



BGP Dynamic Neighbors and Resource Management

This chapter explains how to configure, manage, and monitor BGP dynamic neighbors and resource allocation. It covers advanced neighbor management, such as managing dynamic peering, configuring multi-instance BGP, resetting neighbors, and managing resource allocation.

- [BGP dynamic neighbors, on page 1](#)
- [Multi-instance and multi-AS BGP, on page 5](#)
- [BGP neighbor resets, on page 6](#)
- [Clear BGP caches, tables, and databases, on page 8](#)
- [Disable a BGP neighbor, on page 8](#)
- [Neighbor capability suppressions, on page 9](#)
- [Display BGP system and network statistics, on page 10](#)

BGP dynamic neighbors

A BGP dynamic neighbor is a peer that is discovered and established using a configured IP address range, instead of an explicitly configured or static neighbor configuration.

Dynamic neighbor configuration allows a router to peer with multiple remote neighbors without explicit static entries for each one. The router accepts BGP connections from any peer whose IP address falls within a specified subnet range.

Example:

In a data center, you can configure one dynamic neighbor range to handle a pool of servers or routers that may change frequently.

Benefits of BGP dynamic neighbor support

BGP dynamic neighbor support provides these benefits:

- Allows you to configure each range of IP addresses as a subnet.
- Reduces CLI configuration complexity in large networks.
- Supports both IPv4 and IPv6 peering.

Configure BGP dynamic neighbors using address range

Purpose: Configure BGP dynamic neighbors using an IP prefix, reducing manual neighbor statements in large-scale networks.

Context: Use this when you want to accept BGP peerings from a set of IP addresses, such as a subnet, instead of individually configuring each neighbor.

This figure illustrates a sample topology.

Figure 1: Dynamic neighbor connection topology



After you configure the subnet range for dynamic neighbors on Router A, Router B can initiate a TCP session from any IP address within that range. Router A then automatically establishes the BGP neighbor connection with Router B, with no further CLI configuration required after the initial setup.

In this example, local autonomous system (AS) is 100 and remote AS is 1.

Follow these steps to configure BGP dynamic neighbors using an address range.

Before you begin

- Identify the IP subnet that should be allowed as dynamic neighbors.
- Ensure you know the remote AS number and the local interface for source IP.

Procedure

Step 1 Enter BGP configuration mode with your AS number and configure the dynamic neighbor prefix.

Example:

```
Router#configure
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.0.0.0/16
```

Step 2 Assign the remote AS number and specify the update source interface.

Example:

```
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# update-source FourHundredGige 0/0/0/0
```

Step 3 Enable the address family for IPv4 unicast and commit the configuration.

Example:

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

Step 4 Verify the active configuration.

Example:

```
Router# show running-config
router bgp 100
 neighbor 10.0.0.0/16
   remote-as 1
   update-source FourHundredGige 0/0/0/0
 address-family ipv4 unicast
!
```

The router accepts BGP sessions from any peer with an IP address within the specified range, creating dynamic neighbor entries automatically.

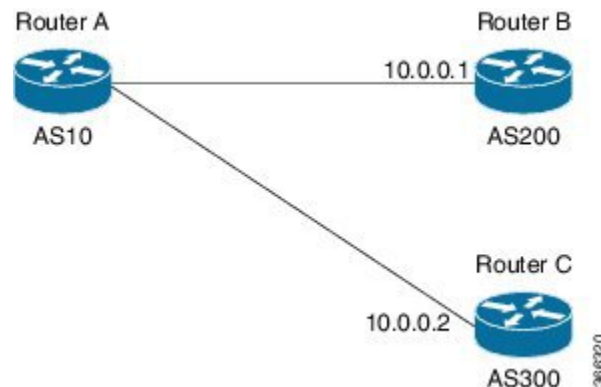
Configure remote AS list

Configure a list of remote autonomous system (AS) numbers for dynamic BGP neighbors.

Context: Use this procedure when you have dynamic neighbors from multiple remote ASes and want to control which AS numbers are accepted.

This figure illustrates a sample topology.

Figure 2: Connection topology with dynamic neighbors from multiple remote ASes



In this example topology, Router B and Router C are configured as remote BGP peers. Both Router B and Router C are in different autonomous systems.

A list of the autonomous systems of the remote routers is created and the list is then configured in Router A under neighbor mode using the **remote-as-list** command.

Before you begin

- Identify the IP subnet that should be allowed as dynamic neighbors.
- Ensure you know the remote AS number and the local interface for source IP.

Procedure

Step 1 Enter BGP configuration mode and define your local AS number.

Example:

```
Router#configure
Router(config)# router bgp 100
Router(config-bgp)#
```

Step 2 Run the **as-list** command to create an AS list.

Example:

```
Router(config-bgp)#as-list test
Router(config-bgp-as-list)#200
Router(config-bgp-as-list)#300
Router(config-bgp-as-list)#exit
Router(config-bgp)#
```

Step 3 Run the **neighbor address/prefix** command to configure the dynamic neighbor prefix.

