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 on the Cisco IOS XR software and complete descriptions of the BGP commands listed in this module, see Related Documents, on page 94 section of this module. To locate documentation for other commands that might appear while performing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing BGP

Release 5.0.0	This feature was introduced.
Release 5.2.3	BGP Nonstop Routing was made a default feature.

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.

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 Default Limits

Cisco IOS XR BGP imposes maximum limits on the number of neighbors that can be configured on the router and on the maximum number of prefixes that are accepted from a peer for a given address family. This limitation safeguards the router from resource depletion caused by misconfiguration, either locally or on the remote neighbor. The following limits apply to BGP configurations:

- The default maximum number of peers that can be configured is 4000. The default can be changed using the **bgp maximum neighbor** command. The *limit* range is 1 to 15000. Any attempt to configure additional peers beyond the maximum limit or set the maximum limit to a number that is less than the number of peers currently configured will fail.
- To prevent a peer from flooding BGP with advertisements, a limit is placed on the number of prefixes that are accepted from a peer for each supported address family. The default limits can be overridden through configuration of the maximum-prefix *limit* command for the peer for the appropriate address family. The following default limits are used if the user does not configure the maximum number of prefixes for the address family:
  - IPv4 Unicast: 1048576
  - IPv4 Labeled-unicast: 131072
  - IPv6 Unicast: 524288
  - IPv6 Labeled-unicast: 131072

  A cease notification message is sent to the neighbor and the peering with the neighbor is terminated when the number of prefixes received from the peer for a given address family exceeds the maximum limit (either set by default or configured by the user) for that address family.

  It is possible that the maximum number of prefixes for a neighbor for a given address family has been configured after the peering with the neighbor has been established and a certain number of prefixes have already been received from the neighbor for that address family. A cease notification message is sent to the neighbor and peering with the neighbor is terminated immediately after the configuration if the configured maximum number of prefixes is fewer than the number of prefixes that have already been received from the neighbor for the address family.

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 IOS XR Software* module of *Cisco IOS XR Routing Configuration Guide* (this publication).

Next Hop as the IPv6 Address of Peering Interface

BGP can carry IPv6 prefixes over an IPv4 session. The next hop for the IPv6 prefixes can be set through a nexthop policy. In the event that the policy is not configured, the nexthops are set as the IPv6 address of the peering interface (IPv6 neighbor interface or IPv6 update source interface, if any one of the interfaces is configured).

If the nexthop policy is not configured and neither the IPv6 neighbor interface nor the IPv6 update source interface is configured, the next hop is the IPv4 mapped IPv6 address.

Scoped IPv4 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 address families share the same gateway context, because they are registered with the IPv4 unicast table in the RIB. As a result, the global IPv4 unicast table processed when an IPv4 unicast next-hop notification is received from the RIB. A mask is maintained in the next hop, indicating the next hop belongs to IPv4 unicast. 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 labeled unicast
- IPv4 unicast
- 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 IOS XR Software* module of *Routing Command Reference for Cisco NCS 6000 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 IOS XR Software* module of .

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.

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/RP0/CPU0:router# configuration
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)#
```

Router Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router(config)# router bgp 112
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)#
```

Neighbor Configuration Mode

The following example shows how to enter neighbor configuration mode:

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Neighbor Address Family Configuration Mode

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

```
RP/0/RP0/CPU0:router(config)# router bgp 112
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-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/RP0/CPU0:router(config-bgp)# neighbor 192.23.1.2
  RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
  RP/0/RP0/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/RP0/CPU0:router(config-bgp)# neighbor 2002::2
  RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2023
  RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast
  RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self
  RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy one 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, 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/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group afmcast1 address-family ipv4 unicast
RP/0/RP0/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/RP0/CPU0:router# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group session1
RP/0/RP0/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/RP0/CPU0:router(config)# router bgp 123
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)#
```

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

```
RP/0/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group nbrgroup1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/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/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.1.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# advertisement-interval 20
```

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

```
RP/0/RP0/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/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 20
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use session-group AS_2
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# router bgp 150
RP/0/RP0/CPU0:router(config-bgp)# session-group AS_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 20
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/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/RP0/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/RP0/CPU0:routerconfig)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# session-group ADV
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group ADV_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 25
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# exit
```

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.2.2
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbr)# use session-group ADV
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group adv_15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 10
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.101.5
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group AS_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.101.10
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# router bgp 142
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# next-hop-self
RP/0/RP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# advertisement-interval 15
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# ebgp-multihop 3
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# weight 100
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit

RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2
RP/0/RP0/CPU0:router(config-bgp-nbr)# use neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# use af-group GROUP_3
```

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# weight 200
```

