



Configuring BGP

- [Restrictions for BGP, on page 1](#)
- [Information About BGP, on page 1](#)
- [How to Configure BGP, on page 13](#)
- [Configuration Examples for BGP, on page 54](#)
- [Monitoring and Maintaining BGP, on page 56](#)
- [Feature History for Border Gateway Protocol, on page 57](#)

Restrictions for BGP

- When you configure the **bgp graceful-restart command**, ensure that the BGP hold time is higher than the minimum graceful restart time for the Extended Fast Software Upgrade (xFSU) capable devices. The minimum graceful restart time is specific to a device. A lower hold time value is configurable, however BGP is not supported in an xFSU event. If a BGP peer sends an unsupported hold timer in the BGP open message, the proposed hold timer is accepted, but BGP is not supported on an xFSU event.
- Layer 3 forwarding is delayed until routing tables are populated on a device when you switch on the device or execute the **clear ip bgp** command.



Note 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.

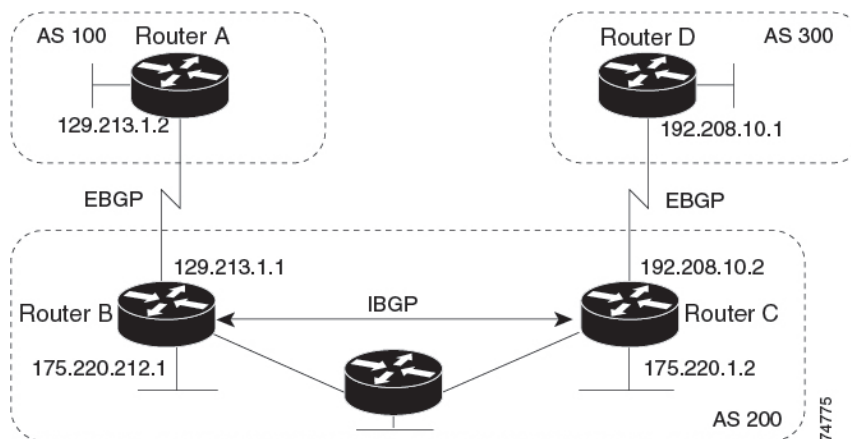
Information About BGP

The Border Gateway Protocol (BGP) is an exterior gateway protocol that is used to set up an interdomain routing system that guarantees the loop-free exchange of routing information between autonomous systems. Autonomous systems are made up of routers that operate under the same administration and that run Interior Gateway Protocols (IGPs), such as RIP or OSPF, within their boundaries and that interconnect by using an Exterior Gateway Protocol (EGP). BGP Version 4 is the standard EGP for interdomain routing in the Internet. The protocol is defined in RFCs 1163, 1267, and 1771.

BGP Network Topology

Routers that belong to the same autonomous system (AS) and that exchange BGP updates run internal BGP (IBGP), and routers that belong to different autonomous systems and that exchange BGP updates run external BGP (EBGP). Most configuration commands are the same for configuring EBGP and IBGP. The difference is that the routing updates are exchanged either between autonomous systems (EBGP) or within an AS (IBGP). The figure given below shows a network that is running both EBGP and IBGP.

Figure 1: EBGP, IBGP, and Multiple Autonomous Systems



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 that run a BGP routing process are often referred to as BGP speakers. BGP uses the Transmission Control Protocol (TCP) as its transport protocol (specifically port 179). Two BGP speakers that have a TCP connection to each other for exchanging routing information are known as peers or neighbors. In the above figure, Routers A and B are BGP peers, as are Routers B and C and Routers C and D. The routing information is a series of AS numbers that describe the full path to the destination network. BGP uses this information to construct a loop-free map of autonomous systems.

The network has these characteristics:

- Routers A and B are running EBGP, and Routers B and C are running IBGP. Note that the EBGP peers are directly connected and that the IBGP peers are not. As long as there is an IGP running that allows the two neighbors to reach one another, IBGP peers do not have to be directly connected.
- All BGP speakers within an AS must establish a peer relationship with each other. That is, the BGP speakers within an AS must be fully meshed logically. BGP4 provides two techniques that reduce the requirement for a logical full mesh: confederations and route reflectors.
- AS 200 is a transit AS for AS 100 and AS 300—that is, AS 200 is used to transfer packets between AS 100 and AS 300.

BGP peers initially exchange their full BGP routing tables and then send only incremental updates. BGP peers also exchange keepalive messages (to ensure that the connection is up) and notification messages (in response to errors or special conditions).

In BGP, each route consists of a network number, a list of autonomous systems that information has passed through (the autonomous system path), and a list of other path attributes. The primary function of a BGP

system is to exchange network reachability information, including information about the list of AS paths, with other BGP systems. This information can be used to determine AS connectivity, to prune routing loops, and to enforce AS-level policy decisions.

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

BGP Version 4 supports classless interdomain routing (CIDR) so you can reduce the size of your routing tables by creating aggregate routes, resulting in supernets. CIDR eliminates the concept of network classes within BGP and supports the advertising of IP prefixes.

Nonstop Forwarding Awareness

To enable this feature with BGP routing, you need to enable Graceful Restart. When the neighboring router is NSF-capable, and this feature is enabled, the Layer 3 device continues to forward packets from the neighboring router, during the interval when the primary Route Processor (RP) in a device is failing and the backup RP is taking over, or while the primary RP is manually reloaded for a nondisruptive software upgrade.

Information About BGP Routing

To enable BGP routing, you establish a BGP routing process and define the local network. Because BGP must completely recognize the relationships with its neighbors, you must also specify a BGP neighbor.

BGP supports two kinds of neighbors: internal and external. Internal neighbors are in the same AS; external neighbors are in different autonomous systems. External neighbors are usually adjacent to each other and share a subnet, but internal neighbors can be anywhere in the same AS.

The switch supports the use of private AS numbers, usually assigned by service providers and given to systems whose routes are not advertised to external neighbors. The private AS numbers are from 64512 to 65535. You can configure external neighbors to remove private AS numbers from the AS path by using the **neighbor remove-private-as** router configuration command. Then when an update is passed to an external neighbor, if the AS path includes private AS numbers, these numbers are dropped.

If your AS will be passing traffic through it from another AS to a third AS, it is important to be consistent about the routes it advertises. If BGP advertised a route before all routers in the network had learned about the route through the IGP, the AS might receive traffic that some routers could not yet route. To prevent this from happening, BGP must wait until the IGP has propagated information across the AS so that BGP is synchronized with the IGP. Synchronization is enabled by default. If your AS does not pass traffic from one AS to another AS, or if all routers in your autonomous systems are running BGP, you can disable synchronization, which allows your network to carry fewer routes in the IGP and allows BGP to converge more quickly.

Routing Policy Changes

Routing policies for a peer include all the configurations that might affect inbound or outbound routing table updates. When you have defined two routers as BGP neighbors, they form a BGP connection and exchange routing information. If you later change a BGP filter, weight, distance, version, or timer, or make a similar configuration change, you must reset the BGP sessions so that the configuration changes take effect.

There are two types of reset, hard reset and soft reset. Cisco IOS Releases 12.1 and later support a soft reset without any prior configuration. To use a soft reset without preconfiguration, both BGP peers must support the soft route refresh capability, which is advertised in the OPEN message sent when the peers establish a TCP session. A soft reset allows the dynamic exchange of route refresh requests and routing information between BGP routers and the subsequent re-advertisement of the respective outbound routing table.

- When soft reset generates inbound updates from a neighbor, it is called dynamic inbound soft reset.
- When soft reset sends a set of updates to a neighbor, it is called outbound soft reset.

A soft inbound reset causes the new inbound policy to take effect. A soft outbound reset causes the new local outbound policy to take effect without resetting the BGP session. As a new set of updates is sent during outbound policy reset, a new inbound policy can also take effect.

The table that is given below lists the advantages and disadvantages hard reset and soft reset.

Table 1: Advantages and Disadvantages of Hard and Soft Resets

| Type of Reset | Advantages | Disadvantages |
|----------------------------|--|---|
| Hard reset | No memory overhead | The prefixes in the BGP, IP, and FIB tables provided by the neighbor are lost. Not recommended. |
| Outbound soft reset | No configuration, no storing of routing table updates | Does not reset inbound routing table updates |
| Dynamic inbound soft reset | Does not clear the BGP session and cache Does not require storing of routing table updates and has no memory overhead | Both BGP routers must support the soft route refresh capability (in Cisco IOS Release 12.1 and later) |

BGP Decision Attributes

When a BGP speaker receives updates from multiple autonomous systems that describe different paths to the same destination, it must choose the single best path for reaching that destination. When chosen, the selected path is entered into the BGP routing table and propagated to its neighbors. The decision is based on the value of attributes that the update contains and other BGP-configurable factors.

