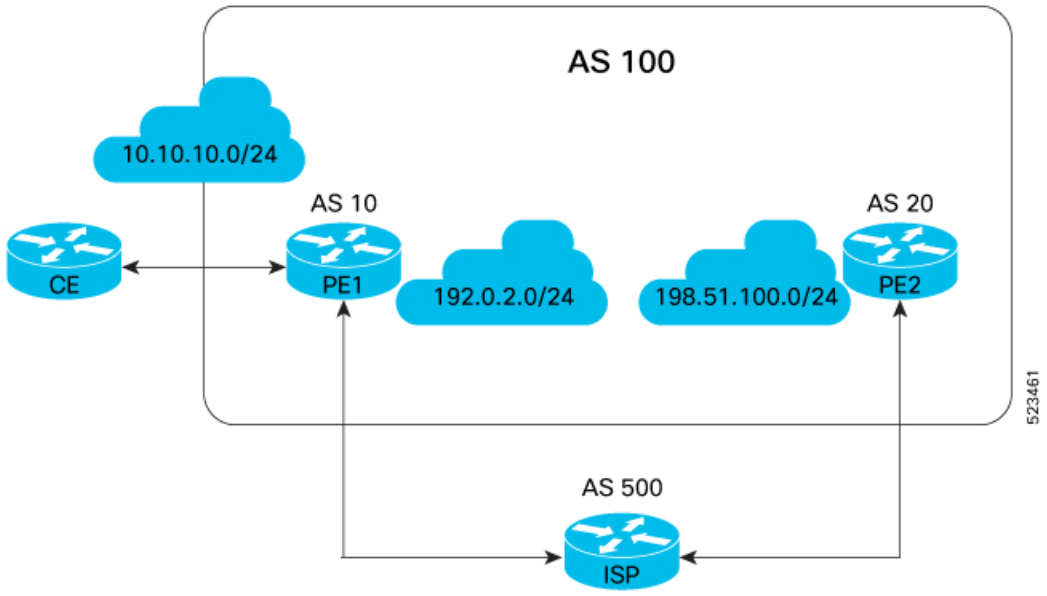
This feature, with its ability to enable BGP peering between routers in the sub-autonomous system (AS) within a confederation allows for specific router updates to be advertised using iBGP. This ensures that in the mesh of routers between sub-ASes in a confederation, the routers maintain consistent routing tables, and ensure proper reachability between networks within the confederation. To enable this featiure, the users need to configure the **allowconfedas-in** command, thus circumventing the split horizon rule. You can specify the number of times the peer routers in the confederation can learn from each other when you configure the **allowconfedas-in** command.

In specific scenarios necessitating routing customization and optimization, breaking the split horizon rule is necessary. This rule restricts routers from sharing routes within the confederation. This feature allows you to

achieve that. You can configure the **allowconfedas-in** command to permit peers to learn routes from the same confederation.

In the topology illustrated in Figure 1: Peering Between BGP Routers Within the Same Confederation, the PE1 router connects to the ISP router via the 192.0.2.0/24 prefix, while the PE2 router connects via the 198.51.100.0/24 prefix. The CE router advertises the 10.10.10.0/24 route to PE1, which, in turn, advertises it to PE2. To achieve this, PE1 advertises the route to the ISP router, which then passes it to PE2 since PE1 and PE2 aren't directly connected. While relaying the advertisement, the ISP router learns the route. PE2, with a confederation AS number of AS 20, examines the AS number list in the advertisement to understand the route's path. PE2 identifies the AS numbers of the ISP router , which is AS 500, and PE1 router, which is AS 100. As the AS numbers of both PE1 and PE2 routers match, indicating they belong to the same confederation, PE2 drops the route in accordance with the split horizon rule. Hence these routers do not learn each others routes. The PE1 and PE2 routers are part of the same confederation and have different AS numbers. In this case, the **allowas-in** command, which prevents dropping of the routes coming from a peer router of the same autonmous system, is not enough to allow the loop detection to be bypassed. Because of this, PE2 will not be able to learn prefixes from PE1 router. To override the split horizon rule and prevent PE2 from discarding the learned route, configure the **allowconfedas-in** command on both the PE1 and PE2 routers. The **allowconfedas-in** command enables you to configure the frequency with which peer routers within the same confederation learn from each other.

Figure 17: Peering Between BGP Routers Within the Same Confederation



Terminology

Autonomous Systems:

BGP, operating as an Exterior Gateway Protocol (EGP), establishes loop-free interdomain routing between autonomous systems (AS). An AS comprises routers under single administration, utilizing IGPs for internal routing. Additionally, it employs EGP to route packets beyond its boundaries.

Sub-Autonomous System

A sub-autonomous system is a distinct subset within a larger autonomous system, possessing individual administrative control. It operates with specific routing policies, contributing to the hierarchical organization and efficient management of network configurations.

Confederation:

To reduce the iBGP mesh, an autonomous system can be segmented into sub-autonomous systems organized into a confederation. Externally, this confederation appears as a single autonomous system. Internally, each autonomous system is fully meshed but maintains limited connections to others in the same confederation. Peers in different autonomous systems engage in eBGP sessions, exchanging routing information resembling iBGP peers, preserving vital parameters like next hop, MED, and local preference.

Autonomous System Number

The Autonomous System Number (ASN) is crucial in networking, serving as a unique identifier for autonomous systems, including sub-autonomous systems within a confederation.

Split Horizon

Split horizon, a network protocol routing rule, boosts stability by prohibiting routers in the same confederation from sharing routes. It prevents a router from advertising routes back to the network from which it learned them. This prevents potential loops, ensuring accurate network topology views and enabling efficient data forwarding, thereby addressing routing issues.

Restrictions for Peering Between BGP Routers Within the Same Confederation

Peer routers within a confederation are restricted in the frequency at which they can exchange information with each other on configuring the **allowconfedas-in** command. The number of times they can share information ranges from 1 to 10. The default value is 3.

Peering Between BGP Routers Within the Same Confederation: Terminology

Autonomous System

BGP, operating as an Exterior Gateway Protocol (EGP), establishes loop-free interdomain routing between autonomous systems (AS). An AS comprises routers under single administration, utilizing IGPs for internal routing. Additionally, it employs EGP to route packets beyond its boundaries.

Sub-Autonomous System

A sub-autonomous system is a distinct subset within a larger autonomous system, possessing individual administrative control. It operates with specific routing policies, contributing to the hierarchical organization and efficient management of network configurations.

Confederation

To reduce the iBGP mesh, an autonomous system can be segmented into sub-autonomous systems organized into a confederation. Externally, this confederation appears as a single autonomous system. Internally, each autonomous system is fully meshed but maintains limited connections to others in the same confederation. Peers in different autonomous systems engage in eBGP sessions, exchanging routing information resembling iBGP peers, preserving vital parameters like next hop, MED, and local preference.