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/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RP0/CPU0:router(config-bgp-afgrp)# route-policy POLICY_1 in
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_1 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-afgrp)# maximum-prefix 2500 75 warning-only
RP/0/RP0/CPU0:router(config-bgp-afgrp)# default-originate
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RP0/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/RP0/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/RP0/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/RP0/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/RP0/CPU0:router(config)# router bgp 113
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# update-source Loopback 0
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# use session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# ebgp-multihop 2
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RP0/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 XR EXEC mode:

```
RP/0/RP0/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/RP0/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/RP0/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/RP0/CPU0:router(config)# router bgp 140
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# remove-private-as
RP/0/RP0/CPU0:router(config-bgp-afgrp)# soft-reconfiguration inbound
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# af-group GROUP_2 address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-afgrp)# use af-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# send-extended-community-ebgp
RP/0/RP0/CPU0:router(config-bgp-afgrp)# capability orf prefix both
RP/0/RP0/CPU0:router(config-bgp-afgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-sngrp)# timers 30 90
RP/0/RP0/CPU0:router(config-bgp-sngrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_1
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# remote-as 1982
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit
RP/0/RP0/CPU0:router(config-nbrgrp)# exit
RP/0/RP0/CPU0:router(config-bgp)# neighbor-group GROUP_2
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# use session-group GROUP_3
RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast
RP/0/RP0/CPU0:routerconfig-bgp-nbrgrp-af)# use af-group GROUP_2
RP/0/RP0/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/RP0/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/RP0/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/RP0/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 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/RP0/CPU0:router(config)# route-policy pass-all
RP/0/RP0/CPU0:router(config-rpl)# pass
RP/0/RP0/CPU0:router(config-rpl)# end-policy
RP/0/RP0/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/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# neighbor 192.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 21
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RP0/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/RP0/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
```

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 IOS XR Software* module in the *Routing Configuration Guide for Cisco NCS 6000 Series Routers* for details on table policy use. See the *Cisco Express Forwarding Commands on Cisco IOS XR Software* module in the *IP Addresses and Services Command Reference for Cisco NCS 6000 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 19
BGP Update Group , on page 20

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 IOS XR Software* module of *Cisco IOS XR 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 22.

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 23 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

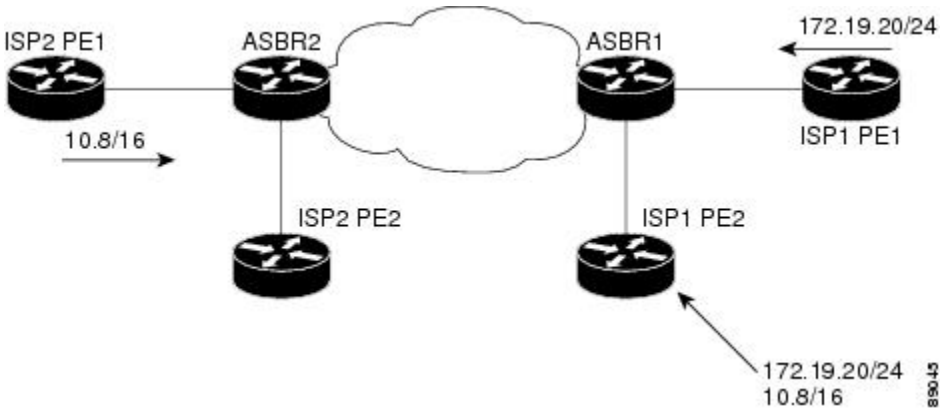
  The single advertised path includes the aggregate cost communities as follows:

  {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/RP0/CPU0:router(config)# route-policy ISP2_PE1
RP/0/RP0/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.

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 23 for information on the BGP best-path algorithm.

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_61 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_61.
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_61.
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_61.
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_61, 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 NCS 6000 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 1: BGP Default Administrative Distances, on page 27.

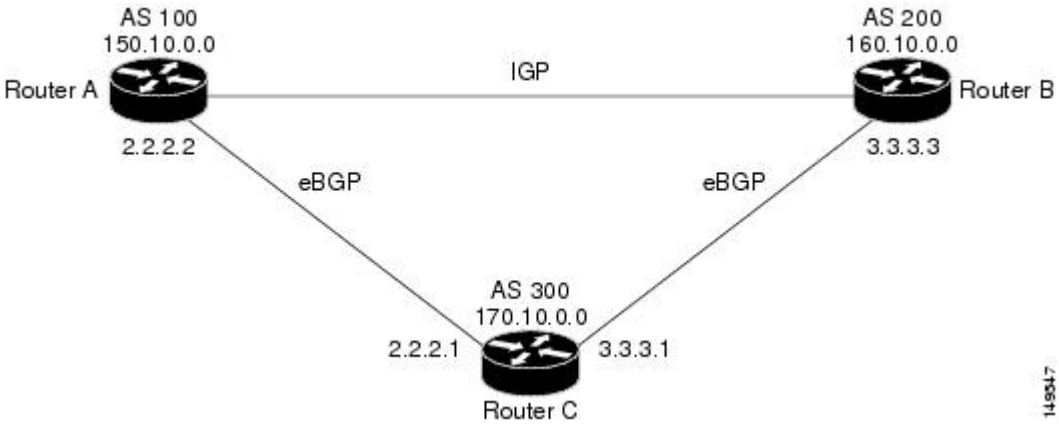
Table 1: BGP Default Administrative Distances

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 2: Back Door Example , on page 28.

Figure 2: Back Door Example



In Figure 2: Back Door Example , on page 28, 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/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/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.