When a BGP peer learns two EBGP paths for a prefix from a neighboring AS, it chooses the best path and inserts that path in the IP routing table. If BGP multipath support is enabled and the EBGP paths are learned from the same neighboring autonomous systems, instead of a single best path, multiple paths are installed in the IP routing table. Then, during packet switching, per-packet or per-destination load-balancing is performed among the multiple paths. The **maximum-paths** router configuration command controls the number of paths allowed.

These factors summarize the order in which BGP evaluates the attributes for choosing the best path:

1. If the path specifies a next hop that is inaccessible, drop the update. The BGP next-hop attribute, automatically determined by the software, is the IP address of the next hop that is going to be used to reach a destination. For EBGP, this is usually the IP address of the neighbor that is specified by the **neighbor remote-as router** configuration command. You can disable next-hop processing by using route maps or the **neighbor next-hop-self** router configuration command.

2. Prefer the path with the largest weight (a Cisco proprietary parameter). The weight attribute is local to the router and not propagated in routing updates. By default, the weight attribute is 32768 for paths that the router originates and zero for other paths. Routes with the largest weight are preferred. You can use access lists, route maps, or the **neighbor weight** router configuration command to set weights.
3. Prefer the route with the highest local preference. Local preference is part of the routing update and exchanged among routers in the same AS. The default value of the local preference attribute is 100. You can set local preference by using the **bgp default local-preference** router configuration command or by using a route map.
4. Prefer the route that was originated by BGP running on the local router.
5. Prefer the route with the shortest AS path.
6. Prefer the route with the lowest origin type. An interior route or IGP is lower than a route learned by EGP, and an EGP-learned route is lower than one of unknown origin or learned in another way.
7. Prefer the route with the lowest multi-exit discriminator (MED) metric attribute if the neighboring AS is the same for all routes considered. You can configure the MED by using route maps or by using the **default-metric** router configuration command. When an update is sent to an IBGP peer, the MED is included.
8. Prefer the external (EBGP) path over the internal (IBGP) path.
9. Prefer the route that can be reached through the closest IGP neighbor (the lowest IGP metric). This means that the router will prefer the shortest internal path within the AS to reach the destination (the shortest path to the BGP next-hop).
10. If the following conditions are all true, insert the route for this path into the IP routing table:
 - Both the best route and this route are external.
 - Both the best route and this route are from the same neighboring autonomous system.
 - Maximum-paths is enabled.
11. If multipath is not enabled, prefer the route with the lowest IP address value for the BGP router ID. The router ID is usually the highest IP address on the router or the loopback (virtual) address, but might be implementation-specific.

Route Maps

Within BGP, route maps can be used to control and to modify routing information and to define the conditions by which routes are redistributed between routing domains. Each route map has a name that identifies the route map (*map tag*) and an optional sequence number.

BGP Filtering

You can filter BGP advertisements by using AS-path filters, such as the **as-path access-list** global configuration command and the **neighbor filter-list** router configuration command. You can also use access lists with the **neighbor distribute-list** router configuration command. Distribute-list filters are applied to network numbers. See the “Controlling Advertising and Processing in Routing Updates” section for information about the **distribute-list** command.

You can use route maps on a per-neighbor basis to filter updates and to modify various attributes. A route map can be applied to either inbound or outbound updates. Only the routes that pass the route map are sent or accepted in updates. On both inbound and outbound updates, matching is supported based on AS path, community, and network numbers. Autonomous system path matching requires the **match as-path access-list** route-map command, community based matching requires the **match community-list** route-map command, and network-based matching requires the **ip access-list** global configuration command.

Prefix List for BGP Filtering

You can use prefix lists as an alternative to access lists in many BGP route filtering commands, including the **neighbor distribute-list** router configuration command. The advantages of using prefix lists include performance improvements in loading and lookup of large lists, incremental update support, easier CLI configuration, and greater flexibility.

Filtering by a prefix list involves matching the prefixes of routes with those listed in the prefix list, as when matching access lists. When there is a match, the route is used. Whether a prefix is permitted or denied is based upon these rules:

- An empty prefix list permits all prefixes.
- An implicit deny is assumed if a given prefix does not match any entries in a prefix list.
- When multiple entries of a prefix list match a given prefix, the sequence number of a prefix list entry identifies the entry with the lowest sequence number.

By default, sequence numbers are generated automatically and incremented in units of five. If you disable the automatic generation of sequence numbers, you must specify the sequence number for each entry. You can specify sequence values in any increment. If you specify increments of one, you cannot insert additional entries into the list; if you choose large increments, you might run out of values.

BGP Community Filtering

One way that BGP controls the distribution of routing information based on the value of the COMMUNITIES attribute. The attribute is a way to group destinations into communities and to apply routing decisions based on the communities. This method simplifies configuration of a BGP speaker to control distribution of routing information.

A community is a group of destinations that share some common attribute. Each destination can belong to multiple communities. AS administrators can define to which communities a destination belongs. By default, all destinations belong to the general Internet community. The community is identified by the COMMUNITIES attribute, an optional, transitive, global attribute in the numerical range from 1 to 4294967200. These are some predefined, well-known communities:

- **internet**—Advertise this route to the Internet community. All routers belong to it.
- **no-export**—Do not advertise this route to EBGp peers.
- **no-advertise**—Do not advertise this route to any peer (internal or external).
- **local-as**—Do not advertise this route to peers outside the local autonomous system.

Based on the community, you can control which routing information to accept, prefer, or distribute to other neighbors. A BGP speaker can set, append, or modify the community of a route when learning, advertising,

or redistributing routes. When routes are aggregated, the resulting aggregate has a COMMUNITIES attribute that contains all communities from all the initial routes.

You can use community lists to create groups of communities to use in a match clause of a route map. As with an access list, a series of community lists can be created. Statements are checked until a match is found. As soon as one statement is satisfied, the test is concluded.

BGP Neighbors and Peer Groups

Often many BGP neighbors are configured with the same update policies (that is, the same outbound route maps, distribute lists, filter lists, update source, and so on). Neighbors with the same update policies can be grouped into peer groups to simplify configuration and to make updating more efficient. When you have configured many peers, we recommend this approach.

To configure a BGP peer group, you create the peer group, assign options to the peer group, and add neighbors as peer group members. You configure the peer group by using the **neighbor** router configuration commands. By default, peer group members inherit all the configuration options of the peer group, including the remote-as (if configured), version, update-source, out-route-map, out-filter-list, out-dist-list, minimum-advertisement-interval, and next-hop-self. All peer group members also inherit changes that are made to the peer group. Members can also be configured to override the options that do not affect outbound updates.

Aggregate Routes

Classless interdomain routing (CIDR) enables you to create aggregate routes (or supernets) to minimize the size of routing tables. You can configure aggregate routes in BGP either by redistributing an aggregate route into BGP or by creating an aggregate entry in the BGP routing table. An aggregate address is added to the BGP table when there is at least one more specific entry in the BGP table.

Routing Domain Confederations

One way to reduce the IBGP mesh is to divide an autonomous system into multiple subautonomous systems and to group them into a single confederation that appears as a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Even though the peers in different autonomous systems have EBGP sessions, they exchange routing information as if they were IBGP peers. Specifically, the next hop, MED, and local preference information are preserved. You can then use a single IGP for all of the autonomous systems.

BGP Route Reflectors

BGP requires that all of the IBGP speakers be fully meshed. When a router receives a route from an external neighbor, it must advertise it to all internal neighbors. To prevent a routing information loop, all IBGP speakers must be connected. The internal neighbors do not send routes that are learned from internal neighbors to other internal neighbors.

With route reflectors, all IBGP speakers need not be fully meshed because another method is used to pass learned routes to neighbors. When you configure an internal BGP peer to be a route reflector, it is responsible for passing IBGP learned routes to a set of IBGP neighbors. The internal peers of the route reflector are divided into two groups: client peers and nonclient peers (all the other routers in the autonomous system). A route reflector reflects routes between these two groups. The route reflector and its client peers form a cluster. The

nonclient peers must be fully meshed with each other, but the client peers need not be fully meshed. The clients in the cluster do not communicate with IBGP speakers outside their cluster.