Autonomous System Number

The Autonomous System Number (ASN) is crucial in networking, serving as a unique identifier for autonomous systems, including sub-autonomous systems within a confederation.

Split Horizon

Split horizon, a network protocol routing rule, boosts stability by prohibiting routers in the same confederation from sharing routes. It prevents a router from advertising routes back to the network from which it learned them. This prevents potential loops, ensuring accurate network topology views and enabling efficient data forwarding, thereby addressing routing issues.

Configure Peering Between BGP Routers Within the Same Confederation

Configuration Example

To enable peering between routers that exist in the same confederation, perform the following steps:

- Enter router configuration mode.
- Assign BGP autonomous systems belonging to a confederation.
- Assign an identifier to the confederation.
- Place the router in neighbor configuration mode for routing and configure the neighbor IP address as a BGP peer.
- Specify either the IPv4 or IPv6 address family and enter address family configuration submode.
- Enable peer routers in the same confederation to learn from each other for a specified number of times.

```
Router# router bgp 65001
Router(config-bgp)# bgp confederation peers 65002
Router(config-bgp)# bgp confederation identifier 100
Router(config-bgp)# neighbor 198.51.100.3
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# allowconfedas-in 1
```

Running Configuration

```
router bgp 65001
 bgp confederation peers 65002
 bgp confederation identifier 100
 neighbor 198.51.100.3
  address-family ipv4 unicast
   allowconfedas-in 1
```

Verification

Verify the learning of routes among BGP peers. This output shows that the peers within the same confederation have learned from each others' routes, and the learning among peers has occurred thrice.

```
show bgp neighbor 198.51.100.3 | in allow
Fri Mar  7 15:38:13.092 +0530
  Inbound soft reconfiguration allowed (override route-refresh)
  My confederation AS number is allowed 3 times in received updates.
```

Virtual Routing Forwarding Next Hop Routing Policy

Table 16: Feature History Table

Feature Name	Release Name	Description
Virtual Routing Forwarding Next Hop Routing Policy	Release 7.11.1	You can now enable a route policy at the BGP next-hop attach point to limit notifications delivered to BGP for specific prefixes, which equips you with better control over routing decisions, and allows for precise traffic engineering and security compliance for each VRF instance, and helps establish redundant paths specific to each VRF. The feature introduces these changes: **CLI:** Modified Command: • The `nexthop route-policy` command is extended to VRF address-family configuration mode. **YANG Data Model** • New XPaths for `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Configure VRF Next Hop Policy

To enable next hop route policy on a VRF table, perform the following steps:

- Configure a route policy and enter route-policy configuration mode.
- Define the route policy to help limit notifications delivered to BGP for specific prefixes.
- Drop the prefix of the routes that matches the conditions set in the route policy.
- Enable BGP routing and enter the router configuration mode.
- Configure a VRF.

- Configure an IPv4 or IPv6 address family.
- Configure route policy filtering using next hops.

```
Router(config)# route-policy nh-route-policy
Router(config-rpl)# if destination in (10.1.1.0/24) and protocol in (connected, static)
then
Router(config-rpl-if)# drop
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config-rpl)# exit
Router(config)# router bgp 500
Router(config-bgp)# vrf vrf10
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# nexthop route-policy nh-route-policy
```

Running Configuration

```
route-policy nh-route-policy
 if destination in (10.1.1.0/24) and protocol in (connected, static) then
  drop
  endif
end-policy
!

router bgp 500
 vrf vrf10
  address-family ipv4 unicast
    nexthop route-policy nh-route-policy
```

Verification

Verify that the configurred next route hop policy is enabled in a VRF table. The "BGP table nexthop route policy" field indicates the route policy used to determine the next hop for BGP routes in the specified VRF instance VRF1.

```
Router# show bgp vrf vrf1 ipv4 unicast
Fri Jul  7 15:51:16.309 +0530
BGP VRF vrf1, state: Active
BGP Route Distinguisher: 1:1
VRF ID: 0x6000000b
BGP router identifier 10.1.1.1, local AS number 65001
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe000000b   RD version: 1356
BGP table nexthop route policy: nh-route-policy --> This is the same route policy that was
 configured.
BGP main routing table version 1362
BGP NSR Initial initsync version 1355 (Reached)
BGP NSR/ISSU Sync-Group versions 1362/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
 Network           Next Hop      Metric  LocPrf Weight Path
Route Distinguisher: 1:1 (default for vrf vrf1)
Route Distinguisher Version: 1356
*> 10.1.1.0/24      0.0.0.0       0       32768  ?
```

```
*> 192.0.2.0/24     10.1.1.1      0       32768  ?
*> 198.50.100.0/24  10.1.1.1      0               101  i
```

Configuration Examples for Implementing BGP

This section provides the following configuration examples:

Enabling BGP: Example

The following shows how to enable BGP.

```
prefix-set static
   2020::/64,
   2012::/64,
   10.10.0.0/16,
   10.2.0.0/24
end-set

route-policy pass-all
  pass
end-policy
route-policy set_next_hop_agg_v4
  set next-hop 10.0.0.1
end-policy

route-policy set_next_hop_static_v4
  if (destination in static) then
    set next-hop 10.1.0.1
  else
    drop
  endif
end-policy

route-policy set_next_hop_agg_v6
  set next-hop 2003::121
end-policy

route-policy set_next_hop_static_v6
  if (destination in static) then
     set next-hop 2011::121
  else
     drop
  endif
end-policy

router bgp 65000
  bgp fast-external-fallover disable
  bgp confederation peers
    65001
    65002
  bgp confederation identifier 1
  bgp router-id 192.0.2.1
  address-family ipv4 unicast
    aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
    aggregate-address 10.3.0.0/24
    redistribute static route-policy set_next_hop_static_v4
  address-family ipv4 multicast
    aggregate-address 10.2.0.0/24 route-policy set_next_hop_agg_v4
    aggregate-address 10.3.0.0/24
```

```
    redistribute static route-policy set_next_hop_static_v4
  address-family ipv6 unicast
    aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
    aggregate-address 2013::/64
    redistribute static route-policy set_next_hop_static_v6
  address-family ipv6 multicast
    aggregate-address 2012::/64 route-policy set_next_hop_agg_v6
    aggregate-address 2013::/64
    redistribute static route-policy set_next_hop_static_v6
  neighbor 10.0.101.60
    remote-as 65000
    address-family ipv4 unicast
    address-family ipv4 multicast
  neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast
    address-family ipv4 multicast
  neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
    address-family ipv4 multicast
      route-policy pass-all in
      route-policy pass-all out
  neighbor 10.0.101.64
    remote-as 5
    update-source Loopback0
    address-family ipv4 unicast
      route-policy pass-all in
      route-policy pass-all out
    address-family ipv4 multicast
      route-policy pass-all in
      route-policy pass-all out
```