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

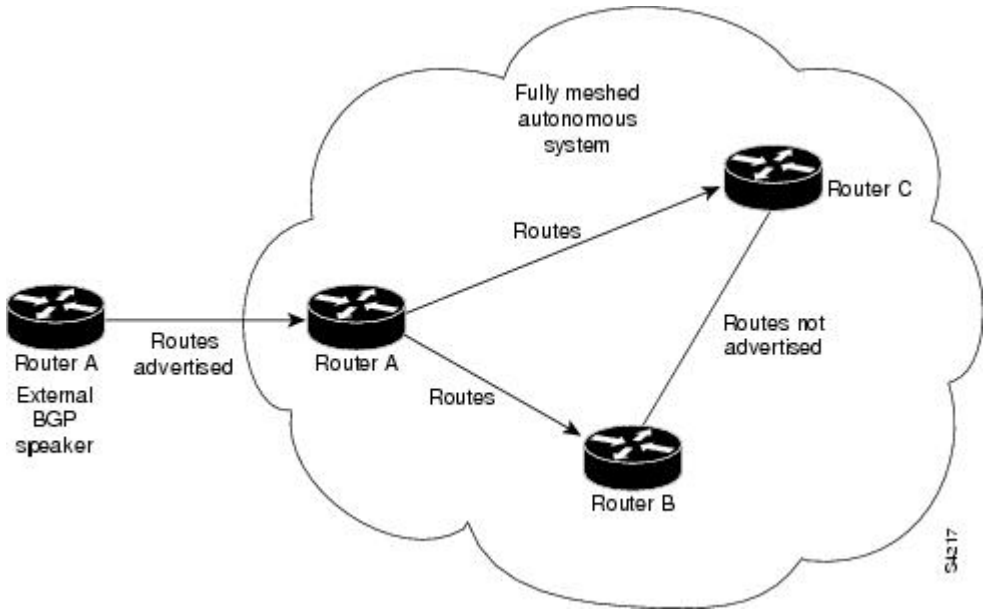
One way to reduce the iBGP mesh is to divide an autonomous system into multiple subautonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections 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.

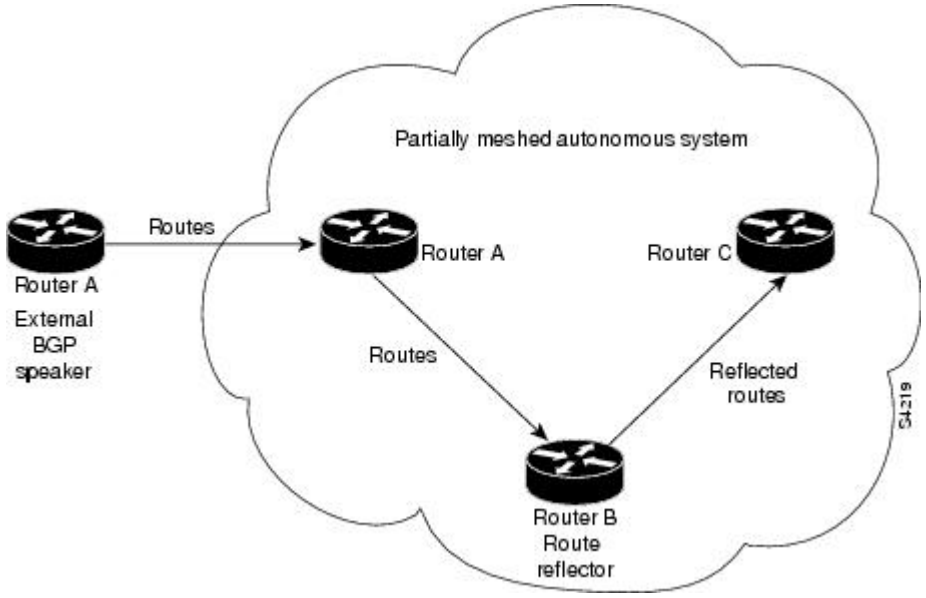
Figure 3: Three Fully Meshed iBGP Speakers, on page 30 illustrates a simple iBGP configuration with three iBGP speakers (routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise 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 3: 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 4: Simple BGP Model with a Route Reflector, on page 30 , 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 4: 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 5: More Complex BGP Route Reflector Model

