Core BGP Configurations

This chapter provides a comprehensive guide to configuring Border Gateway Protocol (BGP) on Cisco IOS XR devices. You will learn fundamental configuration methods, how to use templates for scalable deployments, and how inheritance and precedence rules affect BGP operations. The chapter also covers address family behavior and advanced features such as BGP confederations. By following this chapter, you will be ready to deploy and manage BGP in a variety of network scenarios.

Key aspects of BGP configurations

Cisco IOS XR software uses a neighbor-based model for BGP configuration. All BGP settings are managed at the individual neighbor level.

Key aspects of BGP configuration in Cisco IOS XR include:

- **Neighbor-specific configuration:** All settings for a particular neighbor must be defined under the neighbor configuration. You must apply configuration statements to each BGP neighbor individually.
- **Configuration and update groups:** IOS XR BGP does not support traditional peer groups. Instead, it uses configuration groups and update groups to simplify management and improve performance.
  - **Configuration groups** allow you to create templates with reusable configuration statements that can be applied to multiple neighbors. This approach reduces configuration errors and ensures consistency across multiple BGP neighbor configurations.
  - **Update groups** are automatically generated by the system based on common outbound policies among BGP neighbors. Update groups enable efficient sharing of update messages among neighbors that have the same outbound policies, improving scalability and network performance.

BGP configuration modes

BGP configurations are organized into multiple command modes, with each mode enabling you to configure specific aspects of BGP operation. You can enter each mode at the CLI prompt, and within that mode, use the **?** command for a list of available configuration commands.

These are common BGP configuration modes with examples demonstrating how to access each:

Mode	Description	Example
Global BGP configuration	Applies to overall BGP settings for the router	`router bgp 65000` sets BGP process for AS 65000
Address-family ipv4 unicast	Configures IPv4 unicast-specific routing parameters	`address-family ipv4 unicast` enables IPv4 unicast routes
Address-family ipv4 multicast	Configures IPv4 multicast-specific routing	`address-family ipv4 multicast` enables multicast routing
Address-family ipv6 unicast	Configures IPv6 unicast-specific routing parameters	`address-family ipv6 unicast` enables IPv6 unicast routes
Address-family vpnv4	Configures routing for VPNv4 addresses (L3VPN)	`address-family vpnv4` enables MPLS VPNv4 routing
Peer-group configuration	Sets attributes for a group of BGP neighbors	`neighbor PEERS peer-group` defines a peer group
Individual neighbor configuration	Customizes settings for a single neighbor	`neighbor 192.0.2.1 remote-as 65001`

Mode	Description	Typical Use Case
Global BGP configuration	Applies to overall BGP settings for the router	Set AS number, router ID, and global behaviors
Address-family ipv4 unicast	Configures IPv4 unicast-specific routing parameters	Control IPv4 unicast routing and policies
Address-family ipv4 multicast	Configures IPv4 multicast-specific routing	Manage IPv4 multicast routing in BGP
Address-family ipv6 unicast	Configures IPv6 unicast-specific routing parameters	Enable BGP for IPv6 unicast operations
Address-family vpnv4	Configures routing for VPNv4 addresses (L3VPN)	Implement MPLS VPNs
Peer-group configuration	Sets attributes for a group of BGP neighbors	Simplifies management of multiple neighbors
Individual neighbor configuration	Customizes settings for a single neighbor	Fine-tune parameters per neighbor

Router configuration mode

Use this mode to configure global BGP settings for an autonomous system.

```
Router# configuration
Router(config)# router bgp 140
Router(config-bgp)#
```

Router address family configuration mode

Use this mode to configure address family–specific BGP settings.

```
Router(config)# router bgp 112
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)#
```

Neighbor configuration mode

Use this mode to configure settings for a specific BGP neighbor.

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor 10.0.0.1
Router(config-bgp-nbr)#
```

VRF configuration mode

Use this mode to configure BGP for a specific Virtual Routing and Forwarding (VRF) instance.

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

VRF neighbor configuration mode

Use this mode to configure a BGP neighbor within a specific VRF.

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

VRF neighbor address family configuration mode

Use this mode to configure address family settings for a BGP neighbor in a VRF.