Example:

```
Router(config-bgp)# neighbor 10.0.0.0/16
```

Step 4 Run the **remote-as-list** command to assign the remote AS list to the neighbor.

Example:

```
Router(config-bgp-nbr)# remote-as-list test
```

Step 5 Enable the address family for IPv4 unicast and commit the configuration.

Example:

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

Dynamic neighbor sessions are accepted only if their AS number is in the configured remote AS list.

Configure maximum peers and idle-watch timeout

Limit the number of dynamic BGP neighbors and set the idle timeout for TCP instances.

Context: Use these settings to manage router resources and avoid excessive neighbor creation.

Before you begin

None

Procedure

Step 1 Enter BGP configuration mode with your AS number and configure the dynamic neighbor prefix.

Example:

```
Router#configure
Router(config)# router bgp 100
Router(config-bgp)# neighbor 10.0.0.0/16
```

Step 2 Run the **maximum-peers** command to set the maximum number of dynamic neighbors for this neighbor address range.

Example:

```
Router(config-bgp-nbr) # maximum-peers 16
```

Step 3 Run the **idle-watch-time** command to set the idle-watch timeout in seconds and save the configuration.

Example:

```
Router(config-bgp-nbr) # idle-watch-time 120
Router(config-bgp-nbr) # commit
```

The router limits dynamic neighbor creation for the range and removes idle TCP sessions after the specified time.

Multi-instance and multi-AS BGP

Multi-instance and multi-AS BGP is a configuration approach that

- allows each BGP instance to use a unique AS number
- provides address family isolation by mapping different address families to separate BGP instances, and
- enables higher scaling and more granular resource management by distributing peer sessions and BGP tables across instances.

Restrictions and guidelines for multi-instance BGP

This section outlines the restrictions and provides guidelines for configuring and managing multi-instance BGP on a router.

Instance limits and identification

- A router supports a maximum of four BGP instances.
- Each BGP instance must have a unique router ID.

Address family configuration

- Only one address family can be configured under each BGP instance, except VPNv4, VPNv6, and RT-Constrain.
- IPv4/IPv6 unicast and their labeled variants must reside in the same BGP instance.
- IPv4/IPv6 multicast must reside in the same BGP instance as their unicast counterparts.

Configuration management

All configuration changes for a single instance can be committed together; however, changes for multiple instances must be committed separately.

Guidelines

We recommend using unique update-source interfaces in the default VRF over all instances when peering with the same remote router.

BGP neighbor resets

BGP neighbor reset is an administrative process that

- refreshes or clears BGP sessions to apply configuration changes
- can be performed as a soft or hard reset, and
- helps maintain accurate routing information by forcing policy re-evaluation or session re-establishment.

These resets are primarily categorized into two types, each with a distinct impact on the BGP session:

- Soft reset: Resets only routing information without tearing down the TCP session.
- Hard reset: Tears down the TCP session and removes all routes learned from the neighbor.

Reset BGP neighbors using inbound soft reset

Refresh inbound BGP policies and routes without resetting the entire session.

Context: Use this when inbound policy or attribute changes need to be applied to BGP-learned routes.

Before you begin

None

Procedure

Step 1 Run the **show bgp neighbors** command to verify the route refresh capability from the neighbor is enabled.

Example:

```
Router# show bgp neighbors
```

Step 2 Run the **clear bgp** command to initiate the inbound soft reset.

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.1 soft in
```

- Use * instead of specific IP address to reset all neighbors.
- Use an AS number or *external* keyword for group resets.

The router requests the neighbor to resend routes, applying updated inbound policies.

Reset BGP neighbors using outbound soft reset

Resend all outbound BGP routes and policies to a neighbor.

Context: Use when outbound policy or attributes have changed.

Before you begin

None

Procedure

Step 1 Run the **show bgp neighbors** command to verify the route refresh capability from the neighbor is enabled.

Example:

```
Router# show bgp neighbors
```

Step 2 Run the **clear bgp** command to initiate the outbound soft reset.

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.1 soft out
```

- Use ***** instead of specific IP address to reset all neighbors.
- Use an AS number or *external* keyword for group resets.

The router resends all routes for the specified address family to the neighbor.

Reset BGP neighbors using hard reset

Completely resets the TCP connection and clears all routing information from a BGP neighbor.

Context: Use if you need to force a full session re-establishment or resolve persistent connectivity issues.

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.

Before you begin

None

Procedure

Run the **clear bgp** command with optional *graceful* and *soft* keywords.

Example:

```
Router# clear bgp ipv4 unicast 10.0.0.3 graceful soft out
```

- *graceful*: The *graceful* keyword preserves routes temporarily until the neighbor re-establishes.
- Use *** instead of specific IP address to reset all neighbors.
- Use an AS number or *external* keyword for group resets.

The TCP session is terminated and re-established, and routing tables are refreshed.