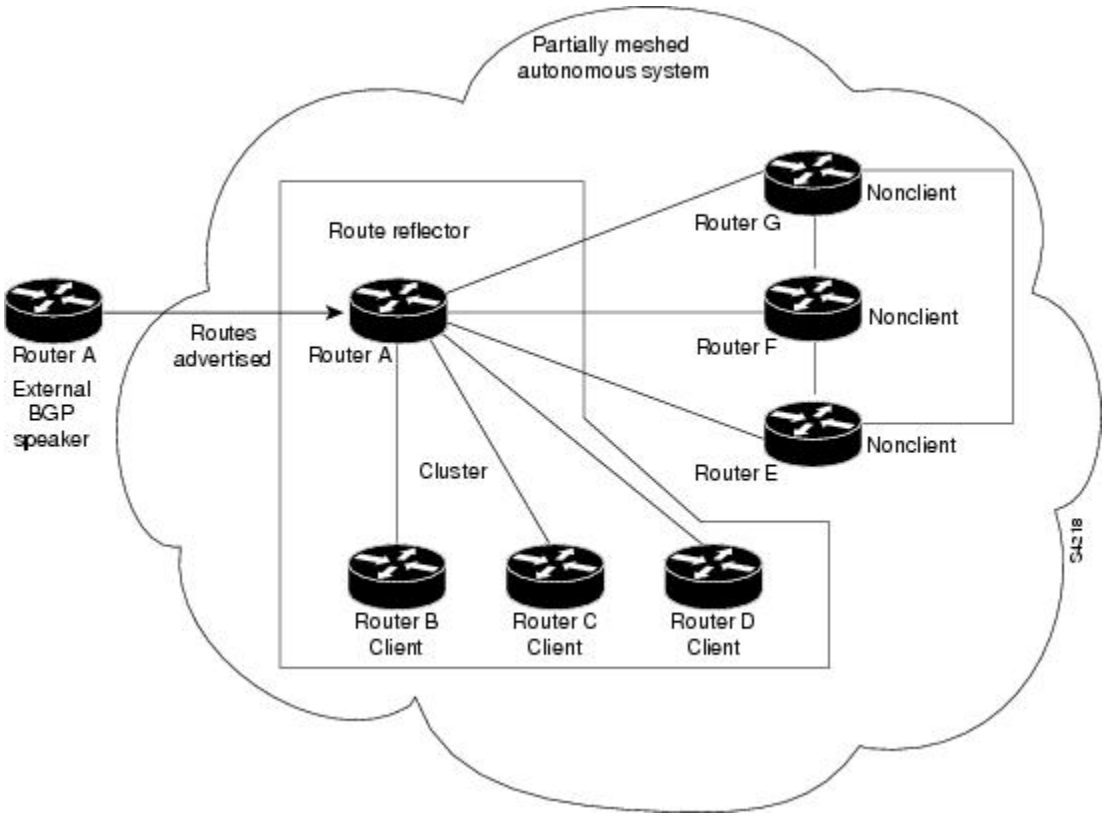


Figure 5: More Complex BGP Route Reflector Model, on page 31 illustrates a more complex route reflector scheme. Router A is the route reflector in a cluster with routers B, C, and D. Routers E, F, and G are fully meshed, nonclient routers.

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.

Remotely Triggered Blackhole 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

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.

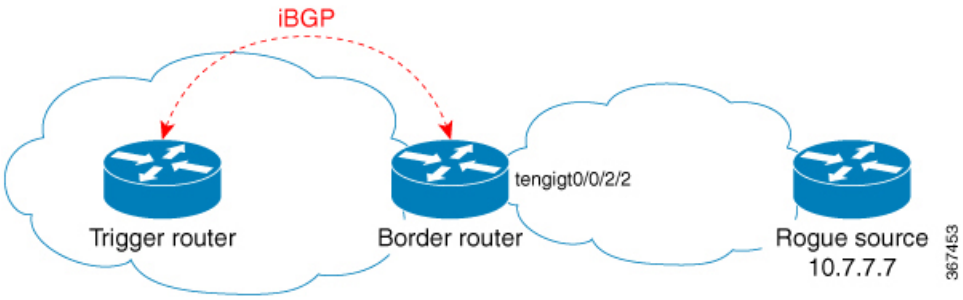
RTBH filtering sets the next-hop of the victim's prefix to the null interface. The traffic destined to the victim is dropped at the ingress.

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 6: 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**

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.

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 NCS 6000 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 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.

- Minimum Disruption Restart (MDR)

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

NSR can be provisioned on a multishelf router. The following guidelines should be observed when provisioning NSR on a multishelf router:

- When provisioning NSR for line cards installed on a single rack, provision the active and standby applications on the distributed route processor (DRP) of that rack. If a rack failure occurs, sessions are dropped, because all line cards go down.
- When provisioning NSR for line cards installed on different racks, use one of the following three options:
  - Provision the active and standby applications on a distributed route processor (DRP) redundant pair, where there is a separate route processor in each rack. This configuration uses up two revenue-producing line-card slots on each rack, but is the most secure configuration.
  - Provision the active and standby applications on a distributed route processor (DRP) pair that spans two racks. In this configuration, the active/standby role of the line cards is not dependent on the active/standby role of the DRPs. This is called *flexible process redundancy* and provides for rack loss and efficient use of LC slots. Use of distributed BGP is not required with this solution.

Note Sessions on line cards in a lost rack are not protected with any of the above options, because there is no line-card redundancy. These options ensure only that sessions on other racks are not affected by a lost rack. However, lost sessions from a lost rack may cause some traffic loss on other racks, because destinations learned through those lost sessions may no longer have alternate routes. Also, rack loss may cause the CPUs on route processors of active racks to slow as they attempt to define new paths for some routes.

BGP Best-External Path

The Border Gateway Protocol (BGP) best–external path functionality supports advertisement of the best–external path to the iBGP and Route Reflector peers when a locally selected bestpath is from an internal peer.

BGP selects one best path and one backup path to every destination. By default, selects one best path . Additionally, BGP selects another bestpath from among the remaining external paths for a prefix. Only a single path is chosen as the best–external path and is sent to other PEs as the backup path.