```
RP/0/RP0/CPU0:router(config)# router bgp 112
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_A
RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.1.2
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)#
```

VPNv6 address family configuration mode

Use this mode to configure VPNv6-specific BGP address family settings.

```
Router(config)# router bgp 150
Router(config-bgp)# address-family vpnv6 unicast
Router(config-bgp-af)#
```

L2VPN address family configuration mode

Use this mode to configure Layer 2 VPN-specific BGP address family settings.

```
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)#
```

BGP submodes for neighbor configuration

BGP neighbor configuration in Cisco IOS XR uses submodes to simplify and organize neighbor and address-family specific settings. Each submode determines the command scope and reduces repetitive command syntax for efficient configuration.

Neighbor submode

Neighbor submode allows you to configure attributes for a specific BGP neighbor without prefixing each command with the **neighbor** keyword.

You can enter neighbor submode by using the **neighbor** *ip-address* command in BGP configuration mode.

Benefit:

Streamlines configuration by targeting a single neighbor, reducing command repetition.

Example:

```
Router(config-bgp)# neighbor 192.23.1.2
Router(config-bgp-nbr)# remote-as 2002
Router(config-bgp-nbr)# address-family ipv4 unicast
```

Address-family submode

Address-family submode, accessed within neighbor submode, enables you to configure address-family-specific settings, such as IPv4 or IPv6 unicast, for a selected neighbor.

You can enter address-family submode for a specific neighbor by using the **address-family <afi> <safi>** command within neighbor submode.

Benefit:

Groups configuration by address family, minimizes command repetition, and clarifies parameter application.

Example:

```
Router(config-bgp)# neighbor 2002::2
Router(config-bgp-nbr)# remote-as 2023
Router(config-bgp-nbr)# address-family ipv6 unicast
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# route-policy one in
```

Note

- Commands entered in a submode automatically apply to the appropriate neighbor and address-family context.
- Leaving a submode returns you to the higher-level mode, and changes only apply within the current submode context.

BGP configuration templates

BGP configuration templates in Cisco IOS XR ensure consistent, scalable, and efficient BGP configuration, especially in large-scale deployments. Templates allow you to define common configurations once and apply them to multiple BGP neighbors, reducing configuration errors, and simplifying ongoing maintenance.

Types of BGP configuration templates

There are three primary types of BGP configuration templates, each designed for specific configuration requirements:

- **af-group:** This template type is used to define address family–specific settings for groups of neighbors. It allows you to configure commands that are dependent on the address family, such as IPv4 or IPv6. Neighbors can inherit these settings by referencing the af-group with the **use** command.
- **session-group:** The session-group template is intended for address family–independent (global) settings, which apply to groups of neighbors regardless of address family. Typical commands set at this level include global BGP options. Neighbors inherit these configuration commands from the session-group template by using the **use** command.
- **neighbor-group:** This template brings together both address family–independent and address family–dependent configurations. It enables you to apply a comprehensive set of commands, both global and specific to a group of neighbors. Neighbors assigned to a neighbor-group inherit all its configuration settings using the **use** command. Additionally, a neighbor-group can reference both af-group and session-group templates, allowing for hierarchical inheritance.

Inheritance behavior

- When you assign a neighbor to a group using the **use** command, the neighbor inherits all settings from the group unless you explicitly override a setting at the neighbor level.
- **neighbor-group** templates can include references to both session-groups and af-groups, allowing hierarchical inheritance.
- If a setting is configured directly under a neighbor, it overrides inherited values from any template.
- Some inherited configuration settings may be hidden if multiple group types are layered.

Key effects and attributes

- Commands entered at the session-group level define global, address family–independent options. These match the commands available in **neighbor** submode.
- Commands entered at the af-group level define address family–specific options, as found in the **neighbor-address-family** configuration submode.
- Commands at the neighbor-group level include both global and address family–dependent options. These commands may reference both af-group and session-group templates through the **use** command.

Configuration examples

af-group:

```
Router(config)# router bgp 140
Router(config-bgp)# af-group afmcast1 address-family ipv4 unicast
Router(config-bgp-afgrp)#
```

session-group:

```
Router# router bgp 140
Router(config-bgp)# session-group session1
Router(config-bgp-sngrp)#
```

neighbor-group:

```
Router(config)# router bgp 123
Router(config-bgp)# neighbor-group nbrgroup1
Router(config-bgp-nbrgrp)#
```

neighbor-group with address family:

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor-group nbrgroup1
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)#
```

Note Directly configured settings always take precedence over inherited template settings.

Precedence of BGP configuration template inheritance

When multiple configuration templates are applied to a BGP neighbor, the final settings are determined based on a specific order of precedence. Understanding this order is essential to troubleshooting and verifying BGP behavior.

Order of inheritance precedence

When multiple configuration sources apply to a BGP neighbor or group, the effective value is determined in the following order (from highest to lowest precedence):

1. **Direct neighbor configuration:** Values set directly on the neighbor override all inherited values.
2. **Session group or address family group:** If not set directly, values from a session group or an address family group override those from a neighbor group.
3. **Neighbor group:** If not set in the neighbor or higher-level groups, values from a neighbor group (including any values it inherits) are applied.
4. **System default:** If a value is not configured at any level, the Cisco IOS XR system default is used.

Inheritance rules summary

- Neighbors can inherit configuration from both session groups and neighbor groups, as well as address family groups.
- Neighbor groups can inherit from session groups, address family groups, and other neighbor groups.
- Session groups can inherit from other session groups.
- Address family groups can inherit from other address family groups.

Address family inheritance

- Inheritance rules apply to both address family-independent and address family-dependent configurations.
- If a property can be set in both an address family-dependent and independent context, the most specific value takes precedence, following the same order.

Examples

Example 1: Direct neighbor configuration takes highest precedence

If you set `advertisement-interval` both on a neighbor group and directly on a neighbor, the neighbor setting is applied.

```
Router(config)# router bgp 140
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# advertisement-interval 15
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 10.1.1.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use neighbor-group AS_1
Router(config-bgp-nbr)# advertisement-interval 20
```

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

```
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.
```

Example 2: Session-group or address family group overrides neighbor group

If a value is set both in a session-group (or address family group) and in a neighbor group, but not directly on the neighbor, the session group or address family group value is used.

```
Router(config)# router bgp 140
Router(config-bgp)# session-group AS_2
Router(config-bgp-sngrp)# advertisement-interval 15
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# advertisement-interval 20
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 192.168.0.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use session-group AS_2
Router(config-bgp-nbr)# use neighbor-group AS_1
```

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

```
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.
```

Example 3: Neighbor group inherited value is used when higher precedence is absent

If the neighbor uses only a neighbor group (not a session group or address family group), and no value is set directly on the neighbor, the neighbor group value is used, either directly or through its own inheritance.

```
Router(config)# router bgp 150
Router(config-bgp)# session-group AS_2
Router(config-bgp-sngrp)# advertisement-interval 20
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group AS_1
Router(config-bgp-nbrgrp)# advertisement-interval 15
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 192.168.1.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# use neighbor-group AS_1
```

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

```
 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.
```

Example 4: Default value is applied if not configured or inherited

If the `advertisement-interval` is not configured directly or inherited, the system default (30 seconds) applies.

Additional notes

- The same inheritance and precedence rules apply when groups inherit values from other groups.
- For any inheritable property, always refer to the effective configuration displayed by the **show bgp neighbors** command to verify which value is ultimately used.

Result

These rules ensure that BGP neighbor and group configurations in Cisco IOS XR software are inherited and applied in a predictable order, allowing flexible and scalable BGP policy management.

Viewing inherited configurations

After applying configuration templates, it's important to verify which settings are active and inherited, using dedicated show commands such as **show bgp neighbors configuration**, **show bgp af-group**, and others. These sections provide sample outputs and tips for interpreting results.

