Understanding BGP

BGP stands for Border Gateway Protocol.

Here are some key characteristics of BGP:

- BGP is an Exterior Gateway Protocol (EGP).
- Its purpose is to establish an interdomain routing system.
- It ensures the loop-free exchange of routing information between autonomous systems (AS).
- AS is a collection of routers under a single administration, typically running Interior Gateway Protocols (IGPs) like RIP or OSPF internally and it correctly cites the defining RFCs (1163, 1267, and 1771).
- Path-Vector Protocol: Unlike interior gateway protocols (IGPs) like OSPF or EIGRP that use metrics like bandwidth or delay, BGP makes routing decisions based on paths, attributes, and policies. It carries a list of AS numbers that a route has traversed.
- Inter-Domain Routing: BGP's primary role is to route traffic between different autonomous systems.
- Policy-Based Routing: BGP allows network administrators to implement complex routing policies, such as traffic engineering, peering agreements, and security filtering.
- Reliable Transport: BGP uses TCP (port 179) as its transport protocol, ensuring reliable delivery of routing updates

Benefits BGP

The benefits of BGP (Border Gateway Protocol) include:

- Scalability for Inter-AS Routing: BGP enables routing between different autonomous systems (ASes), supporting large-scale internet routing by exchanging reachability information across AS boundaries.
- Loop Prevention: BGP uses the AS path attribute to prevent routing loops by tracking the sequence of ASes a route has traversed.
- Policy Control: BGP allows fine-grained control over routing decisions and path selection through various attributes, enabling enforcement of routing policies at the AS level.
- Support for Both IBGP and EBGP: BGP supports internal routing within an AS (IBGP) and external routing between ASes (EBGP), providing flexibility in network design.
- Route Aggregation and CIDR Support: BGP Version 4 supports Classless Inter-Domain Routing (CIDR), allowing route aggregation to reduce routing table size and improve efficiency.
- Reliable Transport: BGP uses TCP (port 179) for reliable session establishment and maintenance between peers.
- Incremental Updates: After the initial full routing table exchange, BGP sends only incremental updates, reducing bandwidth and processing overhead.
- Nonstop Forwarding Awareness: When combined with features like Graceful Restart, BGP can maintain forwarding during route processor failover or software upgrades, enhancing network stability.

Restrictions BGP

- **BGP Graceful Restart and xFSU**: When configuring **bgp graceful-restart** on xFSU-capable devices, the BGP hold time must exceed the device-specific minimum graceful restart time. If the hold time is lower or an unsupported hold timer is received from a peer, BGP will not be supported during an xFSU event, even if the value is accepted.
- **Layer 3 Forwarding Delay**: Upon device startup or after executing the **clear ip bgp** command, Layer 3 forwarding will be delayed for approximately 80 seconds until routing tables are fully populated. You can verify the routing table population status using the **show ip bgp ip-address** command.
- The routing tables require around 80 seconds for population. You can use the show ip bgp ip-address command, in privileged EXEC mode, to check whether the routing tables are populated or not.
-

•

Network Topology

Routers within the same autonomous system (AS) that exchange BGP updates run internal BGP (IBGP). Routers in different autonomous systems that exchange BGP updates run external BGP (EBGP). Most configuration commands are the same for EBGP and IBGP, but EBGP exchanges updates between autonomous systems, while IBGP exchanges updates within an AS. Figure 1 illustrates a network running both EBGP and IBGP.

<figure to be inserted>

Before exchanging information with an external AS, BGP ensures that networks within the AS can be reached by defining internal BGP peering among routers within the AS and by redistributing BGP routing information to IGPs that run within the AS, such as IGRP and OSPF.

Routers running a BGP routing process are called BGP speakers. BGP uses Transmission Control Protocol (TCP) on port 179 as its transport protocol. Two BGP speakers with a TCP connection for exchanging routing information are known as peers or neighbors. In Figure 1, Routers A and B are BGP peers. Routers B and C, and Routers C and D, are also peers. he routing information consists of a series of AS numbers that specify the full path to the destination network. BGP uses this information to build a loop-free map of autonomous systems.

The network has these characteristics:

- Routers A and B run EBGP, while Routers B and C run IBGP. EBGP peers are directly connected, but IBGP peers do not need to be. If an IGP is running that allows the neighbors to reach each other, IBGP peers can be indirectly connected.
- All BGP speakers within an AS must establish peer relationships with each other, forming a full logical mesh. BGP4 provides two techniques to reduce this requirement: confederations and route reflectors.
- AS 200 is a transit AS for AS 100 and AS 300; it transfers packets between AS 100 and AS 300.

BGP peers initially exchange their full routing tables, then send only incremental updates. They also exchange keepalive messages to ensure the connection is up and notification messages if there are errors or special conditions.

In BGP, each route consists of a network number, an autonomous system path (the list of autonomous systems the information has passed through), and other path attributes. The main function of BGP is to exchange network reachability information, including AS paths, with other BGP systems. This information helps determine AS connectivity, prunes routing loops, and enforces AS-level policy decisions.

A router or device running Cisco IOS will not select or use an IBGP route unless it has a route to the next-hop router and has received IGP synchronization, unless synchronization is disabled. When multiple routes are available, BGP bases its path selection on attribute values. SFor details about BGP attributes, refer to the “Configuring BGP Decision Attributes” section.

BGP Version 4 supports classless interdomain routing (CIDR), which reduces the size of routing tables by allowing the creation of aggregate routes and supernets. CIDR removes the concept of network classes in BGP and supports advertising IP prefixes.

BGP Components

- BGP Speaker (Router): A device running the BGP routing process that exchanges routing information with other BGP speakers.
- BGP Peers (Neighbors): Two BGP speakers that establish a TCP connection (port 179) to exchange routing updates. Peers can be:
  1. External BGP (EBGP): Peers in different autonomous systems (AS).
  2. Internal BGP (IBGP): Peers within the same AS.
- Autonomous System (AS): A collection of IP networks under a single administrative domain. BGP manages routing between and within ASes.
- BGP Routing Table: Stores the routes learned from BGP peers, including network prefixes and path attributes.
- Path Attributes: Information associated with each route that influences path selection, such as AS path, next-hop, local preference, and others.
- AS Path: A list of AS numbers that a route has traversed, used to prevent routing loops and assist in path selection.
- TCP Connection: BGP uses TCP (port 179) as its transport protocol to ensure reliable delivery of routing information.
- Route Reflectors and Confederations: Mechanisms to reduce the full mesh IBGP requirement by logically grouping BGP speakers.
- Next-Hop Reachability and Synchronization: Ensures that a router only uses IBGP routes if it can reach the next-hop and synchronization with the IGP is achieved (unless disabled).

Key Concepts

- BGP Peers (Neighbors)
- Autonomous Systems (AS)
- Full Mesh Requirement
- AS Path Attribute
- Route Attributes
- Route Exchange
- Next-Hop Reachability and Synchronization
- Support for Classless Inter-Domain Routing (CIDR) and Route Aggregation
- Nonstop Forwarding Awareness
- Policy Control

BGP Peers (Neighbours)

BGP peers, also known as neighbors, are routers that have established a TCP connection (on port 179) to exchange BGP routing information. These peers can be classified into two types:

- External BGP (eBGP) Peers: Routers that belong to different autonomous systems (AS) and exchange routing information across AS boundaries. Typically, eBGP peers are directly connected.
- Internal BGP (iBGP) Peers: Routers within the same AS that exchange BGP updates. iBGP peers do not need to be directly connected as long as an Interior Gateway Protocol (IGP) provides reachability between them.

Key characteristics of BGP peers include:

- Full Mesh Requirement: All iBGP peers within an AS must be logically fully meshed, meaning each iBGP router must establish a peering session with every other iBGP router. This requirement can be relaxed using route reflectors or confederations.
- BGP Speakers: Routers running BGP are called BGP speakers. They exchange full routing tables initially and then send incremental updates.
- Route Exchange: Peers exchange network reachability information, including AS path attributes, to build a loop-free interdomain routing topology.
- Peer Relationships: Peers maintain the connection by exchanging keepalive messages and use notification messages to signal errors or special conditions.
- Policy Application: BGP peer policies control route advertisement and acceptance, applied individually or via peer groups to simplify management.

Autonomous Systems

An Autonomous System (AS) is a collection of IP networks and routers under the control of a single organization that presents a common routing policy to the Internet. Each AS is assigned a unique AS number (ASN) which is used to identify it globally.

Key points about Autonomous Systems in BGP:

- Purpose: ASes enable the organization of the Internet into manageable routing domains. They allow routing policies to be applied consistently within the AS and control how routing information is exchanged with other ASes.
- Types of BGP Peering:
  - External BGP (eBGP): Peering between routers in different ASes. eBGP peers exchange routing information across AS boundaries.
  - Internal BGP (iBGP): Peering between routers within the same AS. iBGP peers exchange routing information internally to ensure consistent routing within the AS.
- AS Path Attribute: BGP uses the AS path attribute to record the sequence of ASes that routing information has traversed. This helps in loop prevention and policy enforcement.
- Transit AS: An AS that carries traffic between other ASes without being the source or destination. For example, AS 200 can be a transit AS between AS 100 and AS 300.