Displaying BGP Update Groups: Example

The following is sample output from the **show bgp update-group** command run in EXEC configuration mode:

```
show bgp update-group

 Update group for IPv4 Unicast, index 0.1:
   Attributes:
     Outbound Route map:rm
     Minimum advertisement interval:30
   Messages formatted:2, replicated:2
   Neighbors in this update group:
     10.0.101.92

 Update group for IPv4 Unicast, index 0.2:
   Attributes:
     Minimum advertisement interval:30
   Messages formatted:2, replicated:2
   Neighbors in this update group:
     10.0.101.91
```

BGP Neighbor Configuration: Example

The following example shows how BGP neighbors on an autonomous system are configured to share information. In the example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured shares information about networks 172 .16 .0.0 and 192.168 .7.0 with the neighbor routers. The first router listed is in a different autonomous system; the second **neighbor** and **remote-as** commands specify an internal neighbor (with the same autonomous system number) at address 172 .26 .234.2; and the third **neighbor** and **remote-as** commands specify a neighbor on a different autonomous system.

```
route-policy pass-all
 pass
end-policy
router bgp 109
 address-family ipv4 unicast
  network 172.16.0.0 255.255.0.0
  network 192.168.7.0 255.255.0.0
  neighbor 172.16.200.1
   remote-as 167
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-out out
  neighbor 172.26.234.2
   remote-as 109
   exit
 address-family ipv4 unicast
  neighbor 172.26.64.19
   remote-as 99
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

BGP Confederation: Example

The following is a sample configuration that shows several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 666 (specified using the **bgp confederation identifier** command).

In a BGP speaker in autonomous system 6001, the **bgp confederation peers** command marks the peers from autonomous systems 6002 and 6003 as special eBGP peers. Hence, peers 171.16 .232.55 and 171.16 .232.56 get the local preference, next hop, and MED unmodified in the updates. The router at 171 .19 .69.1 is a normal eBGP speaker, and the updates received by it from this peer are just like a normal eBGP update from a peer in autonomous system 666.

```
router bgp 6001
 bgp confederation identifier 666
 bgp confederation peers
  6002
  6003
   exit
 address-family ipv4 unicast
  neighbor 171.16.232.55
```

```
  remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 171.16.232.56
  remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 171.19.69.1
  remote-as 777
```

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special eBGP peers. Peer 171 .17 .70.1 is a normal iBGP peer, and peer 199.99.99.2 is a normal eBGP peer from autonomous system 700.

```
router bgp 6002
 bgp confederation identifier 666
 bgp confederation peers
  6001
  6003
   exit
 address-family ipv4 unicast
  neighbor 171.17.70.1
   remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.57
   remote-as 6001
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.56
   remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 171.19.99.2
   remote-as 700
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special eBGP peers. Peer 192 .168 .200.200 is a normal eBGP peer from autonomous system 701.

```
router bgp 6003
 bgp confederation identifier 666
 bgp confederation peers
  6001
  6002
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.57
   remote-as 6001
   exit
 address-family ipv4 unicast
  neighbor 171.19.232.55
   remote-as 6002
   exit
```

```
 address-family ipv4 unicast
  neighbor 192.168.200.200
   remote-as 701
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
```

The following is a part of the configuration from the BGP speaker 192 .168 .200.205 from autonomous system 701 in the same example. Neighbor 171.16 .232.56 is configured as a normal eBGP speaker from autonomous system 666. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```
router bgp 701
 address-family ipv4 unicast
  neighbor 172.16.232.56
   remote-as 666
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
   exit
 address-family ipv4 unicast
  neighbor 192.168.200.205
   remote-as 701
```

BGP Route Reflector: Example

The following example shows how to use an address family to configure internal BGP peer 10.1.1.1 as a route reflector client for both unicast and multicast prefixes:

```
router bgp 140
 address-family ipv4 unicast
  neighbor 10.1.1.1
   remote-as 140
   address-family ipv4 unicast
    route-reflector-client
    exit
   address-family ipv4 multicast
    route-reflector-client
```

BGP Nonstop Routing Configuration: Example

The following example shows how to enable BGP NSR:

```
configure
router bgp 120
nsr
end
```

The following example shows how to disable BGP NSR:

```
configure
router bgp 120
no nsr
end
```

Primary Backup Path Installation: Example

The following example shows how to enable installation of primary backup path:

```
 router bgp 120
address-family ipv4 unicast
 additional-paths receive
 additional-paths send
 additional-paths selection route-policy bgp_add_path
!
!
end
```

Allocated Local Label Retention: Example

The following example shows how to retain the previously allocated local label for the primary path on the primary PE for 10 minutes after reconvergence:

```
 router bgp 100
 address-family l2vpn vpls-vpws
   retain local-label 10
 end
```

iBGP Multipath Loadsharing Configuration: Example

The following is a sample configuration where 30 paths are used for loadsharing:

```
router bgp 100
 address-family ipv4 multicast
  maximum-paths ibgp 30
 !
!
end
```

Discard Extra Paths Configuration: Example

The following example shows how to configure discard extra paths feature for the IPv4 address family:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 10
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# maximum-prefix 1000 discard-extra-paths
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# commit
```

Displaying Discard Extra Paths Information: Example

The following screen output shows details about the discard extra paths option:

```
RP/0/0/CPU0:ios# show bgp neighbor 10.0.0.1

BGP neighbor is 10.0.0.1
Remote AS 10, local AS 10, internal link
Remote router ID 0.0.0.0
BGP state = Idle (No best local address found)
Last read 00:00:00, Last read before reset 00:00:00
Hold time is 180, keepalive interval is 60 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
Last write 00:00:00, attempted 0, written 0
Second last write 00:00:00, attempted 0, written 0
Last write before reset 00:00:00, attempted 0, written 0
Second last write before reset 00:00:00, attempted 0, written 0
Last write pulse rcvd not set last full not set pulse count 0
Last write pulse rcvd before reset 00:00:00
Socket not armed for io, not armed for read, not armed for write
Last write thread event before reset 00:00:00, second last 00:00:00
Last KA expiry before reset 00:00:00, second last 00:00:00
Last KA error before reset 00:00:00, KA not sent 00:00:00
Last KA start before reset 00:00:00, second last 00:00:00
Precedence: internet
Multi-protocol capability not received
Received 0 messages, 0 notifications, 0 in queue
Sent 0 messages, 0 notifications, 0 in queue
Minimum time between advertisement runs is 0 secs

For Address Family: IPv4 Unicast
BGP neighbor version 0
Update group: 0.1 Filter-group: 0.0 No Refresh request being processed
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 0, suppressed 0, withdrawn 0
Maximum prefixes allowed 10 (discard-extra-paths) <<<<<<<<<<<<<<<<<<<<<<
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was not received during read-only mode
Last ack version 1, Last synced ack version 0
Outstanding version objects: current 0, max 0
Additional-paths operation: None
Send Multicast Attributes

Connections established 0; dropped 0
Local host: 0.0.0.0, Local port: 0, IF Handle: 0x00000000
Foreign host: 10.0.0.1, Foreign port: 0
Last reset 00:00:00
```