Viewing BGP neighbor inheritance

Use the **show bgp neighbors** commands to view inherited configuration details for BGP neighbors. This table summarizes primary keywords, and their functions:

Command options

You can specify one of these keywords to refine the command output:

Keyword	Description
configuration	Displays the effective configuration for a BGP neighbor, including any inherited settings from session groups, neighbor groups, or address family groups.
inheritance	Shows the session groups, neighbor groups, or address family groups from which a neighbor can inherit configuration properties.

Sample BGP neighbor configuration

```
Router(config)# router bgp 142
Router(config-bgp)# af-group GROUP_3 address-family ipv4 unicast
Router(config-bgp-afgrp)# next-hop-self
Router(config-bgp-afgrp)# route-policy POLICY_1 in
Router(config-bgp-afgrp)# exit
Router(config-bgp)# session-group GROUP_2
Router(config-bgp-sngrp)# advertisement-interval 15
Router(config-bgp-sngrp)# exit
Router(config-bgp)# neighbor-group GROUP_1
Router(config-bgp-nbrgrp)# use session-group GROUP_2
Router(config-bgp-nbrgrp)# ebgp-multihop 3
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)# weight 100
Router(config-bgp-nbrgrp-af)# send-community-ebgp
Router(config-bgp-nbrgrp-af)# exit
Router(config-bgp-nbrgrp)# exit
Router(config-bgp)# neighbor 192.168.0.1
Router(config-bgp-nbr)# remote-as 2
Router(config-bgp-nbr)# use neighbor-group GROUP_1
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# use af-group GROUP_3
Router(config-bgp-nbr-af)# weight 200
```

Command usage details

These commands allow you to view BGP neighbor configuration and inheritance details:

- **show bgp neighbors configuration**: Displays the active configuration for the specified neighbor, including inherited settings.
- **show bgp neighbors inheritance**: Shows which groups or templates provide inherited settings for the specified neighbor.

Viewing BGP address family group inheritance

Use the **show bgp af-group** command to view details about BGP address family groups, including details about configuration inheritance, the sources of inherited settings, and related entities that share configuration.

Command options

You can specify one of these keywords to refine the command output:

Keywords	Description
configuration	Displays the effective configuration for the address family group, including parameters inherited from other groups.
inheritance	Shows the address family groups from which the current group inherits configuration settings.
users	Lists the neighbors, neighbor groups, and address family groups that inherit configuration from the specified address family group.

Sample BGP address family groups configuration

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

Example command outputs

- **configuration keyword:**

```
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]
```

This example shows the source of each configuration item. The default-originate command was configured directly on this address family group, as indicated by the brackets [ ]. The remove-private-as command was inherited from address family group GROUP_2, which itself inherited the setting from address family group GROUP_3.

- **users keyword:**

```
Router# show bgp af-group GROUP_2 users

IPv4 Unicast: a:GROUP_1
```

- **inheritance keyword:**

```
Router# show bgp af-group GROUP_1 inheritance

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

This example shows that the address family group GROUP_1 inherits settings directly from GROUP_2, which in turn inherits from GROUP_3.

Viewing BGP session group inheritance

Use the show **bgp session-group** command with the **inheritance** keyword to view how BGP session groups inherit configuration.

Command options

You can specify one of these keywords to refine the command output:

Keywords	Description
configuration	Displays the effective configuration for a session group, including inherited settings.
inheritance	Shows the session groups from which a given session group inherits configuration.
users	Lists the session groups, neighbor groups, and neighbors that inherit configuration from a specified session group.

Sample BGP session group configuration

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

Example command outputs

- **configuration keyword:**

```
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]
```

- **users keyword:**

```
Router# show bgp session-group GROUP_3 users

Session: s:GROUP_1 s:GROUP_2
```

This example shows that both the GROUP_1 and GROUP_2 session groups inherit session parameters from the GROUP_3 session group.

- **inheritance keyword:**