Clear BGP caches, tables, and databases

Remove all entries from specified BGP data structures and reset neighbor sessions.

Context: Use for troubleshooting or when data integrity is in question.

Perform this procedure to remove all contents of a particular cache, table, or database. This can become necessary when the contents of the particular structure have become, or are suspected to be, invalid.

Before you begin

Try [Reset BGP neighbors using hard reset](#) to resolve the issue.

Procedure

Run the **clear bgp** command to clear BGP caches and sessions.

Example:

Router# **clear bgp ipv4 172.20.1.1**

- **clear bgp ipv4** <ip address> : This command clears the caches, tables, and databases for a specific neighbor.
- **clear bgp external**: This command clears the caches, tables, and databases for all external peers.
- **clear bgp ***: This command clears the caches, tables, and databases for all BGP neighbors.

The selected sessions, caches, and databases are reset or cleared.

Disable a BGP neighbor

Administratively shut down a BGP neighbor session without removing its configuration.

Context: Use this procedure to temporarily disable a peer, for maintenance or troubleshooting.

Before you begin

None

Procedure

Step 1 Run the **configure** command to enter global configuration mode.

Example:

```
Router#configure
```

Step 2 Enter BGP configuration mode with your AS number and specify the neighbor.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 172.16.40.24
```

Step 3 Run the **shutdown** command to shut down the neighbor and save the configuration.

Example:

```
Router(config-bgp-nbr)# shutdown
Router(config-bgp-nbr)# commit
```

Disables all active sessions for the specified neighbor, leaving the session disabled and still in the configuration.

Neighbor capability suppressions

Neighbor capability suppression is a BGP feature that

- disables capability negotiation between BGP peers during the Open message exchange
- ensures interoperability with legacy devices that do not support the Capabilities option, and
- allows BGP sessions to be established even if the peer does not understand capabilities negotiation.

In BGP, capability negotiation is a mechanism that allows peers to exchange information about supported protocol extensions during session establishment. By default, BGP peers use this feature to agree on features they both support. However, some older customer premises equipment (CPE) devices do not recognize the Capabilities option, which can prevent successful BGP session setup. The neighbor capability suppression feature disables this negotiation in the Open message, enabling compatibility with such legacy devices.

Suppress BGP neighbor capabilities

Disable BGP capability negotiation during the Open message exchange for compatibility with legacy devices.

Context: Use when peering with devices that do not support BGP capabilities negotiation.

Before you begin

None

Procedure

Step 1 Run the **configure** command to enter global configuration mode.

Example:

```
Router#configure
```

Step 2 Enter BGP configuration mode with your AS number and specify the neighbor.

Example:

```
Router(config)# router bgp 100
Router(config-bgp)# neighbor 172.16.40.24
```

Step 3 Run the **capability suppress all** command to suppress all capabilities to the neighbor and save the configuration.

Example:

```
Router(config-bgp-nbr)# capability suppress all
Router(config-bgp-nbr)# commit
```

Capabilities negotiation is disabled for the specified neighbor.

Display BGP system and network statistics

View BGP routing tables, neighbor information, and performance statistics for troubleshooting and network management.

Context: Use these commands to monitor BGP operation and verify configuration status.

You can use any of these commands to display specific BGP information as needed:

Before you begin

None

Procedure

Step 1 Run the **show bgp cidr-only** command to display routes with non-natural network masks or Classless Inter-domain Routing (CIDR).

Example:

```
Router# show bgp cidr-only
```

Step 2 Run the **show bgp community** command to display routes that match a specific BGP community.

Example:

```
Router# show bgp community 1081:5 exact-match
```

Step 3 Run the **show bgp regexp regular-expression** command to display routes matching an AS path regular expression.

Example:

```
Router# show bgp regexp "^3 "
```

Step 4 Run the **show bgp** command to display all entries in the BGP routing table.

Example:

```
Router# show bgp
```

Step 5 Run the **show bgp neighbors ip-address [advertised-routes | dampened-routes | flap-statistics | performance-statistics | received prefix-filter | routes]** command to display neighbor information.

The following keywords display specific BGP neighbor information:

- *advertised-routes*: all routes that the router advertised to the neighbor
- *dampened-routes*: dampened routes learned from the neighbor
- *flap-statistics*: flap statistics for routes learned from the neighbor
- *performance-statistics*: performance statistics for the BGP process and this neighbor
- *received prefix-filter*: the received prefix list filter
- *routes*: routes learned from the neighbor

Example:

```
Router# show bgp neighbors 10.0.101.1
```

Step 6 Run the **show bgp paths** command to display all BGP paths in the database.

Example:

```
Router# show bgp paths
```

Step 7 Run the **show bgp neighbor-group group-name configuration** command to display the effective configuration for a neighbor group, including any configuration inherited by this neighbor group.

Example:

```
Router# show bgp neighbor-group group_1 configuration
```

Step 8 Run the **show bgp summary** command to display the summary status of all BGP connections.

Example:

```
Router# show bgp summary
```