When the route reflector receives an advertised route, it takes one of these actions, depending on the neighbor:

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

Usually a cluster of clients has a single route reflector, and the cluster is identified by the route reflector router ID. To increase redundancy and to avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the same 4-byte cluster ID so that a route reflector can recognize updates from route reflectors in the same cluster. All the route reflectors serving a cluster should be fully meshed and should have identical sets of client and nonclient peers.

Route Dampening

Route flap dampening is a BGP feature designed to minimize the propagation of flapping routes across an internetwork. A route is considered to be flapping when it is repeatedly available, then unavailable, then available, then unavailable, and so on. When route dampening is enabled, a numeric penalty value is assigned to a route when it flaps. When a route's accumulated penalties reach a configurable limit, BGP suppresses advertisements of the route, even if the route is running. The reuse limit is a configurable value that is compared with the penalty. If the penalty is less than the reuse limit, a suppressed route that is up is advertised again.

Dampening is not applied to routes that are learned by IBGP. This policy prevents the IBGP peers from having a higher penalty for routes external to the AS.

Conditional BGP Route Injection

Routes that are advertised through the BGP are commonly aggregated to minimize the number of routes that are used and reduce the size of global routing tables. However, common route aggregation can obscure more specific routing information that is more accurate but not necessary to forward packets to their destinations. Routing accuracy is obscured by common route aggregation because a prefix that represents multiple addresses or hosts over a large topological area cannot be accurately reflected in a single route. Cisco software provides several methods by which you can originate a prefix into BGP. Prior to the BGP conditional route injection feature, the existing methods included redistribution and using the **network** or **aggregate-address** command. However, these methods assume the existence of more specific routing information (matching the route to be originated) in either the routing table or the BGP table.

BGP conditional route injection allows you to originate a prefix into a BGP routing table without the corresponding match. This feature allows more specific routes to be generated based on administrative policy or traffic engineering information in order to provide more specific control over the forwarding of packets to these more specific routes, which are injected into the BGP routing table only if the configured conditions are met. Enabling this feature will allow you to improve the accuracy of common route aggregation by conditionally injecting or replacing less specific prefixes with more specific prefixes. Only prefixes that are equal to or more specific than the original prefix may be injected. BGP conditional route injection is enabled with the **bgp inject-map exist-map** command and uses two route maps (inject map and exist map) to install one (or more) more specific prefixes into a BGP routing table. The exist map specifies the prefixes that the BGP speaker will track. The inject map defines the prefixes that will be created and installed into the local BGP table.



Note Inject maps and exist maps will only match a single prefix per route map clause. To inject additional prefixes, you must configure additional route map clauses. If multiple prefixes are used, the first prefix that is matched will be used.

BGP Peer Templates

To address some of the limitations of peer groups such as configuration management, BGP peer templates were introduced to support the BGP update group configuration.

A peer template is a configuration pattern that can be applied to neighbors that share policies. Peer templates are reusable and support inheritance, which allows the network operator to group and apply distinct neighbor configurations for BGP neighbors that share policies. Peer templates also allow the network operator to define complex configuration patterns through the capability of a peer template to inherit a configuration from another peer template.

There are two types of peer templates:

- Peer session templates are used to group and apply the configuration of general session commands that are common to all address family and NLRI configuration modes.
- Peer policy templates are used to group and apply the configuration of commands that are applied within specific address families and NLRI configuration modes.

Peer templates improve the flexibility and enhance the capability of neighbor configuration. Peer templates also provide an alternative to peer group configuration and overcome some limitations of peer groups. BGP peer devices using peer templates also benefit from automatic update group configuration. With the configuration of the BGP peer templates and the support of the BGP dynamic update peer groups, the network operator no longer must configure peer groups in BGP and the network can benefit from improved configuration flexibility and faster convergence.



Note 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 from peer templates.

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.

Inheritance in Peer Templates

The inheritance capability is a key component of peer template operation. Inheritance in a peer template is similar to node and tree structures that are commonly found in general computing, for example, file and directory trees. A peer template can directly or indirectly inherit the configuration from another peer template. The directly inherited peer template represents the tree in the structure. The indirectly inherited peer template represents a node in the tree. Because each node also supports inheritance, branches can be created that apply

the configurations of all indirectly inherited peer templates within a chain back to the directly inherited peer template or the source of the tree.

This structure eliminates the need to repeat configuration statements that are commonly reapplied to groups of neighbors because common configuration statements can be applied once and then indirectly inherited by peer templates that are applied to neighbor groups with common configurations. Configuration statements that are duplicated separately within a node and a tree are filtered out at the source of the tree by the directly inherited template. A directly inherited template overwrites any indirectly inherited statements that are duplicated in the directly inherited template.

Inheritance expands the scalability and flexibility of neighbor configuration by allowing you to chain together peer templates configurations to create simple configurations that inherit common configuration statements or complex configurations that apply specific configuration statements along with common inherited configurations. Specific details about configuring inheritance in peer session templates and peer policy templates are provided in the following sections.

When BGP neighbors use inherited peer templates, it can be difficult to determine which policies are associated with a specific template. The **detail** keyword of the **show ip bgp template peer-policy** command displays the detailed configuration of local and inherited policies that are associated with a specific template.

Peer Session Templates

Peer session templates are used to group and apply the configuration of general session commands to groups of neighbors that share session configuration elements. General session commands that are common for neighbors that are configured in different address families can be configured within the same peer session template. Peer session templates are created and configured in peer session configuration mode. Only general session commands can be configured in a peer session template. The following general session commands are supported by peer session templates:

- **description**
- **disable-connected-check**
- **ebgp-multihop**
- **exit peer-session**
- **inherit peer-session**
- **local-as**
- **password**
- **remote-as**
- **shutdown**
- **timers**
- **translate-update**
- **update-source**
- **version**

General session commands can be configured once in a peer session template and then applied to many neighbors through the direct application of a peer session template or through indirect inheritance from a peer

session template. The configuration of peer session templates simplifies the configuration of general session commands that are commonly applied to all neighbors within an autonomous system.

Peer session templates support direct and indirect inheritance. A peer can be configured with only one peer session template at a time, and that peer session template can contain only one indirectly inherited peer session template.



Note If you attempt to configure more than one inherit statement with a single peer session template, an error message will be displayed.

This behavior allows a BGP neighbor to directly inherit only one session template and indirectly inherit up to seven additional peer session templates. This allows you to apply up to a maximum of eight peer session configurations to a neighbor: the configuration from the directly inherited peer session template and the configurations from up to seven indirectly inherited peer session templates. Inherited peer session configurations are evaluated first and applied starting with the last node in the branch and ending with the directly applied peer session template configuration at the source of the tree. The directly applied peer session template will have priority over inherited peer session template configurations. Any configuration statements that are duplicated in inherited peer session templates will be overwritten by the directly applied peer session template. So, if a general session command is reapplied with a different value, the subsequent value will have priority and overwrite the previous value that was configured in the indirectly inherited template. The following examples illustrate the use of this feature.

In the following example, the general session command **remote-as 1** is applied in the peer session template named SESSION-TEMPLATE-ONE:

```
template peer-session SESSION-TEMPLATE-ONE
  remote-as 1
  exit peer-session
```

Peer session templates support only general session commands. BGP policy configuration commands that are configured only for a specific address family or NLRI configuration mode are configured with peer policy templates.

Peer Policy Templates

Peer policy templates are used to group and apply the configuration of commands that are applied within specific address families and NLRI configuration mode. Peer policy templates are created and configured in peer policy configuration mode. BGP policy commands that are configured for specific address families are configured in a peer policy template. The following BGP policy commands are supported by peer policy templates:

- **advertisement-interval**
- **allowas-in**
- **as-override**
- **capability**
- **default-originate**
- **distribute-list**

- **dmzlink-bw**
- **exit-peer-policy**
- **filter-list**
- **inherit peer-policy**
- **maximum-prefix**
- **next-hop-self**
- **next-hop-unchanged**
- **prefix-list**
- **remove-private-as**
- **route-map**
- **route-reflector-client**
- **send-community**
- **send-label**
- **soft-reconfiguration**
- **unsuppress-map**
- **weight**

Peer policy templates are used to configure BGP policy commands that are configured for neighbors that belong to specific address families. Like peer session templates, peer policy templates are configured once and then applied to many neighbors through the direct application of a peer policy template or through inheritance from peer policy templates. The configuration of peer policy templates simplifies the configuration of BGP policy commands that are applied to all neighbors within an autonomous system.