Advertising IPv4 NLRI with IPv6 Next Hops in MP-BGP Networks

Many multiprotocol network deployments today have topologies of devices configured with one type of address family interspersed with devices configured with a different type of address family. An IPv4 core network can be surrounded by IPv6 devices, and vice versa. Such deployments require the use of multiprotocol BGP (MP-BGP) that allow the advertisement of IPv4 NLRI across IPv6 next hops. Hence, when MP-BGP is used to advertise the corresponding reachability information in such heterogenous networks, the BGP router (speaker) advertises the NLRI of a given address family through a next hop of a different address family.

There are several reasons for these heterogenous deployments. A primary reason is the lack of availability of IPv4 addresses to be used on the interfaces of BGP speakers. A second reason is the intent to move to a pure IPv6 deployment, but in phases. Hence, by configuring the **ipv4 forwarding-enable** command on interfaces, the type of address family used does not impact the flow of traffic in the network.

Note Some BGP peer nodes will not establish a BGP session if the neighbor does not support the Extended Next-hop Encoding capabilities (RFC 5549). Use the **capability suppress extended-nexthop-encoding** command to suppress the advertisement of this capability.

This section describes the configuration required to enable IPv4 NLRI advertisement across an IPv6 next hop.

Configuration

Use the following configuration for advertising IPv4 NLRI through IPv6 netxhops.

In the following example, the eBGP peer is configured on the **GigabitEthernet 0/0/0/0** interface, and the iBGP peer is configured on the **GigabitEthernet0/0/0/2** interface of the router being configured.

```
/* Configure the required GigE interfaces with an IPv6 address
 and the ipv4 forwarding-enable command */
Router(config)# interface GigabitEthernet 0/0/0/0
Router(config-if)# ipv6 address 2000::2/64
Router(config-if)# ipv4 forwarding-enable
Router(config-if)# no shut
Router(config-if)# commit
Tue Mar  6 10:11:17.910 IST
Router:Mar  6 10:11:17.968 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/0, changed state to Down
Router:Mar  6 10:11:17.983 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/0, changed state to Up
Router(config-if)# exit

Router(config)# interface GigabitEthernet 0/0/0/2
Router(config-if)# ipv6 address 3000::2/64
Router(config-if)# ipv4 forwarding-enable
Router(config-if)# no shut
Router(config-if)# commit
Tue Mar  6 10:12:15.948 IST
RP/0/0/CPU0:Mar  6 10:12:15.978 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/2, changed state to Down
RP/0/0/CPU0:Mar  6 10:12:15.994 : ifmgr[403]: %PKT_INFRA-LINK-3-UPDOWN : Interface
GigabitEthernet0/0/0/2, changed state to Up
Router(config-if)# exit

/* Create a route policy to set an IPv6 nexthop for IPv4 routes */
Router(config)# route-policy 5549
Router(config-rpl)# if destination in 5549-pfx then set next-hop 20:20::20:200 endif
Router(config-rpl)# end-policy
Router(config)# prefix-set 5549-pfx
Router(config-pfx)# 3.3.3.3/32, 100.1.1.0/24 end-set
Router(config)# commit
Tue Mar  6 10:26:31.749 IST

/* Configure an iBGP peer through the GigabitEthernet0/0/0/2 interface */
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
```

```
Router(config-bgp)# address-family ipv4 multicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 3000::1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# commit
Tue Mar  6 10:16:07.134 IST
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit

/* Configure an eBGP peer through the GigabitEthernet 0/0/0/0 interface
and use the route policy for setting an IPv6 nexthop for IPv4 routes */
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# address-family ipv4 multicast
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 2000::1
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy 5549 out
Router(config-bgp-nbr-af)# commit
Tue Mar  6 10:21:19.434 IST
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit

/* Before proceeding to feature verification,
confirm your configuration by using the show run command from the Executive mode */
```