- Routing Policy: Each AS can implement its own routing policies, controlling how routes are advertised and accepted from other ASes.

Full Mesh Requirement in BGP

The Full Mesh Requirement in BGP states that all BGP speakers within the same Autonomous System (AS) must establish a peer relationship with each other. This means every iBGP router must be directly connected logically to every other iBGP router in the AS to exchange routing information.

Key points about the Full Mesh Requirement:

- Purpose: Ensures that routing information learned from an external BGP (eBGP) peer is propagated to all other routers within the AS.
- Logical Full Mesh: Even if routers are not physically directly connected, they must have a logical peer connection.
- Scalability Challenge: As the number of routers increases, the number of peer connections grows exponentially, making full mesh difficult to manage.
- Solutions to Reduce Full Mesh:
  - Route Reflectors: Allow some routers to reflect routes to others, reducing the need for a full mesh.
  - Confederations: Divide a large AS into smaller sub-ASes to reduce the number of iBGP peers.

AS Path Attribute in BGP

The AS Path Attribute is a fundamental path attribute in BGP that lists the sequence of Autonomous Systems (ASes) a route has traversed to reach a destination network. It is used primarily for:

- Loop Prevention: BGP uses the AS path to detect and prevent routing loops by checking if its own AS number appears in the AS path of a received route. If it does, the route is discarded.
- Path Selection: The AS path length (number of AS hops) is a key factor in BGP's route selection process. Routes with shorter AS paths are generally preferred.
- Policy Enforcement: Network administrators can apply routing policies based on AS path information to control route advertisement and acceptance.

The AS path attribute is carried in every BGP update message and is updated by each AS that propagates the route. When a BGP speaker advertises a route to an external peer, it prepends its own AS number to the AS path.

BGP Route Attributes

BGP uses several route attributes to determine the best path to a destination and to enforce routing policies. These attributes are carried in BGP update messages and influence route selection and advertisement. The main BGP route attributes include:

- Well-Known Mandatory Attributes (must be recognized and included in all BGP updates)

- AS Path: Lists the sequence of Autonomous Systems a route has traversed; used for loop prevention and path selection.
- Next Hop: Specifies the IP address of the next-hop router to reach the destination.
- Origin: Indicates the origin of the route (IGP, EGP, or incomplete).
- Local Preference: Used within an AS to prefer an exit point for outbound traffic.
- Atomic Aggregate: Indicates that the route is an aggregate of multiple routes.
- Aggregator: Identifies the router that performed route aggregation

- Well-Known Discretionary Attributes (recognized by all BGP implementations but optional in updates)
  - Local Preference (also considered discretionary in some contexts).
- Optional Transitive Attributes (may be passed along to other BGP peers even if not recognized)
  - Community: Used to group routes and apply routing policies
  - Extended Community: Provides additional tagging capabilities.
- Optional Non-Transitive Attributes (used only between directly connected BGP peers)

Multi-Exit Discriminator (MED): Suggests preferred entry points into an AS from neighboring ASes.

BGP Route Exchange

Route Exchange in BGP refers to the process by which BGP peers (neighbors) share routing information to build and maintain a consistent and loop-free interdomain routing table. The key points about BGP route exchange are:

Summary

- Initial Full Table Exchange: When two BGP peers establish a session, they first exchange their entire BGP routing tables. This ensures that both peers have a complete view of the reachable networks known to each other.
- Incremental Updates: After the initial exchange, BGP peers send only incremental updates. These updates include new routes, withdrawn routes, or changes to existing routes, minimizing the amount of data transmitted and improving efficiency.
- TCP Transport: BGP uses TCP port 179 to establish a reliable connection between peers, ensuring ordered and error-checked delivery of routing updates.
- Keepalive Messages: To maintain the session, BGP peers periodically send keepalive messages. These confirm that the connection is still active and prevent session timeout.
- Notification Messages: If errors or special conditions occur, BGP peers send notification messages to inform each other and potentially terminate the session if necessary.
- Next-Hop Reachability: For a BGP route to be used, the router must have a route to the next-hop IP address, typically learned via an IGP or static route, ensuring proper forwarding.

- Synchronization with IGP: Unless synchronization is disabled, BGP requires that routes learned via IBGP are also present in the IGP before being used, to prevent routing loops.

This route exchange mechanism allows BGP to efficiently propagate routing information across autonomous systems while supporting policy control and loop prevention.

Workflow

1. •

When...	And...	Then...	And...

Result

What's next

Next-Hop Reachability and Synchronization in BGP

- Next-Hop Reachability: For a BGP route to be considered valid and used for forwarding, the router must have a route to the next-hop IP address specified in the BGP update. This next-hop is typically learned through an Interior Gateway Protocol (IGP) or a static route. Without reachability to the next-hop, the BGP route will not be installed in the routing table, ensuring proper packet forwarding and avoiding blackholing of traffic.
- Synchronization: This is a mechanism to prevent routing loops between BGP and the IGP. When synchronization is enabled, a BGP router will not advertise a route learned via Internal BGP (IBGP) to external peers unless that route is also present in the IGP. This ensures that all routers within the AS have consistent routing information before routes are advertised outside the AS. However, in many modern networks, synchronization is disabled because the IGP and BGP are managed separately, and route reflectors or confederations are used to scale IBGP.

Together, these mechanisms ensure that BGP routes are valid and consistent within and across autonomous systems, maintaining stable and loop-free routing environments.

Support for CIDR and Route Aggregation in BGP

- CIDR (Classless Inter-Domain Routing) Support: BGP fully supports CIDR, allowing the advertisement of IP prefixes of variable length rather than being restricted to classful boundaries. This capability enables more efficient use of IP address space and reduces the size of routing tables by allowing aggregation of routes.
- Route Aggregation: BGP supports route aggregation (also known as route summarization), which allows multiple contiguous IP prefixes to be combined into a single, summarized route advertisement. This reduces the number of routes exchanged between BGP peers, improving routing efficiency and scalability.
- How It Works:
  - Aggregation is configured on the BGP router to advertise a summarized route that represents multiple more specific routes.

- The aggregated route can include attributes such as the AS path and next-hop, which are used to maintain routing consistency.
- BGP also supports the use of the aggregate-address command to configure route aggregation and control attributes like the summary-only option to suppress more specific routes.

- Benefits
  - Reduces routing table size.
  - Limits the propagation of detailed routing information.
  - Enhances network stability and performance.

c-Nonstop Forwarding Awareness

-
-
-

BGP Policy Control

Policy Control in BGP is a critical mechanism that allows network administrators to influence and control the routing decisions and the advertisement of routes between BGP peers. It provides flexibility to enforce routing policies based on various criteria.

Key aspects of BGP Policy Control include:

- Route Filtering: Control which routes are accepted from or advertised to BGP neighbors using prefix lists, route maps, or access control lists (ACLs). This helps in restricting unwanted routes and controlling routing updates.
- Route Maps: These are powerful tools used to match and modify route attributes such as AS path, next-hop, metric, local preference, and community values. Route maps enable conditional policy application.
- AS Path Filtering and Manipulation: Policies can filter routes based on AS path patterns or modify the AS path attribute to influence route selection and prevent routing loops.
- Local Preference: This attribute is used within an AS to prefer certain routes over others. Policy control can set local preference values to influence outbound traffic paths.
- Community Attributes: BGP communities are tags attached to routes that can be used to apply routing policies across multiple routers consistently.
- Next-Hop Control: Policies can modify the next-hop attribute to ensure reachability or influence routing decisions.
- Prefix Lists and Route Maps Combination: Often used together to create granular policies for route advertisement and acceptance.
- Policy Application Points: Policies can be applied inbound (on routes received from peers) or outbound (on routes advertised to peers).

These policy control mechanisms enable fine-grained management of BGP routing behavior, ensuring optimal routing, security, and compliance with organizational requirements.

How to configure BGP

This section provides configurational information about BGP.

Default Setting	
Aggregate address	Disabled: None defined.
AS path access list	None defined.
Auto summary	Disabled.
Best path	The router considers as-path in choosing a route and does not compare similar routes from external BGP peers. Compare router ID: Disabled.
BGP community list	Number: None defined. When you permit a value for the community number, the list defaults to an implicit deny for everything else that has not been permitted. Format: Cisco default format (32-bit number).
BGP confederation identifier/peers	Identifier: None configured. Peers: None identified.
BGP Fast external fallover	Enabled.
BGP local preference	100. The range is 0 to 4294967295 with the higher value preferred.

Default Setting	
BGP network	None specified; no backdoor route advertised.
BGP route dampening	Disabled by default. When enabled: Half-life is 15 minutes. Re-use is 750 (10-second increments). Suppress is 2000 (10-second increments). Max-suppress-time is 4 times half-life; 60 minutes.
BGP router ID	The IP address of a loopback interface if one is configured or the highest IP address configured for a physical interface on the router.
Default information originate (protocol or network redistribution)	Disabled.