Like a peer session template, a peer policy template supports inheritance. However, there are minor differences. A directly applied peer policy template can directly or indirectly inherit configurations from up to seven peer policy templates. So, a total of eight peer policy templates can be applied to a neighbor or neighbor group. Like route maps, inherited peer policy templates are configured with sequence numbers. Also like a route map, an inherited peer policy template is evaluated starting with the **inherit peer-policy** statement with the lowest sequence number and ending with the highest sequence number. However, there is a difference; a peer policy template will not collapse like a route map. Every sequence is evaluated, and if a BGP policy command is reapplied with a different value, it will overwrite any previous value from a lower sequence number.

The directly applied peer policy template and the **inherit peer-policy** statement with the highest sequence number will always have priority and be applied last. Commands that are reapplied in subsequent peer templates will always overwrite the previous values. This behavior is designed to allow you to apply common policy configurations to large neighbor groups and specific policy configurations only to certain neighbors and neighbor groups without duplicating individual policy configuration commands.

Peer policy templates support only policy configuration commands. BGP policy configuration commands that are configured only for specific address families are configured with peer policy templates.

The configuration of peer policy templates simplifies and improves the flexibility of BGP configuration. A specific policy can be configured once and referenced many times. Because a peer policy supports up to eight levels of inheritance, very specific and very complex BGP policies can also be created.

BGP Route Map Next Hop Self

The BGP Route Map Next Hop Self feature provides a way to override the settings for `bgp next-hop unchanged` and `bgp next-hop unchanged allpath` selectively. These settings are global for an address family. For some routes this may not be appropriate. For example, static routes may need to be redistributed with a next hop of self, but connected routes and routes learned via Interior Border Gateway Protocol (IBGP) or Exterior Border Gateway Protocol (EBGP) may continue to be redistributed with an unchanged next hop.

The BGP route map next hop self functionality modifies the existing route map infrastructure to configure a new `ip next-hop self` setting, which overrides the `bgp next-hop unchanged` and `bgp next-hop unchanged allpaths` settings.

The `ip next-hop self` setting is applicable only to VPNv4 and VPNv6 address families. Routes distributed by protocols other than BGP are not affected.

You configure a new `bgp route-map priority` setting to inform BGP that the route map will take priority over the settings for `bgp next-hop unchanged` and `bgp next-hop unchanged allpath`. The `bgp route-map priority` setting only impacts BGP. The `bgp route-map priority` setting has no impact unless you configure the `bgp next-hop unchanged` or `bgp next-hop unchanged allpaths` settings.

How to Configure BGP

The following sections provide configurational information about BGP.

Default BGP Configuration

The table given below shows the basic default BGP configuration.

Table 2: Default BGP Configuration

| Feature | Default Setting |
|------------------------------------|---|
| Aggregate address | Disabled: None defined. |
| AS path access list | None defined. |
| Auto summary | Disabled. |
| Best path | <ul style="list-style-type: none"> The router considers <i>as-path</i> in choosing a route and does not compare s from external BGP peers. Compare router ID: Disabled. |
| BGP community list | <ul style="list-style-type: none"> Number: None defined. When you permit a value for the community number defaults to an implicit deny for everything else that has not been permitted. Format: Cisco default format (32-bit number). |
| BGP confederation identifier/peers | <ul style="list-style-type: none"> Identifier: None configured. Peers: None identified. |

| Feature | Default Setting |
|--|---|
| BGP Fast external fallover | Enabled. |
| BGP local preference | 100. The range is 0 to 4294967295 with the higher value preferred. |
| BGP network | None specified; no backdoor route advertised. |
| BGP route dampening | Disabled by default. When enabled: <ul style="list-style-type: none"> • 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 for a physical interface on the router. |
| Default information originate (protocol or network redistribution) | Disabled. |
| Default metric | Built-in, automatic metric translations. |
| Distance | <ul style="list-style-type: none"> • 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 | <ul style="list-style-type: none"> • 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) | <ul style="list-style-type: none"> • Always compare: Disabled. Does not compare MEDs for paths from neighboring different autonomous systems. • Best path compare: Disabled. • MED missing as worst path: Disabled. • Deterministic MED comparison is disabled. |

| Feature | Default Setting |
|-------------------------------|---|
| Neighbor | <ul style="list-style-type: none"> • 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. • 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: 1. |
| NSF ¹ Awareness | Disabled ² . If enabled, allows Layer 3 switches to continue forwarding packets to neighboring NSF-capable 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. |

¹ Nonstop Forwarding

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

Enabling BGP Routing

To enable BGP routing, perform the following procedure:

Procedure

| | Command or Action | Purpose |
|---------------|---|--|
| 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 routing Example: Device (config)# ip routing | Enables IP routing. |
| Step 4 | router bgp <i>autonomous-system</i> Example: Device (config)# router bgp 45000 | Enables a BGP routing process, assign it an AS number, and enter router configuration mode. The AS number can be from 1 to 65535, with 64512 to 65535 designated as private autonomous numbers. |
| Step 5 | network <i>network-number</i> [mask <i>network-mask</i>] [route-map <i>route-map-name</i>] Example: Device (config-router)# network 10.108.0.0 | Configures a network as local to this AS, and enter it in the BGP table. |
| Step 6 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } remote-as <i>number</i> Example: Device (config-router)# neighbor 10.108.1.2 remote-as 65200 | Adds an entry to the BGP neighbor table specifying that the neighbor that is identified by the IP address belongs to the specified AS. For EBGP, neighbors are usually directly connected, and the IP address is the address of the interface at the other end of the connection. For IBGP, the IP address can be the address of any of the router interfaces. |
| Step 7 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } remove-private-as Example: | (Optional) Removes private AS numbers from the AS-path in outbound routing updates. |

| | Command or Action | Purpose |
|----------------|---|---|
| | Device (config-router) # neighbor 172.16.2.33 remove-private-as | |
| Step 8 | synchronization Example: Device (config-router) # synchronization | (Optional) Enables synchronization between BGP and an IGP. |
| Step 9 | auto-summary Example: Device (config-router) # auto-summary | (Optional) Enables automatic network summarization. When a subnet is redistributed from an IGP into BGP, only the network route is inserted into the BGP table. |
| Step 10 | bgp graceful-restart Example: Device (config-router) # bgp graceful-start | (Optional) Enables NSF awareness on switch. By default, NSF awareness is disabled. |
| Step 11 | end Example: Device (config-router) # end | Returns to privileged EXEC mode. |
| Step 12 | show ip bgp network network-number Example: Device# show ip bgp network 10.108.0.0 | Verifies the configuration. |
| Step 13 | show ip bgp neighbor Example: Device# show ip bgp neighbor | Verifies that NSF awareness (Graceful Restart) is enabled on the neighbor. If NSF awareness is enabled on the switch and the neighbor, this message appears: <i>Graceful Restart Capability: advertised and received</i> If NSF awareness is enabled on the switch, but not on the neighbor, this message appears: <i>Graceful Restart Capability: advertised.</i> |
| Step 14 | copy running-config startup-config Example: Device# copy running-config startup-config | (Optional) Saves your entries in the configuration file. |

Managing Routing Policy Changes

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

Procedure

| | Command or Action | Purpose |
|---------------|--|--|
| Step 1 | show ip bgp neighbors Example: Device# <code>show ip bgp neighbors</code> | Displays whether a neighbor supports the route refresh capability. When supported, this message appears for the router: <i>Received route refresh capability from peer.</i> |
| Step 2 | clear ip bgp {* <i>address</i> <i>peer-group-name</i> } Example: Device# <code>clear ip bgp *</code> | Resets the routing table on the specified connection. <ul style="list-style-type: none"> • 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 {* <i>address</i> <i>peer-group-name</i> } soft out Example: Device# <code>clear ip bgp * soft out</code> | (Optional) Performs an outbound soft reset to reset the inbound routing table on the specified connection. Use this command if route refresh is supported. <ul style="list-style-type: none"> • 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# <code>show ip bgp</code> | Verifies the reset by checking information about the routing table and about BGP neighbors. |
| Step 5 | show ip bgp neighbors Example: Device# <code>show ip bgp neighbors</code> | Verifies the reset by checking information about the routing table and about BGP neighbors. |

Configuring BGP Decision Attributes

To configure BGP decision attributes, perform this procedure:

Before you begin

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

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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 <i>autonomous-system</i> 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 <i>{ip-address peer-group-name}</i> 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 <i>{ip-address peer-group-name}</i> weight <i>weight</i> 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. |

| | Command or Action | Purpose |
|---------|--|--|
| Step 7 | default-metric <i>number</i> Example: <pre>Device(config-router)# default-metric 300</pre> | (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: <pre>Device(config-router)# bgp bestpath med missing-as-worst</pre> | (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: <pre>Device(config-router)# bgp always-compare-med</pre> | (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: <pre>Device(config-router)# bgp bestpath med confed</pre> | (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: <pre>Device(config-router)# bgp deterministic med</pre> | (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 <i>value</i> Example: <pre>Device(config-router)# bgp default local-preference 200</pre> | (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 <i>number</i> Example: <pre>Device(config-router)# maximum-paths 8</pre> or <pre>Device(config-router)# maximum-paths ibgp 2</pre> | (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. <ul style="list-style-type: none"> • <i>number</i>: The number of paths to be added to the IP routing table. <p>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.</p> |

| | Command or Action | Purpose |
|----------------|---|---|
| | | <ul style="list-style-type: none"> • ibgp number: The number of iBGP paths to be added to the IP routing table. <p>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.</p> |
| Step 14 | end Example: <pre>Device(config-router)# end</pre> | Returns to privileged EXEC mode. |
| Step 15 | show ip bgp Example: <pre>Device# show ip bgp</pre> | Verifies the reset by checking information about the routing table and about BGP neighbors. |
| Step 16 | show ip bgp neighbors Example: <pre>Device# show ip bgp neighbors</pre> | Verifies the reset by checking information about the routing table and about BGP neighbors. |
| Step 17 | copy running-config startup-config Example: <pre>Device# copy running-config startup-config</pre> | (Optional) Saves your entries in the configuration file. |

Configuring BGP Filtering with Route Maps

To configure BGP filtering with route maps, perform the following procedure:

Procedure

| | Command or Action | Purpose |
|---------------|--|---|
| Step 1 | enable Example: <pre>Device> enable</pre> | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|---------------|---|---|
| Step 2 | configure terminal Example: Device# <code>configure terminal</code> | Enters global configuration mode. |
| Step 3 | route-map <i>map-tag</i> [permit deny] [<i>sequence-number</i>] Example: Device(config)# <code>route-map</code> <code>set-peer-address permit 10</code> | Creates a route map, and enter route-map configuration mode. |
| Step 4 | set ip next-hop <i>ip-address</i> [... <i>ip-address</i>] [<i>peer-address</i>] Example: Device(config)# <code>set ip next-hop 10.1.1.3</code> | (Optional) Sets a route map to disable next-hop processing <ul style="list-style-type: none"> • 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)# <code>end</code> | Returns to privileged EXEC mode. |
| Step 6 | show route-map [<i>map-name</i>] Example: Device# <code>show route-map</code> | Displays all route maps configured or only the one specified to verify configuration. |
| Step 7 | copy running-config startup-config Example: Device# <code>copy running-config</code> <code>startup-config</code> | (Optional) Saves your entries in the configuration file. |

Configuring BGP Filtering by Neighbor

To configure BGP filter by neighbor, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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 <i>autonomous-system</i> Example: Device(config)# router bgp 109 | Enables a BGP routing process, assign it an AS number, and enter router configuration mode. |
| Step 4 | neighbor { <i>ip-address</i> <i>peer-group name</i> } distribute-list { <i>access-list-number</i> <i>name</i> } { <i>in</i> <i>out</i> } Example: Device(config-router)# neighbor 172.16.4.1 distribute-list 39 in | (Optional) Filters BGP routing updates to or from neighbors as specified in an access list. Note You can also use the neighbor prefix-list router configuration command to filter updates, but you cannot use both commands to configure the same BGP peer. |
| Step 5 | neighbor { <i>ip-address</i> <i>peer-group name</i> } route-map <i>map-tag</i> { <i>in</i> <i>out</i> } Example: Device(config-router)# neighbor 172.16.70.24 route-map internal-map in | (Optional) Applies a route map to filter an incoming or outgoing route. |
| Step 6 | end Example: Device(config)# end | Returns to privileged EXEC mode. |
| Step 7 | show ip bgp neighbors Example: Device# show ip bgp neighbors | Verifies the configuration. |
| Step 8 | copy running-config startup-config Example: | (Optional) Saves your entries in the configuration file. |

| | Command or Action | Purpose |
|--|---|---------|
| | Device# <code>copy running-config startup-config</code> | |

Configuring BGP Filtering by Access Lists and Neighbors

Another method of filtering is to specify an access list filter on both incoming and outbound updates, based on the BGP autonomous system paths. Each filter is an access list based on regular expressions. To use this method, define an autonomous system path access list, and apply it to updates to and from particular neighbors.

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| Step 1 | enable Example: Device> <code>enable</code> | Enables privileged EXEC mode. Enter your password if prompted. |
| Step 2 | configure terminal Example: Device# <code>configure terminal</code> | Enters global configuration mode. |
| Step 3 | ip as-path access-list <i>access-list-number</i> {permit deny} <i>as-regular-expressions</i> Example: Device(config)# <code>ip as-path access-list 1 deny _65535_</code> | Defines a BGP-related access list. |
| Step 4 | router bgp <i>autonomous-system</i> Example: Device(config)# <code>router bgp 110</code> | Enters BGP router configuration mode. |
| Step 5 | neighbor <i>{ip-address peer-group name}</i> filter-list <i>{access-list-number name}</i> {in out weight weight} Example: Device(config-router)# <code>neighbor 172.16.1.1 filter-list 1 out</code> | Establishes a BGP filter based on an access list. |
| Step 6 | end Example: | Returns to privileged EXEC mode. |

| | Command or Action | Purpose |
|---------------|--|--|
| | Device(config)# end | |
| 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 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

| | Command or Action | Purpose |
|---------------|---|---|
| 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 le-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. • <i>network/len</i> is the network number and length (in bits) of the network mask. |

| | Command or Action | Purpose |
|---------------|--|---|
| | | <ul style="list-style-type: none"> (Optional) ge and le values specify the range of the prefix length to be matched. The specified <i>ge-value</i> and <i>le-value</i> must satisfy this condition: $len < ge-value < le-value < 32$ |
| Step 4 | ip prefix-list <i>list-name</i> seq <i>seq-value</i> deny permit <i>network/len</i> [ge <i>ge-value</i>] [le <i>le-value</i>] Example: <pre>Device(config)# ip prefix-list BLUE seq 10 permit 172.24.1.0/24</pre> | (Optional) Adds an entry to a prefix list, and assign a sequence number to the entry. |
| Step 5 | end Example: <pre>Device(config)# end</pre> | Returns to privileged EXEC mode. |
| Step 6 | show ip prefix list [detail summary] <i>name</i> [<i>network/len</i>] [seq <i>seq-num</i>] [longer] [first-match] Example: <pre>Device# show ip prefix list summary test</pre> | Verifies the configuration by displaying information about a prefix list or prefix list entries. |
| Step 7 | copy running-config startup-config Example: <pre>Device# copy running-config startup-config</pre> | (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:

Procedure

| | Command or Action | Purpose |
|---------------|----------------------------------|---|
| Step 1 | enable Example: | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|---------------|--|---|
| | Device> enable | |
| Step 2 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 3 | ip community-list <i>community-list-number</i> { permit deny } <i>community-number</i> Example: Device (config)# ip community-list 1 permit 50000:10 | Creates a community list, and assigns it a number. <ul style="list-style-type: none"> • The <i>community-list-number</i> is an integer from 1 to 99 that identifies one or more permit or deny groups of communities. • The <i>community-number</i> is the number that is configured by a set community route-map configuration command. |
| Step 4 | router bgp <i>autonomous-system</i> Example: Device (config)# router bgp 108 | Enters BGP router configuration mode. |
| Step 5 | neighbor { <i>ip-address</i> <i>peer-group name</i> } 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 <i>list-num</i> 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 |

| | Command or Action | Purpose |
|----------------|---|--|
| | | 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:

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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 <i>autonomous-system</i> | Enters BGP router configuration mode. |
| Step 4 | neighbor <i>peer-group-name</i> peer-group | Creates a BGP peer group. |

| | Command or Action | Purpose |
|---------|---|---|
| Step 5 | neighbor <i>ip-address</i> peer-group <i>peer-group-name</i> | Makes a BGP neighbor a member of the peer group. |
| Step 6 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } remote-as <i>number</i> | Specifies a BGP neighbor. If a peer group is not configured with a remote-as <i>number</i> , use this command to create peer groups containing EBGP neighbors. The range is 1 to 65535. |
| Step 7 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } description <i>text</i> | (Optional) Associates a description with a neighbor. |
| Step 8 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } default-originate [route-map <i>map-name</i>] | (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 { <i>ip-address</i> <i>peer-group-name</i> } send-community | (Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address. |
| Step 10 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } update-source <i>interface</i> | (Optional) Allows internal BGP sessions to use any operational interface for TCP connections. |
| Step 11 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } ebgp-multihop | (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 { <i>ip-address</i> <i>peer-group-name</i> } local-as <i>number</i> | (Optional) Specifies an AS number to use as the local AS. The range is 1 to 65535. |
| Step 13 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } advertisement-interval <i>seconds</i> | (Optional) Sets the minimum interval between sending BGP routing updates. |
| Step 14 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } maximum-prefix <i>maximum</i> [<i>threshold</i>] | (Optional) Controls how many prefixes can be received from a neighbor. The range is 1 to 4294967295. The <i>threshold</i> (optional) is the percentage of maximum at which a warning message is generated. The default is 75 percent. |
| Step 15 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } next-hop-self | (Optional) Disables next-hop processing on the BGP updates to a neighbor. |
| Step 16 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } password {0-7} <i>string</i> | (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: |

| | Command or Action | Purpose |
|---------|--|---|
| | | <ul style="list-style-type: none"> • 0 - no encryption/plaintext • 7 - proprietary encryption type <p>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.</p> <ul style="list-style-type: none"> • <i>string</i> - the password string |
| Step 17 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } route-map <i>map-name</i> { in out } | (Optional) Applies a route map to incoming or outgoing routes. |
| Step 18 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } send-community | (Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address. |
| Step 19 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } timers <i>keepalive holdtime</i> | <p>(Optional) Sets timers for the neighbor or peer group.</p> <ul style="list-style-type: none"> • The <i>keepalive</i> interval is the time within which keepalive messages are sent to peers. The range is 1 to 4294967295 seconds; the default is 60. • The <i>holdtime</i> 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 { <i>ip-address</i> <i>peer-group-name</i> } weight <i>weight</i> | (Optional) Specifies a weight for all routes from a neighbor. |
| Step 21 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } distribute-list { <i>access-list-number</i> <i>name</i> } { in out } | (Optional) Filter BGP routing updates to or from neighbors, as specified in an access list. |
| Step 22 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } filter-list <i>access-list-number</i> { in out weight <i>weight</i> } | (Optional) Establish a BGP filter. |
| Step 23 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } version <i>value</i> | (Optional) Specifies the BGP version to use when communicating with a neighbor. |
| Step 24 | neighbor { <i>ip-address</i> <i>peer-group-name</i> } soft-reconfiguration inbound | (Optional) Configures the software to start storing received updates. |
| Step 25 | end Example: | Returns to privileged EXEC mode. |

| | Command or Action | Purpose |
|----------------|---|--|
| | Device (config) # end | |
| Step 26 | show ip bgp neighbors | 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

To configure aggregate addresses in a routing table, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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 <i>autonomous-system</i> Example: Device (config) # router bgp 106 | Enters BGP router configuration mode. |
| Step 4 | aggregate-address <i>address mask</i> 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 <i>address mask as-set</i> 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 |

| | Command or Action | Purpose |
|----------------|--|--|
| | | aggregating many paths because this route must be continually withdrawn and updated. |
| Step 6 | aggregate-address <i>address-mask</i> summary-only Example: <pre>Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 summary-only</pre> | (Optional) Advertises summary addresses only. |
| Step 7 | aggregate-address <i>address mask</i> suppress-map <i>map-name</i> Example: <pre>Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 suppress-map map1</pre> | (Optional) Suppresses selected, more specific routes. |
| Step 8 | aggregate-address <i>address mask</i> advertise-map <i>map-name</i> Example: <pre>Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 advertise-map map2</pre> | (Optional) Generates an aggregate based on conditions that are specified by the route map. |
| Step 9 | aggregate-address <i>address mask</i> attribute-map <i>map-name</i> Example: <pre>Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 attribute-map map3</pre> | (Optional) Generates an aggregate with attributes that are specified in the route map. |
| Step 10 | end Example: <pre>Device(config)# end</pre> | Returns to privileged EXEC mode. |
| Step 11 | show ip bgp neighbors [<i>advertised-routes</i>] Example: <pre>Device# show ip bgp neighbors</pre> | Verifies the configuration. |
| Step 12 | copy running-config startup-config Example: <pre>Device# copy running-config startup-config</pre> | (Optional) Saves your entries in the configuration file. |

Configuring Routing Domain Confederations

You must specify a confederation identifier that acts as the autonomous system number for the group of autonomous systems.

Procedure

| | Command or Action | Purpose |
|---------------|--|---|
| 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 <i>autonomous-system</i> Example: Device(config)# router bgp 100 | Enters BGP router configuration mode. |
| Step 4 | bgp confederation identifier <i>autonomous-system</i> Example: Device(config)# bgp confederation identifier 50007 | Configures a BGP confederation identifier. |
| Step 5 | bgp confederation peers <i>autonomous-system</i> <i>[autonomous-system ...]</i> 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. |

| | Command or Action | Purpose |
|---------------|---|--|
| Step 8 | show ip bgp network Example: Device# <code>show ip bgp network</code> | Verifies the configuration. |
| Step 9 | copy running-config startup-config Example: Device# <code>copy running-config startup-config</code> | (Optional) Saves your entries in the configuration file. |

Configuring BGP Route Reflectors

To configure BGP route reflectors, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|--|--|
| Step 1 | enable Example: Device> <code>enable</code> | Enables privileged EXEC mode. Enter your password if prompted. |
| Step 2 | configure terminal Example: Device# <code>configure terminal</code> | Enters global configuration mode. |
| Step 3 | router bgp <i>autonomous-system</i> Example: Device(config)# <code>router bgp 101</code> | Enters BGP router configuration mode. |
| Step 4 | neighbor {<i>ip-address</i> <i>peer-group-name</i>} route-reflector-client Example: Device(config-router)# <code>neighbor 172.16.70.24 route-reflector-client</code> | Configures the local router as a BGP route reflector and the specified neighbor as a client. |
| Step 5 | bgp cluster-id <i>cluster-id</i> Example: | (Optional) Configures the cluster ID if the cluster has more than one route reflector. |

| | Command or Action | Purpose |
|---------------|--|---|
| | Device(config-router)# bgp cluster-id 10.0.1.2 | |
| 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. |

Configuring Route Dampening

To configure route dampening, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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. |

| | Command or Action | Purpose |
|----------------|---|--|
| Step 3 | router bgp <i>autonomous-system</i> 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 <i>half-life reuse suppress max-suppress [route-map map]</i> 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 [{ regex <i>regex</i> } { filter-list <i>list</i> } { <i>address mask</i> [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 [{ regex <i>regex</i> } { filter-list <i>list</i> } { <i>address mask</i> [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. |

| | Command or Action | Purpose |
|----------------|---|--|
| Step 11 | copy running-config startup-config Example: <pre>Device# copy running-config startup-config</pre> | (Optional) Saves your entries in the configuration file. |

Conditionally Injecting BGP Routes

Use this task to inject more specific prefixes into a BGP routing table over less specific prefixes that were selected through normal route aggregation. These more specific prefixes can be used to provide a finer granularity of traffic engineering or administrative control than is possible with aggregated routes.

To conditionally injecting BGP routes, perform this procedure:

Before you begin

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

Procedure

| | Command or Action | Purpose |
|---------------|--|---|
| Step 1 | enable Example: <pre>Device> enable</pre> | Enables privileged EXEC mode. Enter your password if prompted. |
| Step 2 | configure terminal Example: <pre>Device# configure terminal</pre> | Enters global configuration mode. |
| Step 3 | router bgp <i>autonomous-system-number</i> Example: <pre>Device(config)# router bgp 40000</pre> | Enters router configuration mode for the specified routing process. |
| Step 4 | bgp inject-map <i>inject-map-name</i> exist-map <i>exist-map-name</i> [copy-attributes] Example: <pre>Device(config-router)# bgp inject-map ORIGINATE exist-map LEARNED_PATH</pre> | Specifies the inject map and the exist map for conditional route injection. <ul style="list-style-type: none"> Use the copy-attributes keyword to specify that the injected route inherits the attributes of the aggregate route. |
| Step 5 | exit Example: | Exits router configuration mode and enters global configuration mode. |