Verification

Use the following show commands to verify the advertisement of IPv4 NLRI through IPv6 nexthops.

```
/* Verify BGP neighbor configuration and the advertisement of nexthops */
Router# show bgp neighbor
BGP neighbor is 10:10::10:10
 Remote AS 100, local AS 100, internal link
 Remote router ID 10.10.10.10
 Cluster ID 30.30.30.30
  BGP state = Established, up for 11:42:17
  NSR State: NSR Ready
  Last read 00:00:10, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:09, attempted 19, written 19
  Second last write 00:01:09, attempted 19, written 19
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Jun 12 11:57:40.005 last full Jun 12 03:38:09.496 pulse count 1766

  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
```

```
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Multi-protocol capability received
  Neighbor capabilities:
    Route refresh: advertised (old + new) and received (old + new)
    4-byte AS: advertised and received
Address family IPv4 Unicast: advertised and received
    Received 866 messages, 0 notifications, 0 in queue
Sent 1021 messages, 0 notifications, 0 in queue
  Minimum time between advertisement runs is 0 secs
  Inbound message logging enabled, 3 messages buffered
  Outbound message logging enabled, 3 messages buffered

 For Address Family: IPv4 Unicast
  BGP neighbor version 120021
  Update group: 0.2 Filter-group: 0.2  No Refresh request being processed
  Route-Reflector Client
    Extended Nexthop Encoding: advertised and received
  Route refresh request: received 0, sent 0
  11 accepted prefixes, 11 are bestpaths
  Exact no. of prefixes denied : 0.
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 60006, suppressed 0, withdrawn 60000
  Maximum prefixes allowed 1048576
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 120021, Last synced ack version 120021
  Outstanding version objects: current 0, max 3
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise routes with local-label via Unicast SAFI

  Connections established 1; dropped 0
  Local host: 30:30::30:30, Local port: 51453, IF Handle: 0x00000000
  Foreign host: 10:10::10:10, Foreign port: 179
  Last reset 00:00:00

/* Verify BGP nexthop encoding and the n
Router# show bgp ipv4 unicast update-group
Mon Jun 12 11:47:31.543 UTC

Update group for IPv4 Unicast, index 0.2:
  Attributes:
    Neighbor sessions are IPv6
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    Route Reflector Client
    4-byte AS capable
    Advertise routes with local-label via Unicast SAFI
    Send AIGP
    Send multicast attributes
    Extended Nexthop Encoding
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 5
  Number of refresh subgroups: 0
  Messages formatted: 156, replicated: 228
```

```
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      10:10::10:10
      20:20::20:20

Router# show bgp 3.3.3.3/32
Mon Jun 12 11:57:59.451 UTC
BGP routing table entry for 3.3.3.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 21          21
Last Modified: Jun 12 00:15:45.314 for 11:42:14
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  3000, (Received from a RR-client)
    20:20::20:20 (metric 1) from 20:20::20:20 (20.20.20.20)
      Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best
      Received Path ID 0, Local Path ID 0, version 21


Router# show bgp ipv4 unicast nexthops
...
    ……. Snippet …..

Gateway Address Family: IPv6 Unicast
Table ID: 0xe0800000
Nexthop Count: 2
Critical Trigger Delay: 3000msec
Non-critical Trigger Delay: 10000msec

Nexthop Version: 3, RIB version: 1
EPE Table Version: 1, EPE Label version: 1
EPE Downloaded Version: 1, EPE Standby Version: 1

Status codes: R/UR Reachable/Unreachable
              C/NC Connected/Not-connected
              L/NL Local/Non-local
              PR   Pending Registration
              I    Invalid (Policy drop)
Next Hop        Status          Metric     Tbl-ID   Notf   LastRIBEvent      RefCount
10:10::10:10    [R][NC][NL]          1e0800000   1/0    11:42:43 (Cri)       11/14
20:20::20:20    [R][NC][NL]          1e0800000   1/0    11:42:43 (Cri)        8/11


Router# show bgp ipv4 unicast nexthops 10:10::10:10
Nexthop: 10:10::10:10
  VRF: Default
  Nexthop ID: 0x6000001, Version: 0x2
  Nexthop Flags: 0x00000080
  Nexthop Handle: 0x7f53e85b136c

  RIB Related Information:
  Firsthop interface handle 0x00000140
    Gateway TBL Id: 0xe0800000    Gateway Flags: 0x00000080
    Gateway Handle: 0x7f540d2e8f00
    Gateway: reachable, non-Connected route, prefix length 128
    Resolving Route: 10:10::10:10/128 (ospf 100)
    Paths: 0
    RIB Nexhop ID: 0x0
```

```
    Status: [Reachable][Not Connected][Not Local]
    Metric: 1
    Registration: Asynchronous, Completed: 2d21h
    Events: Critical (1)/Non-critical (0)
    Last Received: 11:42:55 (Critical)
    Last gw update: (Crit-notif) 11:42:55(rib)
    Reference Count: 11

  Prefix Related Information
    Active Tables: [IPv4 Unicast]
    Metrices: [0x1]
    Reference Counts: [11]
  Interface Handle: 0x0

Router# show route 3.3.3.3/32
Mon Jun 12 12:32:13.503 UTC

Routing entry for 3.3.3.3/32
  Known via "bgp 100", distance 200, metric 0
  Tag 3000, type internal
  Installed Jun 12 00:15:45.626 for 12:16:28
  Routing Descriptor Blocks
    20:20::20:20, from 20:20::20:20
      Route metric is 0
  No advertising protos.


Router# show route 3.3.3.3/32 detail
Mon Jun 12 12:32:16.447 UTC

Routing entry for 3.3.3.3/32
  Known via "bgp 100", distance 200, metric 0
  Tag 3000, type internal
  Installed Jun 12 00:15:45.628 for 12:16:30
  Routing Descriptor Blocks
    20:20::20:20, from 20:20::20:20
      Route metric is 0
      Label: None
      Tunnel ID: None
      Binding Label: None
      Extended communities count: 0
      NHID:0x0(Ref:0)
  Route version is 0x1 (1)
  No local label
  IP Precedence: Not Set
  QoS Group ID: Not Set
  Flow-tag: Not Set
  Fwd-class: Not Set
  Route Priority: RIB_PRIORITY_RECURSIVE (12) SVD Type RIB_SVD_TYPE_LOCAL
  Download Priority 4, Download Version 24
  No advertising protos.

Router# show cef 3.3.3.3/32
Mon Jun 12 12:32:22.627 UTC
3.3.3.3/32, version 24, internal 0x5000001 0x0 (ptr 0x8e0b76b8) [1], 0x0 (0x0), 0x0 (0x0)
 Updated Jun 12 00:15:45.631
 local adjacency fe80::f83b:74ff:fe65:f004
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
   via 20:20::20:20/128, 2 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x8e352034 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop 20:20::20:20/128 via 20:20::20:20/128
```

```
Router# show cef 3.3.3.3/32 detail
Mon Jun 12 12:32:25.415 UTC
3.3.3.3/32, version 24, internal 0x5000001 0x0 (ptr 0x8e0b76b8) [1], 0x0 (0x0), 0x0 (0x0)
 Updated Jun 12 00:15:45.632
 local adjacency fe80::f83b:74ff:fe65:f004
 Prefix Len 32, traffic index 0, precedence n/a, priority 4
  gateway array (0x8eec3170) reference count 6, flags 0x2010, source rib (7), 0 backups
                [1 type 3 flags 0x48501 (0x8e191998) ext 0x0 (0x0)]
  LW-LDI[type=0, refc=0, ptr=0x0, sh-ldi=0x0]
  gateway array update type-time 1 Jun 12 00:15:45.632
 LDI Update time Jun 12 00:15:45.632
   via 20:20::20:20/128, 2 dependencies, recursive [flags 0x6000]
    path-idx 0 NHID 0x0 [0x8e352034 0x0]
    next hop VRF - 'default', table - 0xe0800000
    next hop 20:20::20:20/128 via 20:20::20:20/128


    Load distribution: 0 1 (refcount 1)

    Hash  OK  Interface                 Address
    0     Y   FortyGigE0/0/0/0          fe80::f83b:74ff:fe65:f004
    1     Y   FortyGigE0/0/0/25         fe80::f83b:74ff:fe65:f08c
```

You have successfully configured and verified the advertisement of IPv4 NLRI through IPv6 nexthops.

Configure Per Neighbor TCP MSS: Examples

These examples show how to configure per neighbor TCP MSS, disable per neighbor TCP MSS, and unconfigure TCP MSS.

Topology Scenario

This figure shows a basic scenario for per neighbor TCP MSS configuration.