```
Router# show bgp session-group GROUP_1 inheritance

Session: s:GROUP_2 s:GROUP_3
```

This example shows that the session group GROUP_1 inherits session parameters from the GROUP_3 and GROUP_2 session groups.

Viewing BGP neighbor group inheritance

Use the **show bgp neighbor-group** command to display detailed information about BGP neighbor group configurations, including inheritance and user relationships.

Command options

You can specify one of these keywords to refine the command output:

Keywords	Description
configuration	Displays the effective configuration for the neighbor group, including settings inherited from other groups.
inheritance	Displays the sources of configuration inheritance (address family groups, session groups, and neighbor groups) for the specified neighbor group.
users	Lists the neighbors and neighbor groups that inherit configuration from the specified neighbor group.

Sample BGP neighbor group configuration

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

```
Router(config-bgp-nbrgrp)# use neighbor-group GROUP_2
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Router(config-bgp-nbrgrp-af)# exit
Router(config-nbrgrp)# exit
Router(config-bgp)# neighbor-group GROUP_2
Router(config-bgp-nbrgrp)# use session-group GROUP_3
Router(config-bgp-nbrgrp)# address-family ipv4 unicast
Routerconfig-bgp-nbrgrp-af)# use af-group GROUP_2
Router(config-bgp-nbrgrp-af)# weight 100
```

Example command outputs

- **configuration keyword:**

```
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]
```

This example shows that 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

- **users keyword:**

```
Router# show bgp neighbor-group GROUP_2 users

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

This example shows that the neighbor group GROUP_1 inherits both session-level (address family-independent) and IPv4 unicast configuration parameters from the neighbor group GROUP_2.

- **inheritance keyword:**

```
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
```

This example shows that the neighbor group GROUP_1 inherits session-level (address family-independent) configuration parameters from neighbor group GROUP_2, which in turn inherits its session parameters from session group GROUP_3. Additionally, GROUP_1 inherits IPv4 unicast configuration parameters from GROUP_2. GROUP_2, in turn, inherits these IPv4 unicast parameters from the GROUP_2 address family group, which itself inherits them from the GROUP_3 address family group.

BGP address family configuration and behavior

BGP can support multiple address families, such as IPv4 and IPv6, each of which must be explicitly enabled and configured. Proper address family configuration ensures correct routing behavior for various protocol families.

No default address family in BGP

- You must explicitly configure each required address family at the BGP router level for it to be active.
- To activate a BGP session for a neighbor under a specific address family, you must configure that address family under both the router and the neighbor configuration.
- You can configure a neighbor without any address family, but cannot configure an address family under a neighbor unless that address family is present at the global BGP router level.

Key Point:

No address family is enabled by default in BGP. Every address family must be explicitly defined both at the global router level and under the neighbor if required.

Neighbor address family combinations

- **Default VRF**

  In the default VRF, you can configure both:

  - IPv4 unicast
  - IPv4 labeled-unicast

  Both address families are supported and can be enabled under the same neighbor.

- **Non-default VRF**

  In non-default VRFs, configuring both IPv4 unicast and IPv4 labeled-unicast address families under the same neighbor is not supported.

  The router allows the configuration but displays this error:

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

  If both address families are present in a BGP session for a neighbor in a non-default VRF, behavior becomes unpredictable, which can result in:

  - Incorrect advertisement of prefixes
  - Reachability issues

Tip To avoid reachability issues, explicitly configure a route policy to advertise prefixes only through either IPv4 unicast or IPv4 labeled-unicast address families, not both.

Default address family for show commands

Many BGP operational (show) commands allow specifying the Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI).

For protocol definitions, see RFC 1700 and RFC 2858.

Setting default AFI and SAFI

In Cisco IOS XR, you can define default AFI and SAFI values for parser commands, eliminating the need to specify them repeatedly.

Parser commands to set defaults:

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

Default values:

- AFI: **ipv4**
- SAFI: **unicast**

  To change these defaults, you must use the parser commands.

Note If you specify an AFI or SAFI in a show command, those supplied values override any parser defaults.