BGP calculates the best–external path only when the best path is an iBGP path. If the best path is an eBGP path, then best–external path calculation is not required.

The procedure to determine the best–external path is as follows:

1. Determine the best path from the entire set of paths available for a prefix.
2. Eliminate the current best path.
3. Eliminate all the internal paths for the prefix.
4. From the remaining paths, eliminate all the paths that have the same next hop as that of the current best path.
5. Rerun the best path algorithm on the remaining set of paths to determine the best–external path.

BGP considers the external and confederations BGP paths for a prefix to calculate the best–external path.

BGP advertises the best path and the best–external path as follows:

- On the primary PE—advertises the best path for a prefix to both its internal and external peers
- On the backup PE—advertises the best path selected for a prefix to the external peers and advertises the best–external path selected for that prefix to the internal peers

The **advertise best-external** command enables the advertisement of the best–external path in global address family configuration mode.

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.

BGP Over GRE Interfaces

Cisco IOS XR software provides the capability to run Border Gateway Protocol (BGP) over Generic Routing Encapsulation (GRE) tunnel interfaces.

GRE protocol transports packets of one protocol over another protocol by means of encapsulation. Service Providers can provide IP services between their networks that are connected together by a public network using GRE encapsulation to carry data securely over the public network.

The packet that needs to be transported is first encapsulated in a GRE header, which is further encapsulated in another protocol like IPv4 or IPv6 and then forwarded to the destination.

The Cisco IOS XR software GRE implementation is compliant with GRE encapsulation defined in RFC 2784. Key and Sequence numbering as defined in RFC 2890 is not supported in Cisco IOS XR software GRE implementation. To be backward compliant with RFC 1701, Cisco IOS XR software transmits GRE packets with Reserved0 field set to zero. A receiver that is compliant with RFC 1701 treats key present, sequence number, and strict source route as zero and do not expect key and sequence number. The Cisco IOS XR software discards a GRE packet with any of the bits in Reserved0 field set.

Command Line Interface (CLI) Consistency for BGP Commands

TheBorder 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.

Note BGP Additional Path feature is not supported under vrf.

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.

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.

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

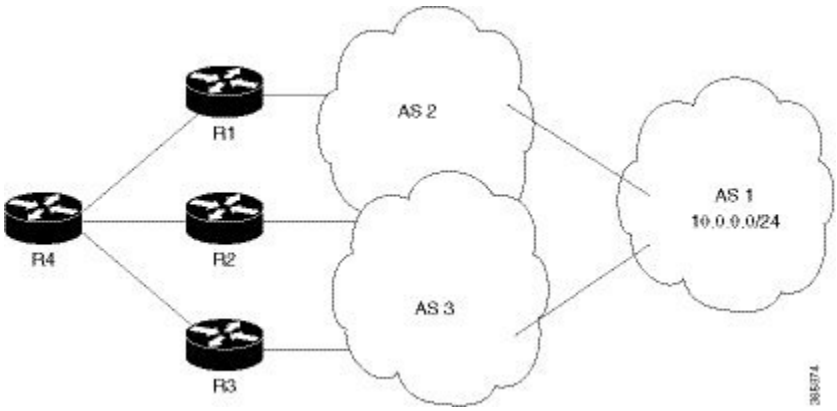
- BGP selective multipath does not impact best path calculations. A best path is always included in the set of multipaths.
- For VPN prefixes, the PE paths are *always* eligible to be multipaths.

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 7: 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/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths ibgp 4 selective
RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths ebgp 5 selective
RP/0/RP0/CPU0:router(config-bgp-af)# commit

/* To configure selective multipath for eiBGP
RP/0/RP0/CPU0:router(config)# router bgp 1
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)# maximum-paths eibgp 6 selective
RP/0/RP0/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/RP0/CPU0:router(config-bgp)# neighbor 1.1.1.1
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit

RP/0/RP0/CPU0:router(config-bgp-nbr)# neighbor 2.2.2.2
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# multipath
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit

RP/0/RP0/CPU0:router(config-bgp-nbr)# neighbor 3.3.3.3
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RP0/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.

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 NCS 6000 Series Routers* and *Interface and Hardware Component Command Reference for the Cisco NCS 6000 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 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 black hole in the network. As a workaround, you must route the traffic to the backup path manually before configuring the **neighbor shutdown** command.

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 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

- Directly connected routes cannot be leaked using BGP VRF Dynamic Route Leaking from default VRF to non-default 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.

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. **commit**
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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**route-policy** *route-policy-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# route-policy drop-as-1234`
	`  RP/0/RP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then`
	`  RP/0/RP0/CPU0:router(config-rpl)# apply check-communities`
	`  RP/0/RP0/CPU0:router(config-rpl)# else`
	`  RP/0/RP0/CPU0:router(config-rpl)# pass`
	`  RP/0/RP0/CPU0:router(config-rpl)# endif`
		(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

	Command or Action	Purpose
Step 3	**end-policy** **Example:** `RP/0/RP0/CPU0:router(config-rpl)# end-policy`	(Optional) Ends the definition of a route policy and exits route policy configuration mode.
Step 4	**commit**	
Step 5	**configure**	
Step 6	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 10	**neighbor** *ip-address* **Example:** `RP/0/RP0/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/RP0/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/RP0/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 (?).