R1 Configuration:

```
router bgp 1
  bgp router-id 10.0.0.1
  address-family ipv4 unicast
  !
  neighbor-group n1
    tcp mss 100
    address-family ipv4 unicast
  !
 !
  neighbor 10.0.0.2
    remote-as 1
    use neighbor-group n1
    address-family ipv4 unicast
  !
 !
!
```

R2 Configuration:

```
router bgp 1
  bgp router-id 10.0.0.2
  address-family ipv4 unicast
  !
  neighbor 10.0.0.1
    remote-as 1
    address-family ipv4 unicast
  !
 !
!
```

Configure Per Neighbor TCP MSS: Example

The following example shows how to configure per neighbor TCP MSS under neighbor group:

```
router bgp 1
bgp router-id 10.0.0.1
address-family ipv4 unicast
!
neighbor-group n1
tcp mss 500
address-family ipv4 unicast
!
!
neighbor 10.0.0.2
remote-as 1
use neighbor-group n1
address-family ipv4 unicast
!
!
!
!
end
```

Disable Per Neighbor TCP MSS: Example

The following example shows how to configure TCP MSS under neighbor group and configure inheritance disable under one of the neighbors inheriting the TCP MSS value:

```
router bgp 1
bgp router-id 10.0.0.1
address-family ipv4 unicast
!
neighbor-group n1
tcp mss 500
address-family ipv4 unicast
!
!
neighbor 10.0.0.2
remote-as 1
use neighbor-group n1
tcp mss inheritance-disable
address-family ipv4 unicast
!
!
!
!
```

```
end
```

Unconfigure TCP MSS: Example

The following example shows how to unconfigure TCP MSS:

```
RP/0/0/CPU0:ios(config)#router bgp 1
RP/0/0/CPU0:ios(config-bgp)#neighbor-group n1
RP/0/0/CPU0:ios(config-bgp-nbrgrp)#no tcp mss 500
RP/0/0/CPU0:ios(config-bgp-nbrgrp)#commit
```

Verify Per Neighbor TCP MSS: Examples

The following example shows how to verify the per neighbor TCP MSS feature on a router:

```
RP/0/0/CPU0:ios#show bgp neighbor 10.0.0.2

BGP neighbor is 10.0.0.2
Remote AS 1, local AS 1, internal link
Remote router ID 10.0.0.2
BGP state = Established, up for 00:09:17
Last read 00:00:16, Last read before reset 00:00:00
Hold time is 180, keepalive interval is 60 seconds
Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
Last write 00:00:16, attempted 19, written 19
Second last write 00:01:16, attempted 19, written 19
Last write before reset 00:00:00, attempted 0, written 0
Second last write before reset 00:00:00, attempted 0, written 0
Last write pulse rcvd Dec 7 11:58:42.411 last full not set pulse count 23
Last write pulse rcvd before reset 00:00:00
Socket not armed for io, armed for read, armed for write
Last write thread event before reset 00:00:00, second last 00:00:00
Last KA expiry before reset 00:00:00, second last 00:00:00
Last KA error before reset 00:00:00, KA not sent 00:00:00
Last KA start before reset 00:00:00, second last 00:00:00
Precedence: internet
Multi-protocol capability received
Neighbor capabilities:
Route refresh: advertised (old + new) and received (old + new)
Graceful Restart (GR Awareness): advertised and received
4-byte AS: advertised and received
Address family IPv4 Unicast: advertised and received
Received 12 messages, 0 notifications, 0 in queue
Sent 12 messages, 0 notifications, 0 in queue
Minimum time between advertisement runs is 0 secs
TCP Maximum Segment Size 500

For Address Family: IPv4 Unicast
BGP neighbor version 4
Update group: 0.2 Filter-group: 0.1 No Refresh request being processed
Route refresh request: received 0, sent 0
0 accepted prefixes, 0 are bestpaths
Cumulative no. of prefixes denied: 0.
Prefix advertised 0, suppressed 0, withdrawn 0
Maximum prefixes allowed 1048576
Threshold for warning message 75%, restart interval 0 min
AIGP is enabled
An EoR was received during read-only mode
Last ack version 4, Last synced ack version 0
```

```
Outstanding version objects: current 0, max 0
Additional-paths operation: None
Send Multicast Attributes
```

The following example shows how to verify the TCP MSS configuration:

```
RP/0/0/CPU0:ios#show bgp neighbor 10.0.0.2 configuration

neighbor 10.0.0.2
remote-as 1 []
tcp-mss 400 [n:n1]
address-family IPv4 Unicast []
```

The following example shows how to display TCP connection endpoints information:

```
RP/0/0/CPU0:ios#show tcp brief

  PCB      VRF-ID     Recv-Q Send-Q Local Address          Foreign Address         State
0x08789b28 0x60000000      0      0  :::179                 :::0                    LISTEN
0x08786160 0x00000000      0      0  :::179                 :::0                    LISTEN
0xecb0c9f8 0x60000000      0      0  10.0.0.1:12404         10.0.0.2:179            ESTAB
0x0878b168 0x60000000      0      0  11.0.0.1:179           11.0.0.2:61177          ESTAB
0xecb0c6b8 0x60000000      0      0  0.0.0.0:179            0.0.0.0:0               LISTEN
0x08781590 0x00000000      0      0  0.0.0.0:179            0.0.0.0:0               LISTEN
```

The following example shows how to display TCP connection information for a specific PCB value:

```
RP/0/0/CPU0:ios#show tcp pcb 0xecb0c9f8

Connection state is ESTAB, I/O status: 0, socket status: 0
Established at Sun Dec 7 11:49:39 2014

PCB 0xecb0c9f8, SO 0xecb01b68, TCPCB 0xecb01d78, vrfid 0x60000000,
Pak Prio: Medium, TOS: 192, TTL: 255, Hash index: 1322
Local host: 10.0.0.1, Local port: 12404 (Local App PID: 19840)
Foreign host: 10.0.0.2, Foreign port: 179

Current send queue size in bytes: 0 (max 24576)
Current receive queue size in bytes: 0 (max 32768) mis-ordered: 0 bytes
Current receive queue size in packets: 0 (max 0)

Timer Starts Wakeups Next(msec)
Retrans 17 2 0
SendWnd 0 0 0
TimeWait 0 0 0
AckHold 13 5 0
KeepAlive 1 0 0
PmtuAger 0 0 0
GiveUp 0 0 0
Throttle 0 0 0

iss: 1728179225 snduna: 1728179536 sndnxt: 1728179536
sndmax: 1728179536 sndwnd: 32517 sndcwnd: 1000
irs: 2055835995 rcvnxt: 2055836306 rcvwnd: 32536 rcvadv: 2055868842

SRTT: 206 ms, RTTO: 300 ms, RTV: 59 ms, KRTT: 0 ms
minRTT: 10 ms, maxRTT: 230 ms

ACK hold time: 200 ms, Keepalive time: 0 sec, SYN waittime: 30 sec
Giveup time: 0 ms, Retransmission retries: 0, Retransmit forever: FALSE
```

```
Connect retries remaining: 30, connect retry interval: 30 secs

State flags: none
Feature flags: Win Scale, Nagle
Request flags: Win Scale

Datagrams (in bytes): MSS 500, peer MSS 1460, min MSS 500, max MSS 1460

Window scales: rcv 0, snd 0, request rcv 0, request snd 0
Timestamp option: recent 0, recent age 0, last ACK sent 0
Sack blocks {start, end}: none
Sack holes {start, end, dups, rxmit}: none

Socket options: SO_REUSEADDR, SO_REUSEPORT, SO_NBIO
Socket states: SS_ISCONNECTED, SS_PRIV
Socket receive buffer states: SB_DEL_WAKEUP
Socket send buffer states: SB_DEL_WAKEUP
Socket receive buffer: Low/High watermark 1/32768
Socket send buffer : Low/High watermark 2048/24576, Notify threshold 0

PDU information:
#PDU's in buffer: 0
FIB Lookup Cache: IFH: 0x200 PD ctx: size: 0 data:
Num Labels: 0 Label Stack:
```