Default Setting	
Default metric	Built-in, automatic metric translations.
Distance	External route administrative distance: 20 (acceptable values are from 1 to 255). Internal route administrative distance: 200 (acceptable values are from 1 to 255). Local route administrative distance: 200 (acceptable values are from 1 to 255).
Distribute list	In (filter networks received in updates): Disabled. Out (suppress networks from being advertised in updates): Disabled.
Internal route redistribution	Disabled.
IP prefix list	None defined.
Multi exit discriminator (MED)	Always compare: Disabled. Does not compare MEDs for paths from neighbors in different autonomous systems. Best path compare: Disabled. MED missing as worst path: Disabled.

Default Setting	
	Deterministic MED comparison is disabled.
Neighbor	Advertisement interval: 30 seconds for external peers; 5 seconds for internal peers.
	Change logging: Enabled.
	Conditional advertisement: Disabled.
	Default originate: No default route is sent to the neighbor.
	Description: None.
	Distribute list: None defined.
	External BGP multihop: Only directly connected neighbors are allowed.
	Filter list: None used.
	Maximum number of prefixes received: No limit.
	Next hop (router as next hop for BGP neighbor): Disabled.

Default Setting	
	Password: Disabled.
	Peer group: None defined; no members assigned.
	Prefix list: None specified.
	Remote AS (add entry to neighbor BGP table): No peers defined.
	Private AS number removal: Disabled.
	Route maps: None applied to a peer.
	Send community attributes: None sent to neighbors.
	Shutdown or soft reconfiguration: Not enabled.
	Timers: keepalive: 60 seconds; holdtime: 180 seconds.
	Update source: Best local address.
	Version: BGP Version 4.
	Weight: Routes learned through BGP peer: 0; routes sourced by the local router: 32768.
Default Setting	
NSF[1] Awareness	Disabled[2]. If enabled, allows Layer 3 switches to continue forwarding packets from a neighboring NSFcapable router during hardware or software changes.
Route reflector	None configured.
Synchronization (BGP and IGP)	Disabled.
Table map update	Disabled.
Timers	Keepalive: 60 seconds; holdtime: 180 seconds.

Default Setting	
NSF[1] Awareness	Disabled[2]. If enabled, allows Layer 3 switches to continue forwarding packets from a neighboring NSFcapable router during hardware or software changes.
Route reflector	None configured.

Default Setting	
Synchronization (BGP and IGP)	Disabled.
Table map update	Disabled.
Timers	Keepalive: 60 seconds; holdtime: 180 seconds.

[1] Nonstop Forwarding

[2] NSF Awareness can be enabled for IPv4 on switches with the Network Advantage license by enabling Graceful Restart.

Procedure

Use the task context details in this topic.

Configuring a BGP routing process

Perform this task to configure a BGP routing process. You must perform the required steps at least once to enable BGP. The optional steps here allow you to configure additional features in your BGP network. Several of the features, such as logging neighbor resets and immediate reset of a peer when its link goes down, are enabled by default but are presented here to enhance your understanding of how your BGP network operates.

Note A device that runs Cisco software can be configured to run only one BGP routing process and to be a member of only one BGP autonomous system. However, a BGP routing process and autonomous system can support multiple concurrent BGP address family and subaddress family configurations.

The configuration in this task is done at Router A in the figure below and would need to be repeated with appropriate changes to the IP addresses (for example, at Router B) to fully achieve a BGP process between the two devices. No address family is configured here for the BGP routing process, so routing information for the IPv4 unicast address family is advertised by default.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp autonomous-system-number**

Example:

```
Device(config)# router bgp 40000
```

Configures a BGP routing process, and enters router configuration mode for the specified routing process.

Use the autonomous-system-number argument to specify an integer, from 0 and 65534, that identifies the device to other BGP speakers.

Step 4 **network network-number [mask network-mask] [route-map route-map-name]**

Example:

```
Device(config-router)# network 10.1.1.0 mask 255.255.255.0
```

(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table.

For exterior protocols, the network command controls which networks are advertised. Interior protocols use the network command to determine where to send updates.

Step 5 **bgp router-id ip-address**

Example:

```
Device(config-router)# bgp router-id 10.1.1.99
```

(Optional) Configures a fixed 32-bit router ID as the identifier of the local device running BGP.

Use the ip-address argument to specify a unique router ID within the network.

Note
Configuring a router ID using the bgp router-id command resets all active BGP peering sessions.

Step 6 **timers bgp keepalive holdtime**

Example:

```
Device(config-router)# timers bgp 70 120
```

(Optional) Sets BGP network timers.

Use the keepalive argument to specify the frequency, in seconds, with which the software sends keepalive messages to its BGP peer. By default, the keepalive timer is set to 60 seconds.

Use the holdtime argument to specify the interval, in seconds, after which the software, having not received a keepalive message, declares a BGP peer dead. By default, the holdtime timer is set to 180 seconds.

Step 7 **bgp fast-external-fallover**

Example:

```
Device(config-router)# bgp fast-external-fallover
```

(Optional) Enables the automatic resetting of BGP sessions.

By default, the BGP sessions of any directly adjacent external peers are reset if the link used to reach them goes down.

Step 8 **bgp log-neighbor-changes**

Example:

```
Device(config-router)# bgp log-neighbor-changes
```

(Optional) Enables logging of BGP neighbor status changes (up or down) and neighbor resets.

Use this command for troubleshooting network connectivity problems and measuring network stability. Unexpected neighbor resets might indicate high error rates or high packet loss in the network and should be investigated.

Step 9 **end**

Example:

```
Device(config-router)# end
```

Exits router configuration mode and enters privileged EXEC mode.

Step 10 **show ip bgp [network] [network-mask]**

Example:

```
Device# show ip bgp
```

(Optional) Displays the entries in the BGP routing table.

Note
Only the syntax applicable to this task is used in this example. For more details, see the Cisco IOS IP Routing: BGP Command Reference.

Managing Routing Policy Changes

To learn if a BGP peer supports the route refresh capability and to reset the BGP session:

Procedure

Step 1 **show ip bgp neighbors**

Example:

```
Device# show ip bgp neighbors
```

Displays whether a neighbor supports the route refresh capability. When supported, this message appears for the router: Received route refresh capability from peer.

Step 2 **clear ip bgp**{*|*address*|*peer-group-name*}

Example:

```
Device# clear ip bgp *
```

Resets the routing table on the specified connection.

- Enter an asterisk (*) to specify that all connections be reset.
- Enter an IP address to specify the connection to be reset.

- Enter a peer group name to reset the peer group.

Step 3 **clear ip bgp** {*|*address*|*peer-group-name*} **soft out**

Example:

```
Device# clear ip bgp * soft out
```

(Optional) Performs an outbound soft reset to reset the inbound routing table on the specified connection. Use this command if route refresh is supported.

- Enter an asterisk (*) to specify that all connections be reset.
- Enter an IP address to specify the connection to be reset.
- Enter a peer group name to reset the peer group.

Step 4 **show ip bgp**

Example:

```
Device# show ip bgp
```

Verifies the reset by checking information about the routing table and about BGP neighbors.

Step 5 **show ip bgp neighbors**

Example:

```
Device# show ip bgp neighbors
```

Verifies the reset by checking information about the routing table and about BGP neighbors.

Configuring BGP Decision Attributes

Before you begin

On the Cisco Catalyst 9300 Series Switches, the hardware-supported maximum-path is 8.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 4500
```

Enables a BGP routing process, assign it an AS number, and enter router configuration mode.

Step 4 **bgp best-path as-path ignore**

Example:

```
Device(config-router)# bgp bestpath as-path ignore
```

(Optional) Configures the router to ignore AS path length in selecting a route.

Step 5 **neighbor** [*ip-address* | *peer-group-name*} **next-hop-self**

Example:

```
Device(config-router)# neighbor 10.108.1.1 next-hop-self
```

(Optional) Disables next-hop processing on BGP updates to a neighbor by entering a specific IP address to be used instead of the next-hop address.

Step 6 **neighbor** [*ip-address* | *peer-group-name*} **weight** *weight*

Example:

```
Device(config-router)# neighbor 172.16.12.1 weight 50
```

(Optional) Assign a weight to a neighbor connection. Acceptable values are from 0 to 65535; the largest weight is the preferred route. Routes that are learned through another BGP peer have a default weight of 0; routes that are sourced by the local router have a default weight of 32768.

Step 7 **default-metric***number*

Example:

```
Device(config-router)# default-metric 300
```

(Optional) Sets a MED metric to set preferred paths to external neighbors. All routes without a MED will also be set to this value. The range is 1 to 4294967295. The lowest value is the most desirable.

Step 8 **bgp bestpath med missing-as-worst**

Example:

```
Device(config-router)# bgp bestpath med missing-as-worst
```

(Optional) Configures the switch to consider a missing MED as having a value of infinity, making the path without a MED value the least desirable path.

Step 9 **bgp always-compare med**

Example:

```
Device(config-router)# bgp always-compare-med
```

(Optional) Configures the switch to compare MEDs for paths from neighbors in different autonomous systems. By default, MED comparison is only done among paths in the same AS.

Step 10 **bgp bestpath med confed**

Example:

```
Device(config-router)# bgp bestpath med confed
```

(Optional) Configures the switch to consider the MED in choosing a path from among those advertised by different subautonomous systems within a confederation.

Step 11 **bgp deterministic med**

Example:

```
Device(config-router)# bgp deterministic med
```

(Optional) Configures the switch to consider the MED variable when choosing among routes advertised by different peers in the same AS.

Step 12 **bgp default local-preference** *value*

Example:

```
Device(config-router)# bgp default local-preference 200
```

(Optional) Change the default local preference value. The range is 0 to 4294967295; the default value is 100. The highest local preference value is preferred.

Step 13 **maximum-paths ibgp** *number*

Example:

```
Device(config-router)# maximum-paths 8
or
Device(config-router)# maximum-paths ibgp 2
```

(Optional) Configures the number of paths to be added to the IP routing table. The default is to only enter the best path in the routing table. Having multiple paths allows load-balancing among the paths.

- number : The number of paths to be added to the IP routing table.

  Note
  Although the switch software allows a maximum of 32 equal-cost routes, the actual number of paths per route is dependent on the switch hardware.

- ibgp number : The number of iBGP paths to be added to the IP routing table.

  Note
  The **maximum-paths ibgp** *number* command is supported from Cisco IOS XE Cupertino 17.9.6 and all subsequent Cisco IOS XE Cupertino 17.9.x releases, and from Cisco IOS XE 17.15.1 and all subsequent releases.

Step 14 **end**

Example:

```
Device(config-router)# end
```

Returns to privileged EXEC mode.

Step 15 **show ip bgp**

Example:

```
Device# show ip bgp
```

Verifies the reset by checking information about the routing table and about BGP neighbors.

Step 16 **show ip bgp neighbors**

Example:

```
Device# show ip bgp neighbors
```

Verifies the reset by checking information about the routing table and about BGP neighbors.

Step 17 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering with Route Maps

-

Before you begin

-

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **route-map** *map-tag* [**permit**|**deny**][*sequence-number*]

Example:

```
Device(config)# route-map set-peer-address permit 10
```

Creates a route map, and enter route-map configuration mode.

Step 4 **set ip next-hop** *ip-address* [*...ip-address*][**peer-address**]

Example:

```
Device(config)# set ip next-hop 10.1.1.3
```

(Optional) Sets a route map to disable next-hop processing

- In an inbound route map, set the next hop of matching routes to be the neighbor peering address, overriding third-party next hops.
- In an outbound route map of a BGP peer, set the next hop to the peering address of the local router, disabling the next-hop calculation.

Step 5 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 6 **show route-map** [*map-name*]

Example:

```
Device# show route-map
```

Displays all route maps configured or only the one specified to verify configuration.

Step 7 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering by Access Lists and Neighbors

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip as-path access-list** *access-list-number* {**permit**|**deny**} *as-regular-expressions*

Example:

```
Device(config)# ip as-path access-list 1 deny _65535_
```

Defines a BGP-related access list.

Step 4 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 110
```

Enters BGP router configuration mode.

Step 5 **neighbor** {*ip-address* | *peer-group name*} **filter-list** {*access-list-number* | *name*} {**in** | **out** | **weight** *weight*}

Example:

```
Device(config-router)# neighbor 172.16.1.1 filter-list 1 out
```

Establishes a BGP filter based on an access list.

Step 6 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 7 **show ip bgp neighbors** [**paths** *regular-expression*]

Example:

```
Device# show ip bgp neighbors
```

Verifies the configuration.

Step 8 **copy running-config startup-config**[**paths** *regular-expression*]

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Configuring Prefix Lists for BGP Filtering

You do not need to specify a sequence number when removing a configuration entry. Show commands include the sequence numbers in their output. Before using a prefix list in a command, you must set up the prefix list. To configure prefix list for BGP filtering, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip prefix-list** *list-name*[**seq** *seq-value*]**deny**|**permit** *network/len*[**ge** *ge-value*][**le** *e-value*]

Example:

```
Device(config)# ip prefix-list BLUE permit 172.16.1.0/24
```

Creates a prefix list with an optional sequence number to **deny**or **permit** access for matching conditions. You must enter at least one **permit** or **deny** clause.

- *network/len* is the network number and length (in bits) of the network mask.
- Optional) **ge** and **le** values specify the range of the prefix length to be matched. The specified *ge-value* and *le-value* must satisfy this condition: *len* < *ge-value* < *le-value* < *32*

Step 4 **ip prefix-list** *list-name*[**seq** *seq-value*]**deny**|**permit** *network/len*[**ge** *ge-value*][**le** *e-value*]

Example:

```
Device(config)# ip prefix-list BLUE seq 10 permit 172.24.1.0/24
```

(Optional) Adds an entry to a prefix list, and assign a sequence number to the entry.

Step 5 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 6 **show ip prefix list** [**detail**|**summary**] *name* [*network/len*] [**seq seq-num**] [**longer**] [**first-match**]

Example:

```
Device# show ip prefix list summary test
```

Verifies the configuration by displaying information about a prefix list or prefix list entries.

Step 7 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Configuring BGP Community Filtering

By default, no COMMUNITIES attribute is sent to a neighbor. You can specify that the COMMUNITIES attribute be sent to the neighbor at an IP address by using the **neighbor send-community** router configuration command.

To configure BGP community filter, perform this procedure:

Before you begin

-

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip community-list** *community-list-number*[**permit**|**deny**]| *community-number*[**ge** *ge-value*

Example:

```
Device(config)# ip community-list 1 permit 50000:10
```

Creates a community list, and assigns it a number.

- The *community-list-number* is an integer from 1 to 99 that identifies one or more permit or deny groups of communities.
- The *community-number* is the number that is configured by a **set community** route-map configuration command.

Step 4 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 108
```

Enters BGP router configuration mode.

Step 5 **neighbor** {*ip-address*|*peer-group name*}**send-community**

Example:

```
Device(config-router)# neighbor 172.16.70.23 send-community
```

Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.

Step 6 **set comm-list** *list-num* **delete**

Example:

```
Device(config-router)# set comm-list 500 delete
```

(Optional) Removes communities from the community attribute of an inbound or outbound update that match a standard or extended community list that is specified by a route map.

Step 7 **exit**

Example:

```
Device(config-router)# end
```

Returns to global configuration mode.

Step 8 **ip bgp-community new-format**

Example:

```
Device(config)# ip bgp-community new format
```

(Optional) Displays and parses BGP communities in the format AA:NN. A BGP community is displayed in a two-part format 2 bytes long. The Cisco default community format is in the format NNAA. In the most recent RFC for BGP, a community takes the form AA:NN, where the first part is the AS number and the second part is a 2-byte number.

Step 9 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 10 **show ip bgp community**

Example:

```
Device# show ip bgp community
```

Verifies the configuration.

Step 11 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Configuring BGP Neighbors and Peer Groups

To assign configuration options to an individual neighbor, specify any of these router configuration commands by using the neighbor IP address. To assign the options to a peer group, specify any of the commands by using the peer group name. You can disable a BGP peer or peer group without removing all the configuration information by using the **neighbor shutdown** router configuration command.

To configure BGP neighbors and peer groups, perform this procedure:

Before you begin

-

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

Enters BGP router configuration mode.

Step 4 **neighbor** *peer-group-name* **peer-group**

Example:

Creates a BGP peer group.

Step 5 **neighbor** *ip-address* **peer-group***peer-group-name*

Example:

Makes a BGP neighbor a member of the peer group.

Step 6 **neighbor** {*ip-address* |*peer-group*} **remote-as** *number*

Example:

Specifies a BGP neighbor. If a peer group is not configured with a **remote-as** *number* , use this command to create peer groups containing EBGP neighbors. The range is 1 to 65535.

Step 7 **neighbor** {*ip-address* |*peer-group-name*} **description** *text*

Example:

(Optional) Associates a description with a neighbor.

Step 8 **neighbor** {*ip-address* |*peer-group-name*} **default-originate** [**route-map** *map-name*]

Example:

(Optional) Allows a BGP speaker (the local router) to send the default route 0.0.0.0 to a neighbor for use as a default route.

Step 9 **neighbor** {*ip-address* |*peer-group-name*} **send-community**

Example:

(Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.

Step 10 **neighbor** {*ip-address* |*peer-group-name*} **update-source** *interface*

Example:

(Optional) Allows internal BGP sessions to use any operational interface for TCP connections.

Step 11 **neighbor** {*ip-address* |*peer-group-name*} **ebgp-multihop**

Example:

(Optional) Allows BGP sessions, even when the neighbor is not on a directly connected segment. The multihop session is not established if the only route to the multihop peer's address is the default route (0.0.0.0).

Step 12 **neighbor** {*ip-address* |*peer-group-name*} **local-as** *number*

Example:

(Optional) Specifies an AS number to use as the local AS. The range is 1 to 65535.

Step 13 **neighbor** {*ip-address* |*peer-group-name*} **advertisement-interval** *seconds*

Example:

(Optional) Sets the minimum interval between sending BGP routing updates.

Step 14 **neighbor** {*ip-address* |*peer-group-name*} **maximum-prefix** *maximum [threshold]*

Example:

(Optional) Controls how many prefixes can be received from a neighbor. The range is 1 to 4294967295. The threshold (optional) is the percentage of maximum at which a warning message is generated. The default is 75 percent.

Step 15 **neighbor** {*ip-address* |*peer-group-name*} **next-hop-self**

Example:

(Optional) Disables next-hop processing on the BGP updates to a neighbor.

Step 16 **neighbor** {*ip-address* |*peer-group-name*} **password** *{0-7}string*

Example:

(Optional) Sets MD5 authentication on a TCP connection to a BGP peer. The same password must be configured on both BGP peers, or the connection between them is not made.

The encryption modes supported for the password are:

- 0 - no encryption/plaintext
- 7 - proprietary encryption type

Type 7 is used only for storing the password in the device configuration. The actual value that gets used at the time of BGP session establishment is the MD5 hash of the plaintext password.

- *string* - the password string

Step 17 **neighbor** {*ip-address* |*peer-group-name*} **route-map** *map-name*{**in**|**out**}

Example:

(Optional) Applies a route map to incoming or outgoing routes.

Step 18 **neighbor** {*ip-address* |*peer-group-name*} **send-community**

Example:

(Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.

Step 19 **neighbor** {*ip-address* |*peer-group-name*} **timers** *keepalive holdtime*

Example:

(Optional) Sets timers for the neighbor or peer group.

- The *keepalive* interval is the time within which keepalive messages are sent to peers. The range is 1 to 4294967295 seconds; the default is 60.
- The *holdtime* is the interval after which a peer is declared inactive after not receiving a keepalive message from it. The range is 1 to 4294967295 seconds; the default is 180.

Step 20 **neighbor** {*ip-address* |*peer-group-name*} **weight** *weight*

Example:

(Optional) Specifies a weight for all routes from a neighbor.

Step 21 **neighbor** {*ip-address* |*peer-group-name*} **distribute-list** { *access-list-number*|*name*} {**in**|**out**}

Example:

(Optional) Filter BGP routing updates to or from neighbors, as specified in an access list.

Step 22 **neighbor** {*ip-address* |*peer-group-name*} **filter-list** { *access-list-number* {**in**|**out**|**weight** *weight*}

Example:

(Optional) Establish a BGP filter.

Step 23 **neighbor** {*ip-address* |*peer-group-name*} **version** *value*

Example:

(Optional) Specifies the BGP version to use when communicating with a neighbor.

Step 24 **neighbor** {*ip-address* |*peer-group-name*} **soft-reconfiguration inbound**

Example:

(Optional) Configures the software to start storing received updates.

Step 25 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 26 **show ip bgp neighbors**

Example:

```
Device(config)# end
```

Verifies the configuration.

Step 27 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Configuring Aggregate Addresses in a Routing Table

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 106
```

Enters BGP router configuration mode.

Step 4 **aggregate-address** *address-mask*

Example:

```
Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0
```

Creates an aggregate entry in the BGP routing table. The aggregate route is advertised as coming from the AS, and the atomic aggregate attribute is set to indicate that information might be missing.

Step 5 **aggregate-address** *address-mask* **as-set**

Example:

```
Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 as-set
```

(Optional) Generates AS set path information. This command creates an aggregate entry following the same rules as the previous command, but the advertised path will be an AS_SET consisting of all elements contained in all paths. Do not use this keyword when aggregating many paths because this route must be continually withdrawn and updated.

Step 6 **aggregate-address** *address-mask* **summary-only**

Example:

```
Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 summary-only
```

(Optional) Advertises summary addresses only.

Step 7 **aggregate-address** *address-mask* **suppress-map** *map-name*

Example:

```
Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 suppress-map map1
```

(Optional) Suppresses selected, more specific routes.

Step 8 **aggregate-address** *address-mask* **advertise-map** *map-name*

Example:

```
Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 advertise-map map2
```

(Optional) Generates an aggregate based on conditions that are specified by the route map.

Step 9 **aggregate-address** *address-mask* **attribute-map** *map-name*

Example:

```
Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 attribute-map map3
```

(Optional) Generates an aggregate with attributes that are specified in the route map.

Step 10 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 11 **show ip bgp neighbors** [**advertised-routes**]

Example:

```
Device# show ip bgp neighbors
```

Verifies the configuration.

Step 12 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Example

What to do next

-

Configuring Routing Domain Confederations

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 100
```

Enters BGP router configuration mode.

Step 4 **bgp confederation identifier** *autonomous-system*

Example:

```
Device(config)# bgp confederation identifier 50007
```

Configures a BGP confederation identifier.

Step 5 **bgp confederation identifier** *autonomous-system*[*autonomous-system...*]

Example:

```
Device(config)# bgp confederation peers 51000 51001 51002
```

Specifies the autonomous systems that belong to the confederation and that will be treated as special EBGP peers.

Step 6 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 7 **show ip bgp neighbor**

Example:

```
Device# show ip bgp neighbor
```

Verifies the configuration.

Step 8 **show ip bgp network**

Example:

```
Device# show ip bgp network
```

Verifies the configuration.

Step 9 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Example

What to do next

-

Configuring BGP Route Reflectors

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 101
```

Enters BGP router configuration mode.

Step 4 **neighbor**{ *ip-address*|*peer-group-name*}**route-reflector-client**

Example:

```
Device(config-router)# neighbor 172.16.70.24 route-reflector-client
```

Configures the local router as a BGP route reflector and the specified neighbor as a client.

Step 5 **bgp cluster-id** *cluster-id*

Example:

```
Device(config-router)# bgp cluster-id 10.0.1.2
```

(Optional) Configures the cluster ID if the cluster has more than one route reflector.

Step 6 **no bgp client-to-client reflection**

Example:

```
Device(config-router)# no bgp client-to-client reflection
```

(Optional) Disables client-to-client route reflection. By default, the routes from a route reflector client are reflected to other clients. However, if the clients are fully meshed, the route reflector does not need to reflect routes to clients.

Step 7 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 8 **show ip bgp**

Example:

```
Device# show ip bgp
```

Verifies the configuration. Displays the originator ID and the cluster-list attributes.

Step 9 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Example

What to do next

-

Configuring Route Dampening

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 100
```

Enters BGP router configuration mode.

Step 4 **bgp dampening**

Example:

```
Device(config-router)# bgp dampening
```

Enables BGP route dampening.

Step 5 **bgp dampening***half-life reuse suppress max-suppress*[**route-map** *map*]

Example:

```
Device(config-router)# bgp dampening 30 1500 10000 120
```

(Optional) Changes the default values of route dampening factors.

Step 6 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 7 **show ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask*[**longer-prefix**]}]

Example:

```
Device# show ip bgp flap-statistics
```

(Optional) Monitors the flaps of all paths that are flapping. The statistics are deleted when the route is not suppressed and is stable.

Step 8 **show ip bgp dampened-paths**

Example:

```
Device# show pi bgp dampened-paths
```

(Optional) Displays the dampened routes, including the time remaining before they are suppressed.

Step 9 **clear ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask*[**longer-prefix**]}]

Example:

```
Device# clear ip bgp flap-statistics
```

(Optional) Clears BGP flap statistics to make it less likely that a route will be dampened.

Step 10 **clear ip bgp dampening**

Example:

```
Device# clear ip bgp dampening
```

(Optional) Clears route dampening information, and unsuppress the suppressed routes.

Step 11 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Conditionally Injecting BGP Routes

Before you begin

This task assumes that the IGP is already configured for the BGP peers.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 40000
```

Enters router configuration mode for the specified routing process.

Step 4 **bgp inject-map** *inject-map-name* **exist-map** *exist-map-name*[**copy-attributes**]

Example:

```
Device(config-router)# bgp inject-map ORIGINATE exist-map LEARNED_PATH
```

Specifies the inject map and the exist map for conditional route injection.

- Use the **copy-attributes** keyword to specify that the injected route inherits the attributes of the aggregate route.

Step 5 **exit**

Example:

```
Device(config-router)# exit
```

Exits router configuration mode and enters global configuration mode.

Step 6 **route-map** *map-tag*[**permit**|**deny**][*sequence-number*]

Example:

```
Device(config)# route-map LEARNED_PATH permit 10
```

Configures a route map and enters route map configuration mode.

Step 7 **match ip-address**{*access-list-number [access-list-number ... | access-list-name ...] | access-list-name [access-list-number ... | access-list-name*]|**prefix-list** *prefix-last-name*[*prefix-list-name...*]}

Example:

```
Device(config-route-map)# match ip address prefix-list SOURCE
```

Specifies the aggregate route to which a more specific route will be injected.