Checking current AFI or SAFI defaults

Use the **show default-afi-safi-vrf** command to view the current default AFI and SAFI settings.

Use case: Redistribute iBGP routes into IGP

In some advanced scenarios, you may need to redistribute iBGP-learned routes into an IGP, such as OSPF or IS-IS. This process should be approached with caution due to potential routing loops.

By default, iBGP routes are not advertised into IGPs. This procedure is required if you need iBGP routes dynamically included in your IGP's routing table.

Before you begin

- Ensure you are in privileged EXEC mode.
- Verify that BGP and your chosen IGP (OSPF or IS-IS) are already configured and operating.
- Understand the impact and risks of redistributing iBGP routes, particularly regarding potential routing loops.

Follow this step to redistribute iBGP routes into an IGP:

Procedure

Enter BGP router configuration mode, and enable redistribution of iBGP-learned routes into the IGP.

Example:

```
Router(config)# router bgp 120
Router(config-bgp)# bgp redistribute-internal
```

BGP Confederations

In very large networks, managing a full iBGP mesh can be challenging. BGP confederations address this by partitioning an AS into sub-ASs, improving scalability and simplifying management.

Characteristics of BGP routing domain confederations

BGP routing domain confederations are used to simplify iBGP mesh requirements and improve scalability in large autonomous systems. Key characteristics include:

- A confederation divides a single autonomous system (AS) into multiple sub-autonomous systems, called member ASes.
- The confederation appears as a single autonomous system to external BGP peers.
- Each member AS maintains a full internal BGP (iBGP) mesh within itself. It establishes only limited external BGP (eBGP-like) connections to other member ASes within the same confederation.
- Peers in different member ASes use eBGP sessions but exchange routing information as if they were iBGP peers within the confederation.
- Routing attributes such as next hop, multi-exit discriminator (MED), and local preference are preserved across confederation boundaries.
- A single Interior Gateway Protocol (IGP) can be used for the entire confederation.

This approach reduces the number of required iBGP sessions and simplifies network management while maintaining compatibility with external BGP peers.

Configure a BGP routing domain confederation

Reduce internal iBGP mesh complexity by grouping multiple autonomous systems (AS) into a BGP confederation, enabling easier management and scalability.

A BGP routing domain confederation divides a single autonomous system (AS) into multiple sub-autonomous systems (sub-ASes). This approach simplifies iBGP connectivity by fully meshing only within each sub-AS, while maintaining a consistent external BGP appearance. All sub-ASes in a confederation use the same IGP.

Before you begin

- Gather the confederation AS number (the confederation identifier).
- Collect the list of sub-AS numbers to include as confederation peers.
- Access the router in global configuration mode.

Procedure

Specify the confederation identifier, and associate all AS numbers that will participate in the confederation.

Example:

```
Router# configure
            Router(config)# router bgp 120
            Router(config-bgp)# bgp confederation identifier 5
            Router(config-bgp)# bgp confederation peers 1091
            Router(config-bgp)# bgp confederation peers 1092
            Router(config-bgp)# bgp confederation peers 1093
            Router(config-bgp)# bgp confederation peers 1094
```

The router now treats the configured sub-ASes as part of the same BGP confederation. To external peers, the group of sub-ASes appears as a single AS.

Example

This is a sample configuration showing several peers within a BGP confederation. The confederation is made up of three internal autonomous systems with AS numbers 6001, 6002, and 6003. To BGP speakers outside the confederation, it appears as a single autonomous system with AS number 666, as specified by the bgp confederation identifier command.

```
router bgp 6001
 bgp confederation identifier 666
 bgp confederation peers
  6002
  6003
   exit
 address-family ipv4 unicast
  neighbor 172.16.232.55
  remote-as 6002
   exit
 address-family ipv4 unicast
  neighbor 172.16.232.56
  remote-as 6003
   exit
 address-family ipv4 unicast
  neighbor 172.19.69.1
  remote-as 777
```

On a BGP router in autonomous system 6001, the bgp confederation peers command designates peers from autonomous systems 6002 and 6003 as special eBGP peers within the confederation. As

a result, peers at 172.16.232.55 and 172.16.232.56 receive updates with local preference, next hop, and MED attributes unchanged. In contrast, the router at 172.19.69.1 is a standard eBGP peer outside the confederation, so the updates it receives from this peer are treated as regular eBGP updates from autonomous system 666.