	Command or Action	Purpose
Step 13	**route-policy** *route-policy-name* { **in** | **out** } **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in`	(Optional) Applies the specified policy to inbound IPv4 unicast routes.
Step 14	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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:** `RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5`	Specifies a BGP confederation identifier.

Step 4	**bgp confederation peers** *as-number*
	Example:
	`RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1091`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1092`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1093`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1094`
	`RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 1095`
	`RP/0/RP0/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	**commit**	

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/RP0/CPU0:router#configure
RP/0/RP0/CPU0:router(config)#lpts pifib hardware police location 0/2/CPU0
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#flow bgp configured rate 4000
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#flow bgp known rate 4000
RP/0/RP0/CPU0:router(config-pifib-policer-per-node)#flow bgp default rate 4000
RP/0/RP0/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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-bgp)# timers bgp 30 90`	Sets a default keepalive time and a default hold time for all neighbors.

	Command or Action	Purpose
Step 4	**neighbor** *ip-address* **Example:** `RP/0/RP0/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	**timers** *keepalive hold-time* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# timers 60 220`	(Optional) Sets the keepalive timer and the hold-time timer for the BGP neighbor.
Step 6	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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 default local-preference** *value* **Example:** `RP/0/RP0/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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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	**default-metric** *value* **Example:** `RP/0/RP0/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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-nbr-af)# weight 41150`	Assigns a weight to all routes learned through the neighbor.
Step 7	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/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:** `RP/0/RP0/CPU0:router(config-bgp)# bgp bestpath as-path ignore`	Configures the BGP software to ignore the autonomous system length when performing best-path selection.
Step 7	**bgp bestpath compare-routerid** **Example:** `RP/0/RP0/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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)# network 172.20.0.0/16`	Configures the local router to originate and advertise the specified network.
Step 5	**commit**	

Configuring Aggregate Addresses

Perform this task to create aggregate entries in a BGP routing table.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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:** `RP/0/RP0/CPU0:router(config-bgp)# bgp redistribute-internal`	Allows the redistribution of iBGP routes into an IGP, such as IS-IS or OSPF.
Step 4	**commit**	

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.

Note BGP doesn't support redistribution of ISIS routes in VRF.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)# redistribute ospf 110`	Causes routes from the specified instance to be redistributed into BGP.
Step 5	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120`	Configures BGP dampening for the specified address family.
Step 5	**commit**	

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* module of *Routing Configuration Guide for Cisco NCS 6000 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)# table-policy tbl-plcy-A`	Applies the specified policy to routes being installed into the routing table.
Step 5	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)# exit`	Exits the current configuration mode.
Step 5	**neighbor-group** *name* **Example:** `RP/0/RP0/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/RP0/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:**	Specifies either an IPv4 or IPv6 address family unicast and enters address family configuration submode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# address-family ipv4 unicast`	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:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# route-policy drop-as-1234 in`	(Optional) Applies the specified policy to inbound IPv4 unicast routes.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp-af)# exit`	Exits the current configuration mode.
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbrgrp)# exit`	Exits the current configuration mode.
Step 11	**neighbor** *ip-address* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 14	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2003`	Creates a neighbor and assigns a remote autonomous system number to it.
Step 6	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RP0/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 (?).

	Command or Action	Purpose
Step 7	**route-reflector-client** **Example:** `RP/0/RP0/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	**commit**	

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* module of *Cisco 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**route-policy** *name*
	Example:
	`RP/0/RP0/CPU0:router(config)# route-policy drop-as-1234`
	`  RP/0/RP0/CPU0:router(config-rpl)# if as-path passes-through '1234' then`
	`  RP/0/RP0/CPU0:router(config-rpl)# apply check-communities`
	`  RP/0/RP0/CPU0:router(config-rpl)# else`
	`  RP/0/RP0/CPU0:router(config-rpl)# pass`
	`  RP/0/RP0/CPU0:router(config-rpl)# endif`
		(Optional) Creates a route policy and enters route policy configuration mode, where you can define the route policy.

	Command or Action	Purpose
Step 3	**end-policy** **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-nbr-af)# route-policy drop-as-1234 in`	Applies the specified policy to inbound routes.
Step 8	**commit**	

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**	

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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. • 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-bgp-af)# nexthop trigger-delay critical 15000`	Sets the critical next-hop trigger delay.
Step 5	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/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/RP0/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	**commit**	

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 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** | **ipv6** {**labeled-unicast** | **unicast**}}
6. Use one of these commands:
   - **send-community-ebgp**
   - **send-extended-community-ebgp**
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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** | **ipv6** {**labeled-unicast** | **unicast**}} **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 unicast`	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: • **labeled-unicast** • **unicast** IPv4 address family mode supports these sub modes: • **labeled-unicast** • **unicast** Refer the **address-family (BGP)** command in *BGP Commands* module of *Routing Command Reference for Cisco NCS 6000 Series Routers* for more information on the Address Family Submode support.
Step 6	Use one of these commands: • **send-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.