- In this example, the prefix list that is named SOURCE is used to redistribute the source of the route.

Step 8 **match ip route-source** {*access-list-number | access-list-name } [access-list-number ... | access-list-name...*]

Example:

```
Device(config-route-map)# match ip route-source prefix-list ROUTE_SOURCE
```

Specifies the match conditions for redistributing the source of the route.

- In this example, the prefix list that is named ROUTE_SOURCE is used to redistribute the source of the route.

Note
The route source is the neighbor address that is configured with the **neighbor remote-as** command. The tracked prefix must come from this neighbor in order for conditional route injection to occur.

Step 9 **exit**

Example:

```
Device(config-router)# exit
```

Exits router configuration mode and enters global configuration mode.

Step 10 **route-map** *map-tag*[**permit**|**deny**][*sequence-number*]

Example:

```
Device(config)# route-map ORIGINATE permit 10
```

Configures a route map and enters route map configuration mode.

Step 11 **set ip-address**{*access-list-number [access-list-number ... | access-list-name ...] | access-list-name [access-list-number ... | access-list-name*]|**prefix-list** *prefix-last-name*[*prefix-list-name...*]}

Example:

```
Device(config-route-map)# set ip address prefix-list ORIGINATED_ROUTES
```

Specifies the routes to be injected. In this example, the prefix list that is named originated_routes is used to redistribute the source of the route.

Step 12 **set community**{*community-number*[**additive**][*well-known-community*]|**none**}

Example:

```
Device(config-route-map)# set community 14616:555 additive
```

Sets the BGP community attribute of the injected route.

Step 13 **exit**

Example:

```
Device(config-route-map)# exit
```

Exits router configuration mode and enters global configuration mode.

Step 14 **ip prefix list** *list-name*[**seq***seq-value*] {**deny** *network / length* | **permit** *network / length*} [**ge** *ge-value*] [**le** *le-value*]

Example:

```
Device(config)# ip prefix-list SOURCE permit 10.1.1.0/24
```

Configures a prefix list. In this example, the prefix list that is named SOURCE is configured to permit routes from network 10.1.1.0/24.

Step 15 **Example:**

Repeat Step 14 for every prefix list to be created.

Step 16 **exit**

Example:

```
Device(config)# exit
```

Exits router configuration mode and enters global configuration mode.

Step 17 **show ip bgp injected-paths**

Example:

```
Device# show ip bgp injected-paths
```

(Optional) Displays information about injected paths.

Configuring Peer Session Templates

The following example creates a peer session template that is named INTERNAL-BGP in session-template configuration mode:

```
router bgp 45000
 template peer-session INTERNAL-BGP
 remote-as 50000
 timers 30 300
 exit-peer-session
```

The following example creates a peer session template named CORE1. This example inherits the configuration of the peer session template named INTERNAL-BGP.

```
router bgp 45000
 template peer-session CORE1
 description CORE-123
 update-source loopback 1
 inherit peer-session INTERNAL-BGP
 exit-peer-session
```

The following example configures the 192.168.3.2 neighbor to inherit the CORE1 peer session template. The 192.168.3.2 neighbor will also indirectly inherit the configuration from the peer session template named

INTERNAL-BGP. The explicit remote-as statement is required for the neighbor inherit statement to work. If a peering is not configured, the specified neighbor will not accept the session template.

```
router bgp 45000
 neighbor 192.168.3.2 remote-as 50000
 neighbor 192.168.3.2 inherit peer-session CORE1
```

Configuring a basic peer session template

Perform this task to create a basic peer session template with general BGP routing session commands that can be applied to many neighbors using one of the next two tasks.

The commands in Step 5 and 6 are optional and could be replaced with any supported general session commands. The following restrictions apply to the peer session templates:

- A peer session template can directly inherit only one session template, and each inherited session template can also contain one indirectly inherited session template. So, a neighbor or neighbor group can be configured with only one directly applied peer session template and seven additional indirectly inherited peer session templates.
- A BGP neighbor cannot be configured to work with both peer groups and peer templates. A BGP neighbor can be configured to belong only to a peer group or to inherit policies only from peer templates.

To configure a basic peer session template, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp autonomous-system- number**

Example:

```
Device(config)# router bgp
101
```

Enters router configuration mode and creates a BGP routing process.

Step 4 **template peer-session sessiontemplate-name**

Example:

```
Device(config-router)# template peer-session
INTERNAL-BGP
```

Enters session-template configuration mode and creates a peer session template.

Step 5 **remote-as autonomous-system- number**

Example:

```
Device(config-router-stmp)# remote-as 202
```

(Optional) Configures peering with a remote neighbor in the specified autonomous system.

Note
Any supported general session

command can be used here. For a list of the supported commands, see the "Restrictions" section.

Step 6 **timers keepalive-interval holdtime**

Example:

```
Device(config-router-stmp)# timers 30 300
```

(Optional) Configures BGP keepalive and hold timers.

The hold time must be at least twice the keepalive time.

Note
Any supported general session

command can be used here. For a list of the supported commands, see the "Restrictions" section.

Step 7 **end**

Example:

```
Device(config-router)# end
```

Exits session-template configuration mode and returns to privileged EXEC mode.

Step 8 **show ip bgp template peersession [session-template- name]**

Example:

```
Device# show ip bgp template peer-session
```

Displays locally configured peer session templates.

The output can be filtered to display a single peer policy template with the sessiontemplate-name argument. This command also supports all standard output modifiers.

Configuring Peer Session Template Inheritance with the inherit peer-session Command

Note The commands in Steps 5 and 6 are optional and could be replaced with any supported general session commands.

To configure peer session template inheritance, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 101
```

Enters router configuration mode and creates a BGP routing process.

Step 4 **description***text-string*

Example:

```
Device(config-router-stmp)# description CORE-123
```

(Optional) Configures a description. The text string can be up to 80 characters.

Note
Any supported general session command can be used here. For a list of the supported commands, see the “Restrictions” section.

Step 5 **update-source** *interface-type interface-number*

Example:

```
Device(config-router-stmp)# update-source loopback 1
```

(Optional) Configures a router to select a specific source or interface to receive routing table updates.

The example uses a loopback interface. The advantage to this configuration is that the loopback interface is not as susceptible to the effects of a flapping interface.

Note
Any supported general session command can be used here. For a list of the supported commands, see the “Restrictions” section.

Step 6 **inherit peer-session** *session-template-name*

Example:

```
Device(config-router-stmp)# inherit peer-session INTERNAL-BGP
```

Configures this peer session template to inherit the configuration of another peer session template.

The example configures this peer session template to inherit the configuration from INTERNAL-BGP. This template can be applied to a neighbor, and the configuration INTERNAL-BGP will be applied indirectly. No additional peer session templates can be directly applied. However, the directly inherited template can contain up to seven indirectly inherited peer session templates.

Step 7 **end**

Example:

```
Device(config-router)# end
```

Returns to privileged EXEC mode.

Step 8 **show ip bgp template peer-session** [*session-template-name*]

Example:

```
Device# show ip bgp template peer-session
```

Displays locally configured peer session templates.

The output can be filtered to display a single peer policy template with the optional *session-template-name* argument. This command also supports all standard output modifiers.

Configuring Peer Session Template Inheritance with the neighbor inherit peer-session Command

To configure peer session template inheritance, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 101
```

Enters router configuration mode and creates a BGP routing process.

Step 4 **neighbor** *ip-address* **remote-as***autonomous-system-number*

Example:

```
Device(config-router)# neighbor 172.16.0.1 remote-as 202
```

Configures a peering session with the specified neighbor.

The explicit **remote-as** statement is required for the neighbor inherit statement in Step 5 to work. If a peering is not configured, the specified neighbor in Step 5 will not accept the session template.

Step 5 **neighbor** *ip-address* **inherit peer-session***session-template-name*

Example:

```
Device(config-router)# neighbor 172.16.0.1 inherit peer-session CORE1
```

Sends a peer session template to a neighbor so that the neighbor can inherit the configuration.

The example configures a device to send the peer session template named CORE1 to the 172.16.0.1 neighbor to inherit. This template can be applied to a neighbor, and if another peer session template is indirectly inherited in CORE1, the indirectly inherited configuration will also be applied. No additional peer session templates can be directly applied. However, the directly inherited template can also inherit up to seven additional indirectly inherited peer session templates.

Step 6 **end**

Example:

```
Device(config-router)# end
```

Returns to privileged EXEC mode.

Step 7 **show ip bgp template peer-session** [*session-template-name*]

Example:

```
Device# show ip bgp template peer-session
```

Displays locally configured peer session templates.

The output can be filtered to display a single peer policy template with the optional *session-template-name* argument. This command also supports all standard output modifiers.

Configuring Peer Policy Templates

Procedure

Example:

Configuring Basic Peer Policy Templates

Note The commands in Steps 5 through 7 are optional and could be replaced with any supported BGP policy configuration commands.