Originating Prefixes With AiGP: Example

The following is a sample configuration for originating prefixes with the AiGP metric attribute:

```
route-policy aigp-policy
  set aigp-metric 4
  set aigp-metric igp-cost
end-policy
!
router bgp 100
 address-family ipv4 unicast
  network 10.2.3.4/24 route-policy aigp-policy
  redistribute ospf osp1 metric 4 route-policy aigp-policy
 !
!
end
```

BGP Accept Own Configuration: Example

This example shows how to configure BGP Accept Own on a PE router.

```
router bgp 100
 neighbor 45.1.1.1
   remote-as 100
   update-source Loopback0
   address-family vpnv4 unicast
    route-policy pass-all in
    accept-own
    route-policy drop_111.x.x.x out
   !
   address-family vpnv6 unicast
    route-policy pass-all in
    accept-own
```

```
    route-policy drop_111.x.x.x out
   !
  !
```

This example shows an InterAS-RR configuration for BGP Accept Own.

```
router bgp 100
 neighbor 45.1.1.1
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
   route-policy rt_stitch1 in
   route-reflector-client
   route-policy add_bgp_ao out
  !
  address-family vpnv6 unicast
   route-policy rt_stitch1 in
   route-reflector-client
   route-policy add_bgp_ao out
  !
 !
extcommunity-set rt cs_100:1
  100:1
end-set
!
extcommunity-set rt cs_1001:1
  1001:1
end-set
!
route-policy rt_stitch1
  if extcommunity rt matches-any cs_100:1 then
    set extcommunity rt cs_1000:1 additive
 endif
end-policy
!
route-policy add_bgp_ao
  set community (accept-own) additive
end-policy
!
```

Configuring BGP Permanent Network

Configuring BGP Permanent Network

Perform this task to configure BGP permanent network. You must configure at least one route-policy to identify the set of prefixes (networks) for which the permanent network (path) is to be configured.

SUMMARY STEPS

1. **configure**
2. **prefix-set** *prefix-set-name*
3. **exit**
4. **route-policy** *route-policy-name*
5. **end-policy**
6. **router bgp** *as-number*
7. **address-family { ipv4 | ipv6 } unicast**
8. **permanent-network route-policy** *route-policy-name*
9. Use the **commit** or **end** command.

10. **show bgp {ipv4 | ipv6} unicast** *prefix-set*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**prefix-set** *prefix-set-name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# prefix-set PERMANENT-NETWORK-IPv4`
	`RP/0/RSP0/CPU0:router(config-pfx)# 1.1.1.1/32,`
	`RP/0/RSP0/CPU0:router(config-pfx)# 2.2.2.2/32,`
	`RP/0/RSP0/CPU0:router(config-pfx)# 3.3.3.3/32`
	`RP/0/RSP0/CPU0:router(config-pfx)# end-set`
		Enters prefix set configuration mode and defines a prefix set for contiguous and non-contiguous set of bits.

Step 3	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pfx)# exit`	Exits prefix set configuration mode and enters global configuration mode.

Step 4	**route-policy** *route-policy-name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# route-policy POLICY-PERMANENT-NETWORK-IPv4`
	`RP/0/RSP0/CPU0:router(config-rpl)# if destination in PERMANENT-NETWORK-IPv4 then`
	`RP/0/RSP0/CPU0:router(config-rpl)# pass`
	`RP/0/RSP0/CPU0:router(config-rpl)# endif`
		Creates a route policy and enters route policy configuration mode, where you can define the route policy.

Step 5	**end-policy** **Example:** `RP/0/RSP0/CPU0:router(config-rpl)# end-policy`	Ends the definition of a route policy and exits route policy configuration mode.
Step 6	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode.
Step 7	**address-family { ipv4 | ipv6 } unicast** **Example:**	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	
Step 8	**permanent-network route-policy** *route-policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af)# permanent-network route-policy POLICY-PERMANENT-NETWORK-IPv4`	Configures the permanent network (path) for the set of prefixes as defined in the route-policy.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show bgp {ipv4 | ipv6} unicast** *prefix-set* **Example:** `RP/0/RSP0/CPU0:routershow bgp ipv4 unicast`	(Optional) Displays whether the prefix-set is a permanent network in BGP.

How to Advertise Permanent Network

Perform this task to identify the peers to whom the permanent paths must be advertised.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family { ipv4 | ipv6 } unicast**
6. **advertise permanent-network**
7. Use the **commit** or **end** command.
8. **show bgp {ipv4 | ipv6} unicast neighbor** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 100`	Specifies the autonomous system number and enters the BGP configuration mode.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.255.255.254`	Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.
Step 4	**remote-as** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 4713`	Assigns the neighbor a remote autonomous system number.
Step 5	**address-family { ipv4 | ipv6 } unicast** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast`	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.
Step 6	**advertise permanent-network** **Example:** `RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise permanent-network`	Specifies the peers to whom the permanent network (path) is advertised.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 8	**show bgp {ipv4 | ipv6} unicast neighbor** *ip-address* **Example:** `RP/0/RSP0/CPU0:routershow bgp ipv4 unicast neighbor 10.255.255.254`	(Optional) Displays whether the neighbor is capable of receiving BGP permanent networks.