	Command or Action	Purpose

	• **send-extended-community-ebgp**
	Example:
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-community-ebgp`
	or
	`RP/0/RP0/CPU0:router(config-bgp-nbr-af)# send-extended-community-ebgp`	

Step 7	**commit**	

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** | **ipv6** } **unicast** **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4** | **ipv6** } **unicast** **redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4** | **ipv6** } **unicast** **redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **address-family** { **ipv4** | **ipv6** } **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** | **ipv6** } **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** | **ipv6** } **unicast** **redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*]

- **address-family** { **ipv4** | **ipv6** } **unicast redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **address-family** { **ipv4** | **ipv6** } **unicast network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
- **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4** | **ipv6** } **unicast**
- **route-policy** *route-policy-name* { **in** | **out** }

8. **commit**
9. **show bgp** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**route-policy** *name* **Example:** `RP/0/RP0/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/RP0/CPU0:router(config)# set extcommunity cost cost_A`	Specifies the BGP extended community attribute for cost.
Step 4	**end-policy** **Example:** `RP/0/RP0/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/RP0/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** | **ipv6** } **unicast redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute eigrp** *process-id* [**match** { **external** | **internal** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]	Applies the cost community to the attach point (route policy).

	Command or Action	Purpose
	• **address-family** { **ipv4** | **ipv6** } **unicast redistribute isis** *process-id* [**level** { **1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **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** | **ipv6** } **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** | **ipv6** } **unicast redistribute rip** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] • **address-family** { **ipv4** | **ipv6** } **unicast network** { *ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*] • **neighbor** *ip-address* **remote-as** *as-number* **address-family** { **ipv4** | **ipv6** } **unicast** • **route-policy** *route-policy-name* { **in** | **out** }	
Step 8	**commit**	
Step 9	**show bgp** *ip-address* **Example:** `RP/0/RP0/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 75.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/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

	Command or Action	Purpose
		stored). The **always** keyword forces the software to store a copy even when route refresh is supported by the peer.
Step 6	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns a remote autonomous system number to it.

	Command or Action	Purpose
Step 5	**keychain** *name* **Example:** `RP/0/RP0/CPU0:router(config-bgp-nbr)# keychain kych_a`	Configures keychain-based authentication.
Step 6	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-nbr)# shutdown`	Disables all active sessions for the specified neighbor.
Step 5	**commit**	

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 | ipv6**} { **unicast** | **labeled-unicast** } **soft** [**in** [**prefix-filter**] | **out**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp neighbors** **Example:** `RP/0/RP0/CPU0:router# show bgp neighbors`	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	**clear bgp** { **ipv4	ipv6**} { **unicast**	**labeled-unicast** } **soft** [**in** [**prefix-filter**]	**out**] **Example:** `RP/0/RP0/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. **out**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp neighbors** **Example:** `RP/0/RP0/CPU0:router# show bgp neighbors`	Verifies that received route refresh capability from the neighbor is enabled.
Step 2	**out** **Example:** `RP/0/RP0/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. }

DETAILED STEPS

	Command or Action	Purpose

Step 1	}
	Example:
	`RP/0/RP0/CPU0:router# clear bgp ipv4 unicast 10.0.0.3`
		Clears a BGP neighbor.

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** | **labeled-unicast** } | **ipv6** { **unicast** | **labeled-unicast** }
2. **clear bgp external**
3. **clear bgp ***

DETAILED STEPS

	Command or Action	Purpose
Step 1	**clear bgp** { **ipv4** { **unicast** | **labeled-unicast** } | **ipv6** { **unicast** | **labeled-unicast** } **Example:** `RP/0/RP0/CPU0:router# clear bgp ipv4 172.20.1.1`	Clears a specified neighbor.
Step 2	**clear bgp external** **Example:** `RP/0/RP0/CPU0:router# clear bgp external`	Clears all external peers.
Step 3	**clear bgp *** **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/CPU0:router# show bgp regexp "^3 "`	Displays routes that match the specified autonomous system path regular expression.
Step 4	**show bgp** **Example:** `RP/0/RP0/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/RP0/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.
Step 6	**show bgp paths** **Example:** `RP/0/RP0/CPU0:router# show bgp paths`	Displays all BGP paths in the database.

	Command or Action	Purpose
Step 7	**show bgp neighbor-group** *group-name* **configuration** **Example:** `RP/0/RP0/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/RP0/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 process detail**
4. **show bgp summary**
5. **show placement program bgp**
6. **show placement program brib**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp process** **Example:** `RP/0/RP0/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/RP0/CPU0:router# show bgp ipv4 unicast summary`	Displays a summary of the neighbors for the IPv4 unicast address family.
Step 3	**show bgp process detail** **Example:** `RP/0/RP0/CPU0:router# show bgp processes detail`	Displays detailed process information including the memory used by each of various internal structure types.
Step 4	**show bgp summary** **Example:**	Displays the status of all BGP connections.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# show bgp summary`	
Step 5	**show placement program bgp** **Example:** `RP/0/RP0/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 6	**show placement program brib** **Example:** `RP/0/RP0/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 update-group** [**neighbor** *ip-address* | *process-id.index* [**summary** | **performance-statistics**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show bgp update-group** [**neighbor** *ip-address*	*process-id.index* [**summary**	**performance-statistics**]] **Example:** `RP/0/RP0/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.