Note The following restrictions apply to the peer policy templates:

- A peer policy template can directly or indirectly inherit up to eight peer policy templates.
- • A BGP neighbor cannot be configured to work with both peer groups and peer templates. A BGP neighbor can be configured to belong only to a peer group or to inherit policies only from peer templates.

To configure a basic peer policy template, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 101
```

Enters router configuration mode and creates a BGP routing process.

Step 4 **template peer-policy** *policy-template-name*

Example:

```
Device(config-router)# template peer-policy GLOBAL
```

Enters policy-template configuration mode and creates a peer policy template.

Step 5 **maximum-prefix** *prefix-limit*[*threshold*][**restart** *restart-interval* | **warning-only**]

Example:

```
Device(config-router-ptmp)# maximum-prefix 10000
```

(Optional) Configures the maximum number of prefixes that a neighbor accept from this peer.

Note
Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the “Peer Policy Templates” section.

Step 6 **weight** *weight-value*

Example:

```
Device(config-router-ptmp)# weight 300
```

(Optional) Sets the default weight for routes that are sent from this neighbor.

Note
Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the "Peer Policy Templates" section.

Step 7 **prefix-list** *prefix-list-name* {**in**|**out**}

Example:

```
Device(config-router-ptmp)# prefix-list NO-MARKETING in
```

(Optional) Filters prefixes that are received by the router or sent from the router. The prefix list in the example filters inbound internal addresses.

Note
Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the "Peer Policy Templates" section.

Step 8 **end**

Example:

```
Device(config-router-ptmp)# end
```

Returns to privileged EXEC mode.

Configuring Peer Policy Template Inheritance with the inherit peer-policy Command

To configure peer policy template inheritance, perform this procedure:

Before you begin

Note The commands in Steps 5 and 6 are optional and could be replaced with any supported BGP policy configuration commands.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 45000
```

Enters router configuration mode and creates a BGP routing process.

Step 4 **template peer-policy** *policy-template-name*

Example:

```
Device(config-router)# template peer-policy NETWORK1
```

Enter policy-template configuration mode and creates a peer policy template.

Step 5 **route-map** *map-name* {**in**|**out** }

Example:

```
Device(config-router-ptmp)# route-map ROUTE in
```

(Optional) Applies the specified route map to inbound or outbound routes.

Note
Any supported BGP policy configuration command can be used here.

Step 6 **inherit peer-policy** *policy-template-name sequence-number*

Example:

```
Device(config-router-ptmp)# inherit peer-policy GLOBAL 10
```

Configures the peer policy template to inherit the configuration of another peer policy template.

- The *sequence-number* argument sets the order in which the peer policy template is evaluated. Like a route map sequence number, the lowest sequence number is evaluated first.
- The example configures this peer policy template to inherit the configuration from GLOBAL. If the template created in these steps is applied to a neighbor, the configuration GLOBAL will also be inherited and applied indirectly. Up to six additional peer policy templates can be indirectly inherited from GLOBAL for a total of eight directly applied and indirectly inherited peer policy templates.
- This template in the example will be evaluated first if no other templates are configured with a lower sequence number.

Step 7 **end**

Example:

```
Device(config-router-ptmp)# end
```

Returns to privileged EXEC mode.

Step 8 **show ip bgp template peer-policy** [*policy-template-name*[**detail**]]

Example:

```
Device# show ip bgp template peer-policy NETWORK1 detail
```

Displays locally configured peer policy templates.

- • The output can be filtered to display a single peer policy template with the *policy-template-name* argument. This command also supports all standard output modifiers.
- Use the **detail** keyword to display detailed policy information.

Example

The following sample output of the **show ip bgp template peer-policy** command with the detail keyword displays details of the policy named NETWORK1. The output in this example shows that the GLOBAL template was inherited. Details of route map and prefix list configurations are also displayed.

```
Device# show ip bgp template peer-policy NETWORK1 detail
Template:NETWORK1, index:2.
Local policies:0x1, Inherited polices:0x80840
This template inherits:
  GLOBAL, index:1, seq_no:10, flags:0x1
Locally configured policies:
  route-map ROUTE in
Inherited policies:
  prefix-list NO-MARKETING in
  weight 300
  maximum-prefix 10000
 Template:NETWORK1 <detail>
Locally configured policies:
  route-map ROUTE in
route-map ROUTE, permit, sequence 10
  Match clauses:
    ip address prefix-lists: DEFAULT
ip prefix-list DEFAULT: 1 entries
   seq 5 permit 10.1.1.0/24
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
Inherited policies:
  prefix-list NO-MARKETING in
ip prefix-list NO-MARKETING: 1 entries
   seq 5 deny 10.2.2.0/24
```

Configuring Peer Policy Template Inheritance with the neighbor inherit peer-policy Command

To configure peer policy template, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **router bgp** *autonomous-system*

Example:

```
Device(config)# router bgp 45000
```

Enters router configuration mode and creates a BGP routing process.

Step 4 **neighbor** *ip-address* **remote-as***autonomous-system-number*

Example:

```
Device(config-router)# neighbor 192.168.1.2 remote-as 40000
```

Configures a peering session with the specified neighbor.

The explicit **remote-as** statement is required for the **neighbor inherit** statement in Step 6 to work. If a peering is not configured, the specified neighbor in Step 6 will not accept the session template.

Step 5 **address-family ipv4** [**multicast**|**unicast**|**vrf** *vrf-name*]

Example:

```
Device(config-router)# address-family ipv4 unicast
```

Enters address family configuration mode to configure a neighbor to accept address family-specific command configurations.

Step 6 **neighbor** *ip-address* **inherit peer-session***policy-template-name*

Example:

```
Device(config-router-af)# neighbor 192.168.1.2 inherit peer-policy GLOBAL
```

Sends a peer session template to a neighbor so that the neighbor can inherit the configuration.

The example configures a router to send the peer policy template that is named GLOBAL to the 192.168.1.2 neighbor to inherit. This template can be applied to a neighbor, and if another peer policy template is indirectly inherited from GLOBAL, the indirectly inherited configuration will also be applied. Up to seven additional peer policy templates can be indirectly inherited from GLOBAL.

Step 7 **end**

Example:

```
Device(config-router-af)# end
```

Exits address family configuration mode and returns to privileged EXEC mode.

Step 8 **show ip bgp neighbors** [*ip-address* [**policy**[**detail**]]]

Example:

```
Device# show ip bgp neighbors 192.168.1.2 policy
```

Displays locally configured peer policy templates.

- The output can be filtered to display a single peer policy template with the *policy-template-name* argument. This command also supports all standard output modifiers.
- Use the **policy** keyword to display the policies that are applied to this neighbor per address family.
- Use the **detail** keyword to display detailed policy information.

Example

The following sample output shows the policies that are applied to the neighbor at 192.168.1.2. The output displays both inherited policies and policies that are configured on the neighbor device. Inherited polices are policies that the neighbor inherits from a peer-group or a peer-policy template.

```
Device# show ip bgp neighbors 192.168.1.2 policy
Neighbor: 192.168.1.2, Address-Family: IPv4 Unicast
Locally configured policies:
 route-map ROUTE in
Inherited polices:
 prefix-list NO-MARKETING in
 route-map ROUTE in
 weight 300
 maximum-prefix 10000
```

Configuring BGP Route Map Next-hop Self

To configure BGP route map next-hop self, perform this procedure:

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **route-map** *map-tag* **permit** *sequence-number*

Example:

```
Device(config)# route-map static-nexthop-rewrite permit 10
```

Defines conditions for redistributing routes from one routing protocol to another routing protocol and enters route-map configuration mode.

Step 4 **match source-protocol** *source-protocol*

Example:

```
Device(config-route-map)# match source-protocol static
```

Matches Enhanced Interior Gateway Routing Protocol (EIGRP) external routes based on a source protocol.

Step 5 **set ip next-hop self**

Example:

```
Device(config-route-map)# set ip next-hop self
```

Configure local routes (for BGP only) with next hop of self.

Step 6 **exit**

Example:

```
Device(config-route-map)# exit
```

Exits route-map configuration mode and enters global configuration mode.

Step 7 **route-map** *map-tag* **permit***sequence-number*

Example:

```
Device(config)# route-map static-nexthop-rewrite permit 20
```

Defines conditions for redistributing routes from one routing protocol to another routing protocol and enters route-map configuration mode.

Step 8 **match route-type internal**

Example:

```
Device(config-route-map)# match route-type internal
```

Redistributes routes of the specified type.

Step 9 **match route-type external**

Example:

```
Device(config-route-map)# match route-type external
```

Redistributes routes of the specified type.

Step 10 **match source-protocol** *source-protocol*

Example:

```
Device(config-route-map)# match source-protocol connected
```

Matches Enhanced Interior Gateway Routing Protocol (EIGRP) external routes based on a source protocol.

Step 11 **exit**

Example:

```
Device(config-route-map)# exit
```

Exits route-map configuration mode and enters global configuration mode.

Step 12 **router bgp** *autonomous-system-number*

Example:

```
Device(config)# router bgp 45000
```

Enters router configuration mode and creates a BGP routing process.

Step 13 **neighbor** {*ip-address* |*ipv6-address*|*peer-group-name*} **remote-as** *autonomous-system-number*

Example:

```
Device(config-router)# neighbor 172.16.232.50 remote-as 65001
```

Adds an entry to the BGP or multiprotocol BGP neighbor table.

Step 14 **address-family vpnv4**

Example:

```
Device(config-router)# address-family vpnv4
```

Specifies the VPNv4 address family and enters address family configuration mode.

Step 15 **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **activate**

Example:

```
Device(config-router-af)# neighbor 172.16.232.50 activate
```

Enables the exchange of information with a Border Gateway Protocol (BGP) neighbor.

Step 16 **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **next-hop unchanged allpaths**

Example:

```
Device(config-router-af)# neighbor 172.16.232.50 next-hop unchanged allpaths
```

Enables an external EBGP peer that is configured as multihop to propagate the next hop unchanged.

- *string* - the password string

Step 17 **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **route-map** *map-name* **out**

Example:

```
Device(config-router-af)# neighbor 172.16.232.50 route-map static-nexthop-rewrite out
```

Applies a route map to an outgoing route.

Step 18 **exit**

Example:

```
Device(config-router-af)# exit
```

Exits address family configuration mode and enters router configuration mode.

Step 19 **address-family ipv4 [unicast | multicast | vrf** |**vrf** *vrf-name*

Example:

```
Device(config-router)# address-family ipv4 unicast vrf inside
```

Specifies the IPv4 address family and enters address family configuration mode.

Step 20 **bgp route-map priority**

Example:

```
Device(config-router-af)# bgp route-map priority
```

Configures the route map priority for the local BGP routing process

Step 21 **redistribute** *protocol*

Example:

```
Device(config-router-af)# redistribute static
```

Redistributes routes from one routing domain into another routing domain.

Step 22 **redistribute** *protocol*

Example:

```
Device(config-router-af)# redistribute connected
```

Redistributes routes from one routing domain into another routing domain.

Step 23 **exit-address-family**

Example:

```
Device(config-router-af)# exit address-family
```

Exits address family configuration mode and enters router configuration mode.

Step 24 **end**

Example:

```
Device(config-router)# end
```

Exits router configuration mode and enters privileged EXEC mode.

Configuration Examples for BGP

Configuring Conditional BGP Route Injection

The following sample output is similar to the output that will be displayed when the **show ip bgp injected-paths** command is entered:

```
Device# show ip bgp injected-paths

BGP table version is 11, local router ID is 10.0.0.1
Status codes:s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes:i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.0.0       10.0.0.2                               0 ?
*> 172.17.0.0/16    10.0.0.2                               0 ?
```

Configuring Peer Session Templates

The following example creates a peer session template that is named INTERNAL-BGP in session-template configuration mode:

```
router bgp 45000
 template peer-session INTERNAL-BGP
 remote-as 50000
 timers 30 300
 exit-peer-session
```

The following example creates a peer session template named CORE1. This example inherits the configuration of the peer session template named INTERNAL-BGP.

```
router bgp 45000
 template peer-session CORE1
 description CORE-123
 update-source loopback 1
 inherit peer-session INTERNAL-BGP
 exit-peer-session
```

The following example configures the 192.168.3.2 neighbor to inherit the CORE1 peer session template. The 192.168.3.2 neighbor will also indirectly inherit the configuration from the peer session template named INTERNAL-BGP. The explicit remote-as statement is required for the neighbor inherit statement to work. If a peering is not configured, the specified neighbor will not accept the session template.

```
router bgp 45000
 neighbor 192.168.3.2 remote-as 50000
 neighbor 192.168.3.2 inherit peer-session CORE1
```

Configuring Peer Policy Templates

The following example creates a peer policy template that is named GLOBAL and enters policy-template configuration mode:

```
router bgp 45000
 template peer-policy GLOBAL
  weight 1000
  maximum-prefix 5000
  prefix-list NO_SALES in
  exit-peer-policy
```

The following example creates a peer policy template that is named PRIMARY-IN and enters policy-template configuration mode:

```
router bgp 45000
 template peer-policy PRIMARY-IN
  prefix-list ALLOW-PRIMARY-A in
  route-map SET-LOCAL in
  weight 2345
  default-originate
  exit-peer-policy
```

The following example creates a peer policy template named CUSTOMER-A. This peer policy template is configured to inherit the configuration from the peer policy templates that are named PRIMARY-IN and GLOBAL.

```
router bgp 45000
 template peer-policy CUSTOMER-A
  route-map SET-COMMUNITY in
  filter-list 20 in
  inherit peer-policy PRIMARY-IN 20
  inherit peer-policy GLOBAL 10
  exit-peer-policy
```

The following example configures the 192.168.2.2 neighbor in address family mode to inherit the peer policy template named CUSTOMER-A. Assuming this example is a continuation of the example above, because the peer policy template named CUSTOMER-A above inherited the configuration from the templates that are named PRIMARY-IN and GLOBAL, the 192.168.2.2 neighbor will also indirectly inherit the peer policy templates that are named PRIMARY-IN and GLOBAL.

```
router bgp 45000
 neighbor 192.168.2.2 remote-as 50000
 address-family ipv4 unicast
```

```
  neighbor 192.168.2.2 inherit peer-policy CUSTOMER-A
  end
```

Configuring BGP Route Map next-hop self

This section contains an example of how to configure BGP Route Map next-hop self.

In this example, a route map is configured that matches the networks where you wish to override settings for bgp next-hop unchanged and bgp next-hop unchanged allpath. Subsequently, next-hop self is configured. After this, the bgp route map priority is configured for the specified address family so that the previously specified route map takes priority over the settings for bgp next-hop unchanged and bgp next-hop unchanged allpath. This configuration results in static routes being redistributed with a next hop of self, but connected routes and routes learned via IBGP or EBGP continue to be redistributed with an unchanged next hop.

```
route-map static-nexthop-rewrite permit 10
 match source-protocol static
 set ip next-hop self
route-map static-nexthop-rewrite permit 20
 match route-type internal
 match route-type external
 match source-protocol connected
!
router bgp 65000
 neighbor 172.16.232.50 remote-as 65001
 address-family vpnv4
   neighbor 172.16.232.50 activate
   neighbor 172.16.232.50 next-hop unchanged allpaths
   neighbor 172.16.232.50 route-map static-nexthop-rewrite out
 exit-address-family
 address-family ipv4 unicast vrf inside
   bgp route-map priority
   redistribute static
   redistribute connected
exit-address-family
end
```

Monitoring and Maintaining BGP

To effectively monitor and maintain Border Gateway Protocol (BGP) operations, network administrators have access to various commands that allow for clearing and displaying BGP-related information. This functionality is crucial for troubleshooting, optimizing resource utilization, and ensuring network stability.

Administrators can clear the contents of specific BGP caches, tables, or databases when data integrity is suspected to be compromised. This helps in resetting the BGP state and resolving potential issues.

Furthermore, detailed statistics can be displayed, including the contents of BGP routing tables, caches, and databases. This information provides insights into resource consumption, aids in diagnosing network problems, and helps in understanding node reachability and packet forwarding paths.

Workflow

The following table lists privileged EXEC commands for clearing and displaying BGP information:

1.	**clear ip bgp address** *address*	Resets a particular BGP connection.
	clear ip bgp *	Resets all BGP connections.
	clear ip bgp peer-group *tag*	Removes all members of a BGP peer group.

show ip bgp *prefix*	Displays peer groups and peers not in peer groups to which the prefix has been advertised. Also displays prefix attributes such as the next hop and the local prefix.
show ip bgp cidr-only	Displays all BGP routes that contain subnet and supernet network masks.
show ip bgp community [*community-number*] [**exact**]	Displays routes that belong to the specified communities.
show ip bgp community *community-number* **exact-match**]	Displays routes that are permitted by the community list.
show ip bgp filter-list *access-list-number*	Displays routes that are matched by the specified AS path access list.
show ip bgp inconsistent-as	Displays the routes with inconsistent originating autonomous systems.
show ip bgp regexp *regular-expression*	Displays the routes that have an AS path that matches the specified regular expression entered on the command line.
show ip bgp	Displays the contents of the BGP routing table.
show ip bgp neighbors [*address*]	Displays detailed information on the BGP and TCP connections to individual neighbors.
show ip bgp neighbors [*address*] [**advertised-routes** |**dampened-routes** |**flap-statistics** |**paths** *regular-expression*|**received-routes** | **routes**]	Displays routes learned from a particular BGP neighbor.
show ip bgp paths	Displays all BGP paths in the database.
show ip bgp peer-group [*tag*] [**summary**]	Displays information about BGP peer groups.
show ip bgp summary	Displays the status of all BGP connections.

The **bgp log-neighbor changes** command is enabled by default. It allows to log messages that are generated when a BGP neighbor resets, comes up, or goes down.