BGP confederation peerings

A BGP confederation peering is a routing relationship within a BGP confederation that:

- enables routers in different sub-autonomous systems (sub-AS) of the same confederation to exchange routing information
- allows specific route advertisements using iBGP while circumventing standard iBGP full mesh requirements, and
- provides mechanisms to override the split horizon rule and control route learning between peer routers.

Autonomous system (AS):

An autonomous system is a collection of routers under a single administrative domain that use Interior Gateway Protocols (IGPs) for internal routing and Exterior Gateway Protocols (EGPs), such as BGP, for inter-domain communication.

Sub-autonomous system (sub-AS):

A sub-autonomous system is a distinct subset within a larger autonomous system, with its own administrative control and routing policies. Sub-ASs are used within BGP confederations to simplify iBGP mesh requirements.

Confederation:

A BGP confederation divides a single large AS into multiple sub-ASs, reducing the iBGP mesh. Externally, the confederation appears as a single AS, while internally it consists of multiple sub-ASs. Routers within different sub-ASs peer using eBGP, but exchange routing information similar to iBGP sessions, preserving BGP path attributes.

Autonomous System Number (ASN):

The ASN uniquely identifies each autonomous system or sub-autonomous system in BGP routing.

Split horizon:

Split horizon is a BGP routing rule that prevents a router from advertising a route back into the sub-AS (or confederation) from which it was learned, thereby preventing routing loops.

Table 1: Feature History Table

Feature Name	Release Name	Description
Peering Between BGP Routers Within the Same Confederation	Release 7.11.1	You can now enable BGP peering between routers in the sub-autonomous system (AS) within a confederation to advertise specific router updates using iBGP. This capability ensures that the mesh of routers between sub-ASes in a confederation maintains consistent routing tables, ensuring proper network reachability. Enabling this feature helps improve preventing performance reduction and traffic management challenges. The feature introduces these changes: **CLI:** • `allowconfedas-in` **YANG Data Models** • New XPaths for `Cisco-IOS-XR-ipv4-bgp-cfg.yang` • `Cisco-IOS-XR-um-router-bgp-cfg` (see GitHub, YANG Data Models Navigator)

Challenge of full iBGP mesh in large networks

In large-scale networks, maintaining a full iBGP mesh within an autonomous system becomes impractical as the number of routers increases.

Role of BGP confederations

BGP confederations allow a single AS to be partitioned into multiple sub-ASs, each of which maintains a full iBGP mesh internally. Routers in different sub-ASs establish eBGP-like sessions (called confederation peerings), but inside the confederation, the attributes of iBGP are preserved.