	Command or Action	Purpose
		• 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-bgp)# nsr disable`	Disables BGP Nonstop routing.
Step 4	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-bgp)# nsr disable`	Enables BGP Nonstop routing.
Step 4	**commit**	

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. **address-family ipv4 unicast** | **ipv6 unicast** }
8. **additional-paths receive**
9. **additional-paths send**
10. **additional-paths selection route-policy** *route-policy-name*
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	**route-policy** *route-policy-name* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-rpl-else)#pass`
	`RP/0/RP0/CPU0:router(config-rpl-else)#endif`
		Passes the route for processing and ends the if statement.

Step 5	**end-policy** **Example:** `RP/0/RP0/CPU0:router(config-rpl)#end-policy`	Ends the route policy definition of the route policy and exits route-policy configuration mode.
Step 6	**router bgp** *as-number* **Example:** `RP/0/RP0/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	**address-family ipv4 unicast** | **ipv6 unicast** } **Example:** `RP/0/RP0/CPU0:router(config-bgp)#address-family ipv4 unicast`	Specifies the address family and enters address family configuration submode.
Step 8	**additional-paths receive** **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)#additional-paths receive`	Configures receive capability of multiple paths for a prefix to the capable peers.
Step 9	**additional-paths send** **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)#additional-paths send`	Configures send capability of multiple paths for a prefix to the capable peers .
Step 10	**additional-paths selection route-policy** *route-policy-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp-af)#additional-paths selection route-policy add_path_policy`	Configures additional paths selection capability for a prefix.
Step 11	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**route-policy** *aigp_policy* **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-rpl)# set aigp-metric igp-cost`	Sets the internal routing protocol cost as the aigp metric.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-rpl)# exit`	Exits route-policy configuration mode.

	Command or Action	Purpose
Step 5	**router bgp** *as-number* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp-af)#redistribute ospf osp route-policy aigp_policy metric 1`	Allows the redistribution of AiBGP metric into OSPF.
Step 8	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**vrf** *vrf_name* **Example:** `RP/0/RSP0/CPU0:PE51_-9010(config)#vrf vrf_1`	Enters VRF configuration mode.

	Command or Action	Purpose
Step 3	**address-family** {**ipv4** | **ipv6**} **unicast** **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-vrf-af)#import from default-vrf route-policy rpl_dynamic_route_import`
	or
	`RP/0/RP0/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	**commit**	

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.

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 1.1.1.1
  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 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

  neighbor 10.0.101.60
    remote-as 65000
    address-family ipv4 unicast

  neighbor 10.0.101.61
    remote-as 65000
    address-family ipv4 unicast

  neighbor 10.0.101.62
    remote-as 3
    address-family ipv4 unicast
      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
```

Displaying BGP Update Groups: Example

The following is sample output from the **show bgp update-group** command run in XR EXEC 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 131. 108.0.0 and 192. 31.7.0 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 131. 108.234.2; 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 131..0.0 255..0.0
  network 192.31.7.0 255.0.0.0
  neighbor 131.108.200.1
   remote-as 167
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-out out
  neighbor 131.108.234.2
   remote-as 109
   exit
 address-family ipv4 unicast
  neighbor 150.136.64.19
   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. 69.232.55 and 171. 69.232.56 get the local preference, next hop, and MED unmodified in the updates. The router at 160. 69.69.1 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 171.69.232.55
  remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 171.69.232.56
  remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 160.69.69.1
  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 170. 70.70.1 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 170.70.70.1
   remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 171.69.232.57
   remote-as 6001
   exit
 address-family ipv4 unicast
  neighbor 171.69.232.56
   remote-as 6003
   exit
```

```
 address-family ipv4 unicast
  neighbor 199.69.99.2
   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 200. 200.200.200 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 171.69.232.57
   remote-as 6001
   exit
 address-family ipv4 unicast
  neighbor 171.69.232.55
   remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 200.200.200.200
   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 200. 200.200.205 from autonomous system 701 in the same example. Neighbor 171. .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 171.69.232.56
   remote-as 666
   exit
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy pass-all out
   exit
 address-family ipv4 unicast
  neighbor 200.200.200.205
   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:

```
router bgp 140
 address-family ipv4 unicast
  neighbor 10.1.1.1
   remote-as 140
   address-family ipv4 unicast
    route-reflector-client
    exit
```

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
```

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
```

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
```

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 NCS 6000 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 NCS 6000 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 NCS 6000 Series Routers*
Bidirectional Forwarding Detection (BFD)	*Interface and Hardware Component Configuration Guide for Cisco NCS 6000 Series Routers* and *Interface and Hardware Component Command Reference for the Cisco NCS 6000 Series Routers*
Task ID information.	Configuring AAA Services on module of *System Security Configuration Guide for Cisco NCS 6000 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

Standards	Title
draft-ietf-idr-avoid-transition-00.txt	*Avoid BGP Best Path Transitions from One External to Another*, by Enke Chen, Srihari Sangli
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)

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