| | Command or Action | Purpose |
|----------------|--|--|
| | Device (config-router) # exit | |
| Step 6 | route-map <i>map-tag</i> [permit deny] <i>[sequence-number]</i> Example: Device (config) # route-map LEARNED_PATH permit 10 | Configures a route map and enters route map configuration mode. |
| Step 7 | match ip address { <i>access-list-number</i> <i>[access-list-number... access-list-name...]</i> <i>access-list-name [access-list-number... </i> <i>access-list-name]</i> prefix-list <i>prefix-list-name</i> <i>[prefix-list-name...]</i> } Example: Device (config-route-map) # match ip address prefix-list SOURCE | Specifies the aggregate route to which a more specific route will be injected. <ul style="list-style-type: none"> 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 { <i>access-list-number</i> <i>access-list-name</i> } [<i>access-list-number...</i> <i>access-list-name...</i>] Example: Device (config-route-map) # match ip route-source prefix-list ROUTE_SOURCE | Specifies the match conditions for redistributing the source of the route. <ul style="list-style-type: none"> In this example, the prefix list that is named ROUTE_SOURCE is used to redistribute the source of the route. <p>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.</p> |
| Step 9 | exit Example: Device (config-route-map) # exit | Exits route map configuration mode and enters global configuration mode. |
| Step 10 | route-map <i>map-tag</i> [permit deny] <i>[sequence-number]</i> Example: Device (config) # route-map ORIGINATE permit 10 | Configures a route map and enters route map configuration mode. |
| Step 11 | set ip address { <i>access-list-number</i> <i>[access-list-number... access-list-name...]</i> <i>access-list-name [access-list-number... </i> | 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. |

| | Command or Action | Purpose |
|----------------|---|---|
| | <code>access-list-name] prefix-list prefix-list-name [prefix-list-name...]}</code> Example: <pre>Device(config-route-map)# set ip address prefix-list ORIGINATED_ROUTES</pre> | |
| Step 12 | set community {community-number [additive] [well-known-community] none} Example: <pre>Device(config-route-map)# set community 14616:555 additive</pre> | Sets the BGP community attribute of the injected route. |
| Step 13 | exit Example: <pre>Device(config-route-map)# exit</pre> | Exits route map 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: <pre>Device(config)# ip prefix-list SOURCE permit 10.1.1.0/24</pre> | 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 | Repeat Step 14 for every prefix list to be created. | -- |
| Step 16 | exit Example: <pre>Device(config)# exit</pre> | Exits global configuration mode and returns to privileged EXEC mode. |
| Step 17 | show ip bgp injected-paths Example: <pre>Device# show ip bgp injected-paths</pre> | (Optional) Displays information about injected paths. |

Configuring Peer Session Templates

Use the following tasks to create and configure a peer session template:

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.



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



Note 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

| | Command or Action | Purpose |
|---------------|--|---|
| 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 <i>autonomous-system-number</i> Example: Device(config)# router bgp 101 | Enters router configuration mode and creates a BGP routing process. |
| Step 4 | template peer-session <i>session-template-name</i> Example: Device(config-router)# template peer-session INTERNAL-BGP | Enters session-template configuration mode and creates a peer session template. |
| Step 5 | remote-as <i>autonomous-system-number</i> Example: Device(config-router-stmp)# remote-as 202 | (Optional) Configures peering with a remote neighbor in the specified autonomous system. Note |

| | Command or Action | Purpose |
|---------------|--|--|
| | | Any supported general session command can be used here. For a list of the supported commands, see the “Restrictions” section. |
| Step 6 | timers <i>keepalive-interval hold-time</i> 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 peer-session [<i>session-template-name</i>] 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 <i>session-template-name</i> argument. This command also supports all standard output modifiers. |

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

This task configures peer session template inheritance with the **inherit peer-session** command. It creates and configures a peer session template and allows it to inherit a configuration from another peer session template.



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

| | Command or Action | Purpose |
|---------------|---|---|
| Step 1 | enable Example: Device> enable | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|---------------|---|--|
| Step 2 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 3 | router bgp <i>autonomous-system-number</i> Example: Device(config)# router bgp 101 | Enters router configuration mode and creates a BGP routing process. |
| Step 4 | template peer-session <i>session-template-name</i> Example: Device(config-router)# template peer-session CORE1 | Enter session-template configuration mode and creates a peer session template. |
| Step 5 | description <i>text-string</i> 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 6 | update-source <i>interface-type interface-number</i> 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 7 | inherit peer-session <i>session-template-name</i> 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 |

| | Command or Action | Purpose |
|---------------|--|---|
| | | template can contain up to seven indirectly inherited peer session templates. |
| Step 8 | end Example: Device(config-router) # end | Exits session-template configuration mode and enters privileged EXEC mode. |
| Step 9 | 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 <i>session-template-name</i> argument. This command also supports all standard output modifiers. |

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

This task configures a device to send a peer session template to a neighbor to inherit the configuration from the specified peer session template with the **neighbor inherit peer-session** command. Use the following steps to send a peer session template configuration to a neighbor to inherit.

To configure peer session template inheritance, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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 <i>autonomous-system-number</i> Example: Device(config)# router bgp 101 | Enters router configuration mode and creates a BGP routing process. |
| Step 4 | neighbor <i>ip-address</i> remote-as <i>autonomous-system-number</i> Example: | Configures a peering session with the specified neighbor. The explicit remote-as statement is required for the neighbor inherit statement in Step 5 to |

| | Command or Action | Purpose |
|---------------|--|---|
| | Device(config-router)# neighbor 172.16.0.1 remote-as 202 | work. If a peering is not configured, the specified neighbor in Step 5 will not accept the session template. |
| Step 5 | neighbor <i>ip-address</i> inherit peer-session <i>session-template-name</i> 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 | Exits router configuration mode and enters privileged EXEC mode. |
| Step 7 | show ip bgp template peer-session <i>[session-template-name]</i> 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 <i>session-template-name</i> argument. This command also supports all standard output modifiers. |

Configuring Peer Policy Templates

Use the following tasks to create and configure a peer policy template:

Configuring Basic Peer Policy Templates

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



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 basic peer policy templates, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| 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 <i>autonomous-system-number</i> Example: Device(config)# router bgp 45000 | Enters router configuration mode and creates a BGP routing process. |
| Step 4 | template peer-policy <i>policy-template-name</i> Example: Device(config-router)# template peer-policy GLOBAL | Enters policy-template configuration mode and creates a peer policy template. |
| Step 5 | maximum-prefix <i>prefix-limit [threshold]</i> [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 <i>weight-value</i> 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 |

| | Command or Action | Purpose |
|---------------|---|--|
| | | supported commands, see the “Peer Policy Templates” section. |
| Step 7 | prefix-list <i>prefix-list-name</i> { in out } Example: <pre>Device(config-router-ptmp) # prefix-list NO-MARKETING in</pre> | (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: <pre>Device(config-router-ptmp) # end</pre> | Exits policy-template configuration mode and returns to privileged EXEC mode. |

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

This task configures peer policy template inheritance using the **`inherit peer-policy`** command. It creates and configure a peer policy template and allows it to inherit a configuration from another peer policy template.



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

To configure peer policy template inheritance, perform this procedure:

Procedure

| | Command or Action | Purpose |
|---------------|--|---|
| Step 1 | enable Example: <pre>Device> enable</pre> | Enables privileged EXEC mode. Enter your password if prompted. |
| Step 2 | configure terminal Example: <pre>Device# configure terminal</pre> | Enters global configuration mode. |
| Step 3 | router bgp <i>autonomous-system-number</i> Example: | Enters router configuration mode and creates a BGP routing process. |

| | Command or Action | Purpose |
|---------------|---|--|
| | Device(config)# <code>router bgp 45000</code> | |
| Step 4 | template peer-policy <i>policy-template-name</i> Example: Device(config-router)# <code>template peer-policy NETWORK1</code> | Enter policy-template configuration mode and creates a peer policy template. |
| Step 5 | route-map <i>map-name</i> {in out} Example: Device(config-router-ptmp)# <code>route-map ROUTE in</code> | (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 <i>policy-template-name sequence-number</i> Example: Device(config-router-ptmp)# <code>inherit peer-policy GLOBAL 10</code> | Configures the peer policy template to inherit the configuration of another peer policy template. <ul style="list-style-type: none"> • The <i>sequence-number</i> 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)# <code>end</code> | Exits policy-template configuration mode and returns to privileged EXEC mode. |
| Step 8 | show ip bgp template peer-policy [<i>policy-template-name</i>][detail] Example: | Displays locally configured peer policy templates. <ul style="list-style-type: none"> • The output can be filtered to display a single peer policy template with the |

| | Command or Action | Purpose |
|--|---|--|
| | Device# <code>show ip bgp template peer-policy NETWORK1 detail</code> | <p><i>policy-template-name</i> argument. This command also supports all standard output modifiers.</p> <ul style="list-style-type: none"> Use the detail keyword to display detailed policy information. |

Examples

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 policies: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

This task configures a device to send a peer policy template to a neighbor to inherit using the **neighbor inherit peer-policy** command. Perform the following steps to send a peer policy template configuration to a neighbor to inherit.

When BGP neighbors use multiple levels of peer templates, it can be difficult to determine which policies are applied to the neighbor. The **policy** and **detail** keywords of the `show ip bgp neighbors` command display the inherited policies and policies that are configured directly on the specified neighbor.

To configure peer policy template, perform this procedure:

Procedure

| | Command or Action | Purpose |
|--------|---|---|
| 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 <i>autonomous-system-number</i> Example: Device(config)# router bgp 45000 | Enters router configuration mode and creates a BGP routing process. |
| Step 4 | neighbor <i>ip-address</i> remote-as <i>autonomous-system-number</i> 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 <i>vrf-name</i>] 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 <i>ip-address</i> inherit peer-policy <i>policy-template-name</i> Example: Device(config-router-af)# neighbor 192.168.1.2 inherit peer-policy GLOBAL | Sends a peer policy 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: | Exits address family configuration mode and returns to privileged EXEC mode. |

| | Command or Action | Purpose |
|---------------|--|---|
| | Device(config-router-af)# end | |
| Step 8 | show ip bgp neighbors [<i>ip-address</i>] policy [detail] Example: Device# show ip bgp neighbors 192.168.1.2 policy | Displays locally configured peer policy templates. <ul style="list-style-type: none"> • The output can be filtered to display a single peer policy template with the <i>policy-template-name</i> 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. |

Examples

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 policies 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 policies:
  prefix-list NO-MARKETING in
  route-map ROUTE in
  weight 300
  maximum-prefix 10000
```

Configuring BGP Route Map Next-hop Self

Perform this task to modify the existing route map by adding the ip next-hop self-setting and overriding the bgp next-hop unchanged and bgp next-hop unchanged all-paths settings.

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

Procedure

| | Command or Action | Purpose |
|---------------|---|---|
| Step 1 | enable Example: Device> enable | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|--------|---|---|
| Step 2 | configure terminal Example: Device# configure terminal | Enters global configuration mode. |
| Step 3 | route-map map-tag permit <i>sequence-number</i> 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 <i>sequence-number</i> 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. |

| | Command or Action | Purpose |
|---------|---|--|
| Step 10 | match source-protocol <i>source-protocol</i> Example: <pre>Device(config-route-map)# match source-protocol connected</pre> | Matches Enhanced Interior Gateway Routing Protocol (EIGRP) external routes based on a source protocol. |
| Step 11 | exit Example: <pre>Device(config-route-map)# exit</pre> | Exits route-map configuration mode and enters global configuration mode. |
| Step 12 | router bgp <i>autonomous-system-number</i> Example: <pre>Device(config)# router bgp 45000</pre> | Enters router configuration mode and creates a BGP routing process. |
| Step 13 | neighbor <i>{ip-address ipv6-address peer-group-name}</i> remote-as <i>autonomous-system-number</i> Example: <pre>Device(config-router)# neighbor 172.16.232.50 remote-as 65001</pre> | Adds an entry to the BGP or multiprotocol BGP neighbor table. |
| Step 14 | address-family vpnv4 Example: <pre>Device(config-router)# address-family vpnv4</pre> | Specifies the VPNv4 address family and enters address family configuration mode. |
| Step 15 | neighbor <i>{ip-address ipv6-address peer-group-name}</i> activate Example: <pre>Device(config-router-af)# neighbor 172.16.232.50 activate</pre> | Enables the exchange of information with a Border Gateway Protocol (BGP) neighbor. |
| Step 16 | neighbor <i>{ip-address ipv6-address peer-group-name}</i> next-hop unchanged allpaths Example: <pre>Device(config-router-af)# neighbor 172.16.232.50 next-hop unchanged allpaths</pre> | Enables an external EBGP peer that is configured as multihop to propagate the next hop unchanged. |
| Step 17 | neighbor <i>{ip-address ipv6-address peer-group-name}</i> route-map map-name out | Applies a route map to an outgoing route. |

| | Command or Action | Purpose |
|----------------|---|---|
| | Example: <pre>Device(config-router-af)# neighbor 172.16.232.50 route-map static-nexthop-rewrite out</pre> | |
| Step 18 | exit Example: <pre>Device(config-router-af)# exit</pre> | Exits address family configuration mode and enters router configuration mode. |
| Step 19 | address-family ipv4 [unicast multicast vrf vrf-name] Example: <pre>Device(config-router)# address-family ipv4 unicast vrf inside</pre> | Specifies the IPv4 address family and enters address family configuration mode. |
| Step 20 | bgp route-map priority Example: <pre>Device(config-router-af)# bgp route-map priority</pre> | Configures the route map priority for the local BGP routing process |
| Step 21 | redistribute protocol Example: <pre>Device(config-router-af)# redistribute static</pre> | Redistributes routes from one routing domain into another routing domain. |
| Step 22 | redistribute protocol Example: <pre>Device(config-router-af)# redistribute connected</pre> | Redistributes routes from one routing domain into another routing domain. |
| Step 23 | exit-address-family Example: <pre>Device(config-router-af)# exit address-family</pre> | Exits address family configuration mode and enters router configuration mode. |
| Step 24 | end Example: <pre>Device(config-router)# end</pre> | Exits router configuration mode and enters privileged EXEC mode. |

Configuration Examples for BGP

The following sections provide configuration examples for BGP.

Example: 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 ?
```

Example: 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
```

Examples: 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
```

Example: 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

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

You can display specific statistics, such as the contents of BGP routing tables, caches, and databases. You can use the information to get resource utilization and solve network problems. You can also display information about node reachability and discover the routing path your device's packets are taking through the network.

The table given below lists the privileged EXEC commands for clearing and displaying BGP.

Table 3: IP BGP Clear and Show Commands

| | |
|---|---|
| clear ip bgp <i>address</i> | Resets a particular BGP connection. |
| clear ip bgp * | Resets all BGP connections. |
| clear ip bgp peer-group <i>tag</i> | Removes all members of a BGP peer group. |
| show ip bgp <i>prefix</i> | Displays peer groups and peers not in peer groups to which has been advertised. Also displays prefix attributes such as hop and the local prefix. |
| show ip bgp cidr-only | Displays all BGP routes that contain subnet and supernet masks. |
| show ip bgp community [<i>community-number</i>] [exact] | Displays routes that belong to the specified communities. |

| | |
|--|---|
| show ip bgp community-list <i>community-list-number</i> [exact-match] | Displays routes that are permitted by the community list. |
| show ip bgp filter-list <i>access-list-number</i> | Displays routes that are matched by the specified AS path filter. |
| show ip bgp inconsistent-as | Displays the routes with inconsistent originating autonomous system (AS) numbers. |
| show ip bgp regexp <i>regular-expression</i> | Displays the routes that have an AS path that matches the regular expression entered on the command line. |
| show ip bgp | Displays the contents of the BGP routing table. |
| show ip bgp neighbors [<i>address</i>] | Displays detailed information on the BGP and TCP connections to individual neighbors. |
| show ip bgp neighbors [<i>address</i>] [advertised-routes dampened-routes flap-statistics paths <i>regular-expression</i> 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 [<i>tag</i>] [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.

Feature History for Border Gateway Protocol

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

| Release | Feature | Feature Information |
|------------------------------|-------------------------|--|
| Cisco IOS XE Everest 16.5.1a | Border Gateway Protocol | The Border Gateway Protocol (BGP) is an exterior gateway protocol used to set up an interdomain routing system that guarantees the loop-free exchange of routing information between autonomous systems. |

| Release | Feature | Feature Information |
|--------------------------------|---------------------------------|--|
| Cisco IOS XE Gibraltar 16.11.1 | Conditional BGP Route Injection | Conditional BGP Route Injection allows you to originate a prefix into a BGP routing table without the corresponding match. |
| | BGP Peer Templates | A BGP Peer Template is a configuration pattern that can be applied to neighbors that share policies. Peer templates are reusable and support inheritance, which allows the network operator to group and apply distinct neighbor configurations for BGP neighbors that share policies. |
| | BGP Route Map Next Hop Self | The BGP Route Map Next Hop Self feature provides a way to override the settings for <code>bgp next-hop unchanged</code> and <code>bgp next-hop unchanged allpath</code> selectively. |

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to <https://cfng.cisco.com/>