Impact of the split horizon rule

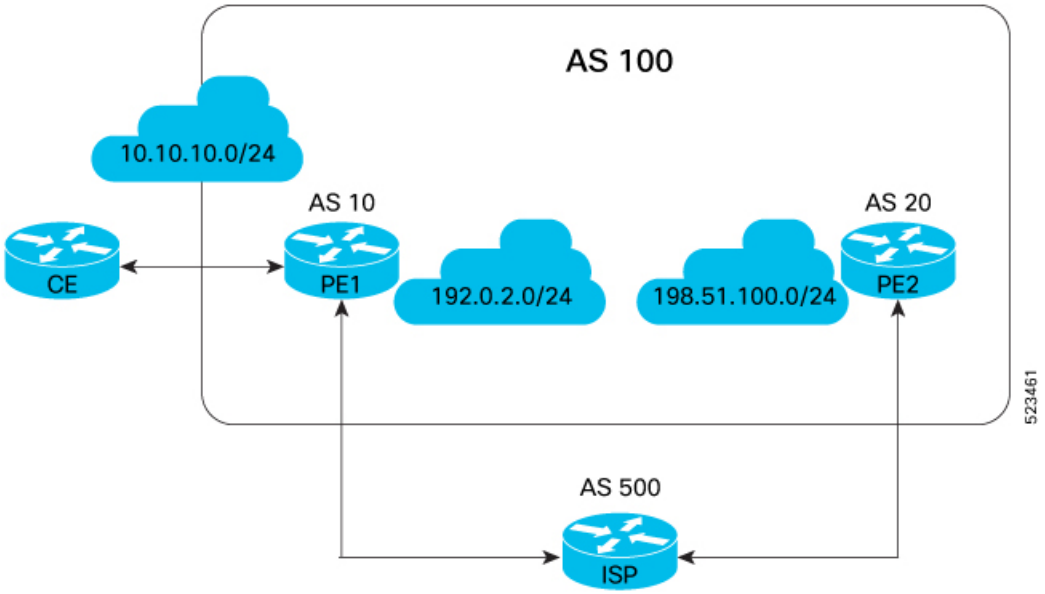
By default, the split horizon rule prevents routers in the same confederation from learning routes from one another if the route originated within the confederation.

Breaking split horizon

In scenarios requiring increased route flexibility or network customization, it may be necessary to break the split horizon rule. The **allowconfedas-in** command enables routers to bypass this restriction, allowing selected routes to be learned by peer routers within the confederation, and providing granular control over the number of times a route may be re-accepted within the confederation.

Example

Figure 1: BGP confederation peerings



In this sample topology, Router PE1 and Router PE2 are both part of a BGP confederation but belong to different sub-ASs (e.g., PE1 in sub-AS 100, PE2 in sub-AS 20). The CE router advertises the route 10.10.10.0/24 to PE1, which then advertises it to the ISP router (AS 500). The ISP router then passes the route to PE2. PE2 sees the confederation's AS numbers in the AS_PATH and, by default, drops the route due to the split horizon rule. To permit route learning in this scenario, configure the `allowconfedas-in` command on both PE1 and PE2. This allows PE2 to accept the 10.10.10.0/24 prefix from PE1, even though both routers are in the same confederation.

Limitations of configuring BGP confederation peerings

Configure allowconfedas-in only within specified limits

When you configure BGP confederation peering using the **allowconfedas-in** command, observe these limitations:

- Peer routers within a confederation can exchange information with each other only a limited number of times when the **allowconfedas-in** command is configured.
- The number of times information can be exchanged is limited to a range of 1 to 10.
- By default, the maximum number of exchanges is set to 3.

Configure BGP peering within a confederation

Enable peering between BGP routers that belong to the same BGP confederation, allowing them to exchange routing information.

Use this task when you want routers with different BGP autonomous system (AS) numbers, but within the same confederation, to peer and share routes.

Before you begin

- Ensure you are in privileged EXEC mode on each router.
- Confirm that you have the correct confederation AS numbers and that the routers are configured to use BGP.
- Identify the IP addresses of the routers you want to peer.

Follow these steps to configure BGP peering within a confederation:

Procedure

Step 1 Configure peer routers in the same confederation to learn from each other for a specified number of times.

Example:

```
Router# router bgp 65001
Router(config-bgp)# bgp confederation peers 65002
Router(config-bgp)# bgp confederation identifier 100
Router(config-bgp)# neighbor 198.51.100.3
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# allowconfedas-in 1
```

Step 2 Use the **show bgp neighbor** command to verify route learning among confederation peers.

Example:

```
show bgp neighbor 198.51.100.3 | in allow
Fri Mar  7 15:38:13.092 +0530
  Inbound soft reconfiguration allowed (override route-refresh)
  My confederation AS number is allowed 3 times in received updates.
```

This output shows that the peers within the same confederation have learned from each other's routes, and the learning among these peers has occurred three times.

The routers within the same BGP confederation are now configured as peers and can exchange routes, allowing repeated AS numbers in the received updates as specified.