BGP Unequal Cost Recursive Load Balancing: Example

This is a sample configuration for unequal cost recursive load balancing:

```
interface Loopback0
 ipv4 address 20.20.20.20 255.255.255.255
!
interface MgmtEth0/RSP0/CPU0/0
 ipv4 address 8.43.0.10 255.255.255.0
!
interface TenGigE0/3/0/0
 bandwidth 8000000
 ipv4 address 11.11.11.11 255.255.255.0
 ipv6 address 11:11:0:1::11/64
!
interface TenGigE0/3/0/1
 bandwidth 7000000
 ipv4 address 11.11.12.11 255.255.255.0
 ipv6 address 11:11:0:2::11/64
!
interface TenGigE0/3/0/2
 bandwidth 6000000
 ipv4 address 11.11.13.11 255.255.255.0
 ipv6 address 11:11:0:3::11/64
!
interface TenGigE0/3/0/3
 bandwidth 5000000
 ipv4 address 11.11.14.11 255.255.255.0
 ipv6 address 11:11:0:4::11/64
!
interface TenGigE0/3/0/4
 bandwidth 4000000
 ipv4 address 11.11.15.11 255.255.255.0
 ipv6 address 11:11:0:5::11/64
!
interface TenGigE0/3/0/5
 bandwidth 3000000
 ipv4 address 11.11.16.11 255.255.255.0
 ipv6 address 11:11:0:6::11/64
!
interface TenGigE0/3/0/6
 bandwidth 2000000
 ipv4 address 11.11.17.11 255.255.255.0
 ipv6 address 11:11:0:7::11/64
!
interface TenGigE0/3/0/7
```

```
 bandwidth 1000000
 ipv4 address 11.11.18.11 255.255.255.0
 ipv6 address 11:11:0:8::11/64
!
interface TenGigE0/4/0/0
 description CONNECTED TO IXIA 1/3
 transceiver permit pid all
!
interface TenGigE0/4/0/2
 ipv4 address 9.9.9.9 255.255.0.0
 ipv6 address 9:9::9/64
 ipv6 enable
!
route-policy pass-all
  pass
end-policy
!
router static
 address-family ipv4 unicast
  202.153.144.0/24 8.43.0.1
 !
!
router bgp 100
 bgp router-id 20.20.20.20
 address-family ipv4 unicast
  maximum-paths eibgp 8
  redistribute connected
 !
 neighbor 11.11.11.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.12.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.13.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.14.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.15.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
```

```
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.16.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.17.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 11.11.18.12
  remote-as 200
  dmz-link-bandwidth
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
!
end
```

VRF Dynamic Route Leaking Configuration: Example

These examples show how to configure VRF dynamic route leaking:

Import Routes from default-VRF to non-default-VRF

```
vrf vrf_1
 address-family ipv6 unicast
  import from default-vrf route-policy rpl_dynamic_route_import
 !
end
```

Import Routes from non-default-VRF to default-VRF

```
vrf vrf_1
 address-family ipv6 unicast
    export to default-vrf route-policy rpl_dynamic_route_export
 !
end
```

Resilient Per-CE Label Mode Configuration: Example

Configuring Resilient Per-CE Label Mode Under VRF Address Family: Example

This example shows how to configure resilient per-ce label mode under VRF address family:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 666
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf-pe
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# label mode per-ce
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# end
```

Configuring Resilient Per-CE Label Mode Using a Route-Policy: Example

This example shows how to configure resilient per-ce label mode using a route-policy:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy route1
RP/0/RSP0/CPU0:router(config-rpl)# set label mode per-ce
RP/0/RSP0/CPU0:router(config-rpl)# end
```

Flow-tag propagation

The flow-tag propagation feature enables you to establish a co-relation between route-policies and user-policies. Flow-tag propagation using BGP allows user-side traffic-steering based on routing attributes such as, AS number, prefix lists, community strings and extended communities. Flow-tag is a logical numeric identifier that is distributed through RIB as one of the routing attribute of FIB entry in the FIB lookup table. A flow-tag is instantiated using the 'set' operation from RPL and is referenced in the C3PL PBR policy, where it is associated with actions (policy-rules) against the flow-tag value.

You can use flow-tag propagation to:

- Classify traffic based on destination IP addresses (using the Community number) or based on prefixes (using Community number or AS number).
- Select a TE-group that matches the cost of the path to reach a service-edge based on customer site service level agreements (SLA).
- Apply traffic policy (TE-group selection) for specific customers based on SLA with its clients.
- Divert traffic to application or cache server.

Restrictions for Flow-Tag Propagation

Some restrictions are placed with regard to using Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) and flow-tag feature together. These include:

- A route-policy can have either 'set qos-group' or 'set flow-tag,' but not both for a prefix-set.
- Route policy for qos-group and route policy flow-tag cannot have overlapping routes. The QPPB and flow tag features can coexist (on same as well as on different interfaces) as long as the route policy used by them do not have any overlapping route.
- Mixing usage of qos-group and flow-tag in route-policy and policy-map is not recommended.

Where to Go Next

For detailed information about BGP commands, see *Routing Command Reference for Cisco ASR 9000 Series Routers*

Additional References

The following sections provide references related to implementing BGP.

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Routing Command Reference for Cisco ASR 9000 Series Routers*
Cisco Express Forwarding (CEF) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*IP Addresses and Services Command Reference for Cisco ASR 9000 Series Routers*
MPLS VPN configuration information.	*MPLS Configuration Guide for Cisco ASR 9000 Series Routers*
Bidirectional Forwarding Detection (BFD)	*Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers* and *Interface and Hardware Component Command Reference for Cisco ASR 9000 Series Routers*
Task ID information.	Configuring AAA Services on Cisco ASR 9000 Series Router module of *System Security Configuration Guide for Cisco ASR 9000 Series Routers*

Standards

Standards	Title
draft-bonica-tcp-auth-05.txt	*Authentication for TCP-based Routing and Management Protocols*, by R. Bonica, B. Weis, S. Viswanathan, A. Lange, O. Wheeler
draft-ietf-idr-bgp4-26.txt	*A Border Gateway Protocol 4*, by Y. Rekhter, T.Li, S. Hares
draft-ietf-idr-bgp4-mib-15.txt	*Definitions of Managed Objects for the Fourth Version of Border Gateway Protocol (BGP-4)*, by J. Hass and S. Hares
draft-ietf-idr-cease-subcode-05.txt	*Subcodes for BGP Cease Notification Message*, by Enke Chen, V. Gillet
draft-ietf-idr-avoid-transition-00.txt	*Avoid BGP Best Path Transitions from One External to Another*, by Enke Chen, Srihari Sangli

Standards	Title
draft-ietf-idr-as4bytes-12.txt	*BGP Support for Four-octet AS Number Space*, by Quaizar Vohra, Enke Chen

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
RFC 1700	Assigned Numbers
RFC 1997	BGP Communities Attribute
RFC 2385	Protection of BGP Sessions via the TCP MD5 Signature Option
RFC 2439	BGP Route Flap Damping
RFC 2545	Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing
RFC 2796	BGP Route Reflection - An Alternative to Full Mesh IBGP
RFC 2858	Multiprotocol Extensions for BGP-4
RFC 2918	Route Refresh Capability for BGP-4
RFC 3065	Autonomous System Confederations for BGP
RFC 3392	Capabilities Advertisement with BGP-4
RFC 4271	A Border Gateway Protocol 4 (BGP-4)
RFC 4364	BGP/MPLS IP Virtual Private Networks (VPNs)

RFCs	Title
RFC 4724	*Graceful Restart Mechanism for BGP*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport