Configuring BGP

This chapter describes how to configure Border Gateway Protocol (BGP). For a complete description of the BGP commands in this chapter, refer to the "BGP Commands" chapter of the *Network Protocols Command Reference, Part 1*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

The Border Gateway Protocol, as defined in RFCs 1163 and 1267, is an Exterior Gateway Protocol (EGP). It allows you to set up an interdomain routing system that automatically guarantees the loop-free exchange of routing information between autonomous systems.

For protocol-independent features, see the chapter "Configuring IP Routing Protocol-Independent Features" in this document.

Cisco's BGP Implementation

In BGP, each route consists of a network number, a list of autonomous systems that information has passed through (called the *autonomous system path*), and a list of other *path attributes*. We support BGP Versions 2, 3, and 4, as defined in RFCs 1163, 1267, and 1771, respectively.

The primary function of a BGP system is to exchange network reachability information with other BGP systems, including information about the list of autonomous system paths. This information can be used to construct a graph of autonomous system connectivity from which routing loops can be pruned and with which autonomous system-level policy decisions can be enforced.

You can configure the value for the Multi Exit Discriminator (MED) metric attribute using route maps. (The name of this metric for BGP Versions 2 and 3 is INTER_AS_METRIC.) When an update is sent to an IBGP peer, the MED is passed along without any change. This action enables all the peers in the same autonomous system to make a consistent path selection.

A third-party next-hop router address is used in the NEXT_HOP attribute, regardless of the autonomous system of that third-party router. The Cisco IOS software automatically calculates the value for this attribute.

Transitive, optional path attributes are passed along to other BGP-speaking routers.

BGP Version 4 supports classless interdomain routing (CIDR), which lets you 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. CIDR routes can be carried by OSPF, Enhanced IGRP, and ISIS-IP, and RIP.

See the "BGP Route Map Examples" section at the end of this chapter for examples of how to use route maps to redistribute BGP Version 4 routes.

How BGP Selects Paths

A router running Cisco IOS Release 12.0 or later does not select or use an IBGP route unless both of the following are true:

- the router has a route available to the next-hop router
- the router has received synchronization via an IGP (unless IGP synchronization has been disabled)

BGP bases its decision process on the attribute values. When faced with multiple routes to the same destination, BGP chooses the best route for routing traffic toward the destination. The following process summarizes how BGP chooses the best route.

1. If the next hop is inaccessible, do not consider it.

   This is why it is important to have an IGP route to the next hop.
2. If the path is internal, synchronization is enabled, and the route is not in the IGP, do not consider the route.
3. Prefer the path with the largest weight (weight is a Cisco proprietary parameter).
4. If the routes have the same weight, prefer the route with the largest local preference.
5. If the routes have the same local preference, prefer the route that was originated by the local router.

   For example, a route might be originated by the local router using the **network bgp** command, or through redistribution from an IGP.
6. If the local preference is the same, or if no route was originated by the local router, prefer the route with the shortest autonomous system path.
7. If the autonomous system path length is the same, prefer the route with the lowest origin code (IGP < EGP < INCOMPLETE).
8. If the origin codes are the same, prefer the route with the lowest Multi Exit Discriminator (MED) metric attribute.

   This comparison is only done if the neighboring autonomous system is the same for all routes considered, unless **bgp always-compare-med** is enabled.
9. Prefer an external (EBGP) path over an internal (IBGP) path.

   All confederation paths are considered internal paths.
10. Prefer the route that can be reached through the closest IGP neighbor (the lowest IGP metric).

    This means the router will prefer the shortest internal path within the autonomous system to reach the destination (the shortest path to the BGP next-hop).
11. 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.

    Note EBGP load sharing can occur at this point, which means that multiple paths can be installed in the forwarding table.
12. 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.

BGP Multipath Support

When a BGP speaker learns two identical EBGP paths for a prefix from a neighboring AS, it will choose the path with the lowest route-id as the best path. This best path is installed in the IP routing table. If BGP multipath support is enabled and the EBGP paths are learned from the same neighboring AS, instead of picking one best path, multiple paths are installed in the IP routing table.

During packet switching, depending on the switching mode, either per-packet or per-destination load balancing is performed among the multiple paths. A maximum of six paths is supported. The **maximum-paths** router configuration command controls the number of paths allowed. By default, BGP will install only one path to the IP routing table.

Basic BGP Configuration Tasks

The BGP configuration tasks are divided into basic and advanced tasks. The first three basic tasks are required to configure BGP; the remaining basic and advanced tasks are optional.

Basic BGP configuration tasks are discussed in the following sections:

- Enable BGP Routing
- Configure BGP Neighbors
- Configure BGP Soft Reconfiguration
- Reset BGP Connections
- Configure BGP Interactions with IGPs
- Configure BGP Administrative Weights
- Configure BGP Route Filtering by Neighbor
- Configure BGP Path Filtering by Neighbor
- Disable Next-Hop Processing on BGP Updates
- Configure the BGP Version
- Set the Network Weight
- Configure the Multi Exit Discriminator Metric
- Monitor and Maintain BGP

Advanced BGP Configuration Tasks

Advanced, optional BGP configuration tasks are discussed in the following sections:

- Use Route Maps to Modify Updates
- Reset EBGP Connections Immediately upon Link Failure
- Configure Aggregate Addresses
- Disable Automatic Summarization of Network Numbers
- Configure BGP Community Filtering
- Configure a Routing Domain Confederation

- Configure a Route Reflector
- Configure Neighbor Options
- Configure BGP Peer Groups
- Indicate Backdoor Routes
- Modify Parameters While Updating the IP Routing Table
- Set Administrative Distance
- Adjust BGP Timers
- Change the Local Preference Value
- Redistribute Network 0.0.0.0
- Select Path Based on MEDs from Other Autonomous Systems
- Configure Route Dampening

For information on configuring features that apply to multiple IP routing protocols (such as redistributing routing information), see the chapter "Configuring IP Routing Protocol-Independent Features."

Enable BGP Routing

To enable BGP routing, establish a BGP routing process by performing the following steps starting in global configuration mode:

Task		Command
Step 1	Enable a BGP routing process, which places you in router configuration mode.	**router bgp** *autonomous-system*
Step 2	Flag a network as local to this autonomous system and enter it to the BGP table.	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*]

Note For exterior protocols, a reference to an IP network from the **network** router configuration command controls only which networks are advertised. This is in contrast to Interior Gateway Protocols (IGP), such as IGRP, which also use the **network** command to determine where to send updates.

Note The **network** command is used to inject IGP routes into the BGP table. The *network-mask* portion of the command allows supernetting and subnetting. A maximum of 200 entries of the command are accepted. Alternatively, you could use the **redistribute** command to achieve the same result.

Configure BGP Neighbors

Like other Exterior Gateway Protocols (EGPs), BGP must completely understand the relationships it has with its neighbors. BGP supports two kinds of neighbors: internal and external. *Internal neighbors* are in the same autonomous system; *external neighbors* are in different autonomous systems. Normally, external neighbors are adjacent to each other and share a subnet, while internal neighbors may be anywhere in the same autonomous system.

To configure BGP neighbors, perform the following task in router configuration mode:

Task	Command
Specify a BGP neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number*

See the "BGP Neighbor Configuration Examples" section at the end of this chapter for an example of configuring BGP neighbors.

Configure BGP Soft Reconfiguration

Whenever there is a change in the policy, the BGP session has to be cleared for the new policy to take effect. Clearing a BGP session causes cache invalidation and results in a tremendous impact on the operation of networks.

Soft reconfiguration allows policies to be configured and activated without clearing the BGP session. Soft reconfiguration can be done on a per-neighbor basis.

- When soft reconfiguration is used to generate inbound updates from a neighbor, it is called inbound soft reconfiguration.
- When soft reconfiguration is used to send a new set of updates to a neighbor, it is called outbound soft reconfiguration.

Performing inbound reconfiguration enables the new inbound policy to take effect. Performing outbound reconfiguration causes the new local outbound policy take effect without resetting the BGP session. As a new set of updates is sent during outbound policy reconfiguration, a new inbound policy of the neighbor can also take effect.

In order to generate new inbound updates without resetting the BGP session, the local BGP speaker should store all the received updates without modification, regardless of whether it is accepted or denied by the current inbound policy. This is memory intensive and should be avoided. On the other hand, outbound soft reconfiguration does not have any memory overhead. One could trigger an outbound reconfiguration in the other side of the BGP session to make the new inbound policy take effect.

To allow inbound reconfiguration, BGP should be configured to store all received updates. Outbound reconfiguration does not require preconfiguration.

You can configure the Cisco IOS software to start storing received updates, which is required for inbound BGP soft reconfiguration. Outbound reconfiguration does not require inbound soft reconfiguration to be enabled.

To configure BGP soft configuration, perform the following task in router configuration mode:

Task	Command
Configure BGP soft reconfiguration.	**neighbor** {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**

Our implementation of BGP supports BGP Versions 2, 3, and 4. If the neighbor does not accept default Version 4, dynamic version negotiation is implemented to negotiate down to Version 2.

If you specify a BGP peer group by using the *peer-group-name* argument, all members of the peer group will inherit the characteristic configured with this command.

Reset BGP Connections

Once you have defined two routers to be BGP neighbors, they will form a BGP connection and exchange routing information. If you subsequently change a BGP filter, weight, distance, version, or timer, or make a similar configuration change, you must reset BGP connections for the configuration change to take effect. Perform either of the following tasks in EXEC mode to reset BGP connections:

Task	Command
Reset a particular BGP connection.	**clear ip bgp** *address*
Reset all BGP connections.	**clear ip bgp ***

Configure BGP Interactions with IGPs

If your autonomous system will be passing traffic through it from another autonomous system to a third autonomous system, it is very important that your autonomous system be consistent about the routes that it advertises. For example, if your BGP were to advertise a route before all routers in your network had learned about the route through your IGP, your autonomous system could receive traffic that some routers cannot yet route. To prevent this from happening, BGP must wait until the IGP has propagated routing information across your autonomous system. This causes BGP to be *synchronized* with the IGP. Synchronization is enabled by default.

In some cases, you do not need synchronization. If you will not be passing traffic from a different autonomous system through your autonomous system, or if all routers in your autonomous system will be running BGP, you can disable synchronization. Disabling this feature can allow you to carry fewer routes in your IGP and allow BGP to converge more quickly. To disable synchronization, perform the following task in router configuration mode:

Task	Command
Disable synchronization between BGP and an IGP.	**no synchronization**

When you disable synchronization, you should also clear BGP sessions using the **clear ip bgp** command.

See the “BGP Path Filtering by Neighbor Example” section at the end of this chapter for an example of BGP synchronization.

In general, you will not want to redistribute most BGP routes into your IGP. A common design is to redistribute one or two routes and to make them exterior routes in IGRP, or have your BGP speaker generate a default route for your autonomous system. When redistributing from BGP into IGP, only the routes learned using EBGP get redistributed.

In most circumstances, you also will not want to redistribute your IGP into BGP. Just list the networks in your autonomous system with **network** router configuration commands and your networks will be advertised. Networks that are listed this way are referred to as *local networks* and have a BGP origin attribute of “IGP.” They must appear in the main IP routing table and can have any source; for example, they can be directly connected or learned via an IGP. The BGP routing process periodically scans the main IP routing table to detect the presence or absence of local networks, updating the BGP routing table as appropriate.

If you do perform redistribution into BGP, you must be very careful about the routes that can be in your IGP, especially if the routes were redistributed from BGP into the IGP elsewhere. This creates a situation where BGP is potentially injecting information into the IGP and then sending such information back into BGP, and vice versa.

Networks that are redistributed into BGP from the EGP protocol will be given the BGP origin attribute “EGP.” Other networks that are redistributed into BGP will have the BGP origin attribute of “incomplete.” The origin attribute in our implementation is only used in the path selection process.

Configure BGP Administrative Weights

An administrative weight is a number that you can assign to a path so that you can control the path selection process. The administrative weight is local to the router. A weight can be a number from 0 to 65535. Paths that the Cisco IOS software originates have weight 32768 by default; other paths have weight 0. If you have particular neighbors that you want to prefer for most of your traffic, you can assign a higher weight to all routes learned from that neighbor.

Perform the following task in router configuration mode to configure BGP administrative weights:

Task	Command
Specify a weight for all routes from a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **weight** *weight*

In addition, you can assign weights based on autonomous system path access lists. A given weight becomes the weight of the route if the autonomous system path is accepted by the access list. Any number of weight filters are allowed.

To assign weights based on autonomous system path access lists, perform the following tasks starting in global configuration mode:

Task		Command
Step 1	Define a BGP-related access list.	**ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expression*
Step 2	Enter router configuration mode.	**router bgp** *autonomous-system*
Step 3	Configure administrative weight on all incoming routes matching an autonomous system path filter.	**neighbor** *ip-address* **filter-list** *access-list-number* **weight** *weight*

Configure BGP Route Filtering by Neighbor

If you want to restrict the routing information that the Cisco IOS software learns or advertises, you can filter BGP routing updates to and from particular neighbors. To do this, define an access list and apply it to the updates. Distribute-list filters are applied to network numbers and not autonomous system paths.

To filter BGP routing updates, perform the following task in router configuration mode:

Task	Command
Filter BGP routing updates to/from neighbors as specified in an access list.	**neighbor** {*ip-address* | *peer-group-name*} **distribute-list** *access-list-number* | *name* {**in** | **out**}

Configure BGP Path Filtering by Neighbor

In addition to filtering routing updates based on network numbers, you can 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 do this, define an autonomous system path access list and apply it to updates to and from particular neighbors. See the "Regular Expressions" appendix in the *Dial Solutions Command Reference* for more information on forming regular expressions.

To configure BGP path filtering, perform the following tasks starting in global configuration mode:

Task		Command
Step 1	Define a BGP-related access list.	**ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expression*
Step 2	Enter router configuration mode.	**router bgp** *autonomous-system*
Step 3	Establish a BGP filter.	**neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number* {**in** | **out** | **weight** *weight*}

See the "BGP Path Filtering by Neighbor Example" section at the end of this chapter for an example of BGP path filtering by neighbor.

Disable Next-Hop Processing on BGP Updates

You can configure the Cisco IOS software to disable next-hop processing for BGP updates to a neighbor. This might be useful in nonmeshed networks such as Frame Relay or X.25, where BGP neighbors might not have direct access to all other neighbors on the same IP subnet.

To disable next-hop processing, perform the following task in router configuration mode:

Task	Command
Disable next-hop processing on BGP updates to a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **next-hop-self**

Configuring this command causes the current router to advertise itself as the next hop for the specified neighbor. Therefore, other BGP neighbors will forward to it packets for that address. This is useful in a nonmeshed environment, since you know that a path exists from the present router to that address. In a fully meshed environment, this is not useful, since it will result in unnecessary extra hops and because there might be a direct access through the fully meshed cloud with fewer hops.

Configure the BGP Version

By default, BGP sessions begin using BGP Version 4 and negotiating downward to earlier versions if necessary. To prevent negotiation and force the BGP version used to communicate with a neighbor, perform the following task in router configuration mode:

Task	Command
Specify the BGP version to use when communicating with a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **version** *value*

Set the Network Weight

Weight is a parameter that affects the best path selection process. To set the absolute weight for a network, perform the following task in router configuration mode:

Task	**Command**
Set the weight for a network.	**network** *address mask* **weight** *weight* [**route-map** *map-name*]

Configure the Multi Exit Discriminator Metric

BGP uses the Multi Exit Discriminator (MED) metric as a hint to external neighbors about preferred paths. (The name of this metric for BGP Versions 2 and 3 is INTER_AS_METRIC.) You can set the MED of the redistributed routes by performing the following task. All the routes without a MED will also be set to this value. Perform the following task in router configuration mode:

Task	**Command**
Set a multi exit discriminator.	**default-metric** *number*

Alternatively, you can set the MED using the **route-map** command. See the "BGP Route Map Examples" section at the end of this chapter for examples of using BGP route maps.

Advanced BGP Configuration Tasks

This section contains advanced BGP configuration tasks.

Use Route Maps to Modify Updates

You can use a route map on a per-neighbor basis to filter updates and 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 the inbound and the outbound updates, we support matching based on autonomous system path, community, and network numbers. Autonomous system path matching requires the **as-path access-lis**t command, community based matching requires the **community-list** command and network-based matching requires the **ip access-list** command. Perform the following task in router configuration mode:

Task	**Command**
Apply a route map to incoming or outgoing routes.	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *route-map-name* {**in** | **out**}

See the "BGP Route Map Examples" section at the end of this chapter for BGP route-map examples.

Reset EBGP Connections Immediately upon Link Failure

Normally, when a link between external neighbors goes down, the BGP session will not be reset immediately. If you want the EBGP session to be reset as soon as an interface goes down, perform the following task in router configuration mode:

Task	Command
Automatically reset EBGP sessions.	**bgp fast-external-fallover**

Configure Aggregate Addresses

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 using the conditional aggregation feature described in the following task table. An aggregate address will be added to the BGP table if there is at least one more specific entry in the BGP table.

To create an aggregate address in the routing table, perform one or more of the following tasks in router configuration mode:

Task	Command
Create an aggregate entry in the BGP routing table.	**aggregate-address** *address mask*
Generate an aggregate with AS-SET.	**aggregate-address** *address mask* **as-set**
Advertise summary addresses only.	**aggregate-address** *address-mask* **summary-only**
Suppress selected, more specific routes.	**aggregate-address** *address mask* **suppress-map** *map-name*
Generate an aggregate based on conditions specified by the route map.	**aggregate-address** *address mask* **advertise-map** *map-name*
Generate an aggregate with attributes specified in the route map.	**aggregate-address** *address mask* **attribute-map** *map-name*

See the “BGP Aggregate Route Examples” section at the end of this chapter for examples of using BGP aggregate routes.

Disable Automatic Summarization of Network Numbers

In BGP Version 3, when a subnet is redistributed from an IGP into BGP, only the network route is injected into the BGP table. By default, this automatic summarization is enabled. To disable automatic network number summarization, perform the following task in router configuration mode:

Task	Command
Disable automatic network summarization.	**no auto-summary**

Configure BGP Community Filtering

BGP supports transit policies via controlled distribution of routing information. The distribution of routing information is based on one of the following three values:

- IP address (see the “Configure BGP Route Filtering by Neighbor” section earlier in this chapter).
- The value of the AS_PATH attribute (see the “Configure BGP Path Filtering by Neighbor” section earlier in this chapter).
- The value of the COMMUNITIES attribute (as described in this section).

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

A *community* is a group of destinations that share some common attribute. Each destination can belong to multiple communities. Autonomous system administrators can define to which communities a destination belongs. By default, all destinations belong to the general Internet community. The community is carried as the COMMUNITIES attribute.

The COMMUNITIES attribute is an optional, transitive, global attribute in the numerical range from 1 to 4,294,967,200. Along with Internet community, there are a few predefined, well-known communities, as follows:

- **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).

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 you learn, advertise, or redistribute 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. Just like 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.

To create a community list, perform the following task in global configuration mode:

Task	Command
Create a community list.	**ip community-list** *community-list-number* {**permit** | **deny**} *community-number*

To set the COMMUNITIES attribute and match clauses based on communities, see the **match community-list** and **set community** commands in the “Redistribute Routing Information” section in the “Configuring IP Routing Protocol-Independent Features” chapter.

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 performing the following task in router configuration mode:

Task	Command
Specify that the COMMUNITIES attribute be sent to the neighbor at this IP address.	**neighbor** {*ip-address* | *peer-group-name*} **send-community**

Configure a Routing Domain Confederation

One way to reduce the IBGP mesh is to divide an autonomous system into multiple autonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself, and has a few connections to other autonomous systems in the same confederation. 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 is preserved. This enables to us to retain a single Interior Gateway Protocol (IGP) for all of the autonomous systems.

To configure a BGP confederation, you must specify a confederation identifier. To the outside world, the group of autonomous systems will look like a single autonomous system with the confederation identifier as the autonomous system number. To configure a BGP confederation identifier, perform the following tasks in router configuration mode:

Task	Command
Configure a BGP confederation.	**bgp confederation identifier** *autonomous-system*

In order to treat the neighbors from other autonomous systems within the confederation as special EBGP peers, perform the following task in router configuration mode:

Task	Command
Specify the autonomous systems that belong to the confederation.	**bgp confederation peers** *autonomous-system* [*autonomous-system* ...]

See the “BGP Confederation Example” section at the end of this chapter for an example configuration of several peers in a confederation.

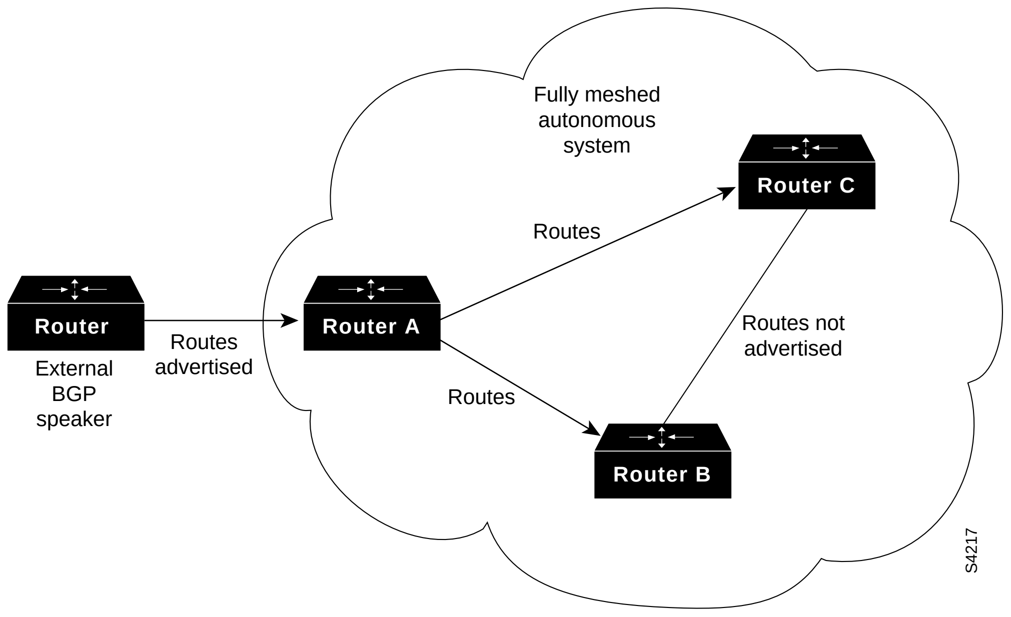
For an alternative way to reduce the IBGP mesh, see the next section, “Configure a Route Reflector.”

Configure a Route Reflector

BGP requires that all of the IBGP speakers be fully meshed. However, this requirement does not scale when there are many IBGP speakers. Instead of configuring a confederation, another way to reduce the IBGP mesh is to configure a *route reflector.*

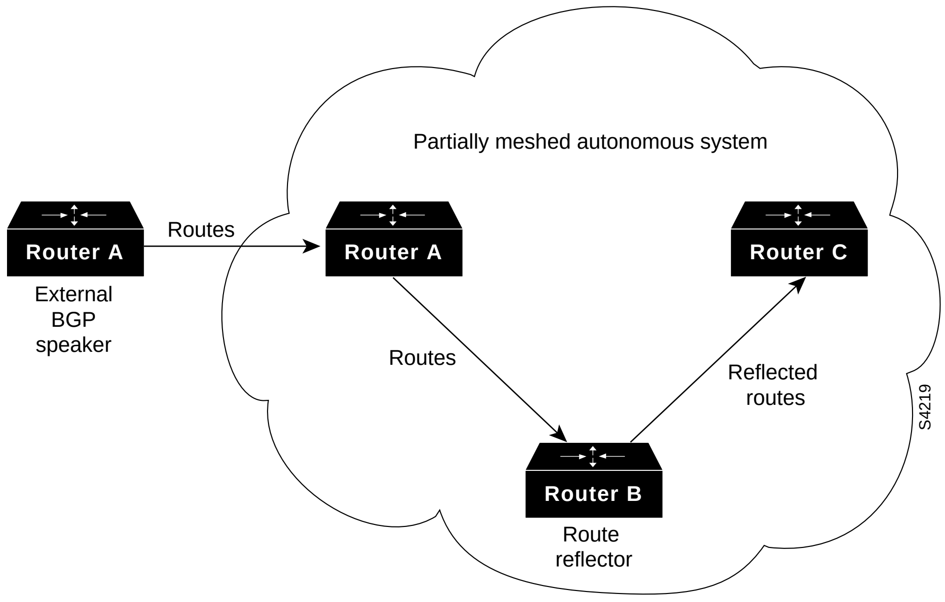
Figure 25 illustrates a simple IBGP configuration with three IBGP speakers (Routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both Routers B and C. Routers B and C do not readvertise the IBGP learned route to other IBGP speakers because the routers do not pass routes learned from internal neighbors on to other internal neighbors, thus preventing a routing information loop.

Figure 25 Three Fully Meshed IBGP Speakers



With route reflectors, all IBGP speakers need not be fully meshed because there is a method to pass learned routes to neighbors. In this model, an internal BGP peer is configured to be a route reflector responsible for passing IBGP learned routes to a set of IBGP neighbors. In Figure 26, Router B is configured as a route reflector. When the route reflector receives routes advertised from Router A, it advertises them to Router C, and vice versa. This scheme eliminates the need for the IBGP session between Routers A and C.

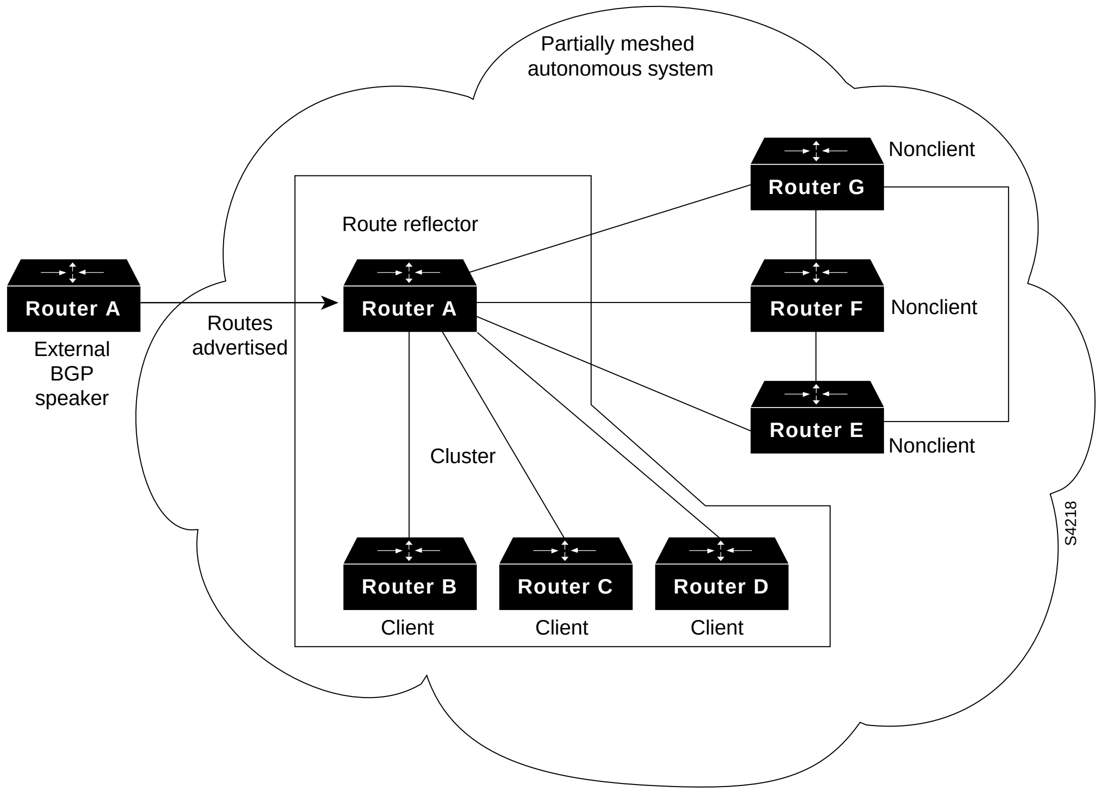
Figure 26 Simple BGP Model with a Route Reflector



The internal peers of the route reflector are divided into two groups: client peers and all the other routers in the autonomous system (nonclient peers). A route reflector reflects routes between these two groups. The route reflector and its client peers form a *cluster*. The nonclient peers must be fully meshed with each other, but the client peers need not be fully meshed. The clients in the cluster do not communicate with IBGP speakers outside their cluster.

Figure 27 illustrates a more complex route reflector scheme. Router A is the route reflector in a cluster with Routers B, C, and D. Routers E, F, and G are fully meshed, nonclient routers.

Figure 27 More Complex BGP Route Reflector Model



When the route reflector receives an advertised route, depending on the neighbor, it does the following:

- 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.

To configure a route reflector and its clients, perform the following task in router configuration mode:

Task	Command
Configure the local router as a BGP route reflector and the specified neighbor as a client.	**neighbor** *ip-address* **route-reflector-client**

Along with route reflector-aware BGP speakers, it is possible to have BGP speakers that do not understand the concept of route reflectors. They can be members of either client or nonclient groups. This allows easy, gradual migration from the old BGP model to the route reflector model. Initially, you could create a single cluster with a route reflector and a few clients. All the other IBGP speakers could be nonclient peers to the route reflector and then more clusters could be created gradually.

An autonomous system can have multiple route reflectors. A route reflector treats other route reflectors just like other IBGP speakers. A route reflector can be configured to have other route reflectors in a client group or nonclient group. In a simple configuration, the backbone could be divided into many clusters. Each route reflector would be configured with other route reflectors as nonclient peers (thus, all the route reflectors will be fully meshed). The clients are configured to maintain IBGP sessions with only the route reflector in their cluster.

Usually a cluster of clients will have a single route reflector. In that case, the cluster is identified by the router ID of the route reflector. To increase redundancy and avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the 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 all of them should have identical sets of client and nonclient peers.

If the cluster has more than one route reflector, configure the cluster ID by performing the following task in router configuration mode:

Task	Command
Configure the cluster ID.	**bgp cluster-id** *cluster-id*

Use the **show ip bgp** command to display the originator ID and the cluster-list attributes.

By default, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients. However, if the clients are fully meshed, the route reflector does not need to reflect routes to clients. To disable client-to-client route reflection, perform the following task in router configuration mode:

Task	Command
Disable client-to-client route reflection.	**no bgp client-to-client reflection**

Note If client-to-client reflection is enabled, the clients of a route reflector cannot be members of a peer group.

As the IBGP learned routes are reflected, it is possible for routing information to loop. The route reflector model has the following mechanisms to avoid routing loops:

- Originator-ID is an optional, nontransitive BGP attribute. This is a 4-byte attributed created by a route reflector. The attribute carries the router ID of the originator of the route in the local autonomous system. Therefore, if a misconfiguration causes routing information to come back to the originator, the information is ignored.
- Cluster-list is an optional, nontransitive BGP attribute. It is a sequence of cluster IDs that the route has passed. When a route reflector reflects a route from its clients to nonclient peers, it appends the local cluster ID to the cluster-list. If the cluster-list is empty, it creates a new one. Using this attribute, a route reflector can identify if routing information is looped back to the same cluster due to misconfiguration. If the local cluster ID is found in the cluster-list, the advertisement is ignored.
- Using **set** clauses in outbound route maps modifies attributes, possibly creating routing loops. To avoid this, **set** clauses of outbound route maps are ignored for routes reflected to IBGP peers.

Configure Neighbor Options

There are many ways to customize a BGP neighbor, and these customized features can also be applied to a peer group. To avoid documenting each option twice (for both neighbors and peer groups), all of the supported neighbor options appear in the next section, "Configure BGP Peer Groups" under "Assign Options to the Peer Group."

To provide BGP routing information to a large number of neighbors, you can configure BGP to accept neighbors based on an access list. If a neighbor attempts to initiate a BGP connection, its address must be accepted by the access list for the connection to be accepted. If you do this, the router will not attempt to initiate a BGP connection to these neighbors, so the neighbors must be explicitly configured to initiate the BGP connection. If no access list is specified, all connections are accepted.

Specify an access list of BGP neighbors by performing the following task in router configuration mode:

Task	Command
Specify an access list of BGP neighbors.	**neighbor any** [*access-list-number* | *name*]

Configure BGP Peer Groups

Often, in a BGP speaker, many 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, more importantly, to make updating more efficient. When you have many peers, this approach is highly recommended.

The three steps to configure a BGP peer group, described in the following sections, are:

Step 1 Create the Peer Group

Step 2 Assign Options to the Peer Group

Step 3 Make Neighbors Members of the Peer Group

Create the Peer Group

To create a BGP peer group, perform the following task in router configuration mode:

Task	Command
Create a BGP peer group.	**neighbor** *peer-group-name* **peer-group**

Assign Options to the Peer Group

After you create a peer group, you configure the peer group with **neighbor** commands. By default, members of the peer group inherit all the configuration options of the peer group. Members can also be configured to override the options that do not affect outbound updates.

Peer group members will always inherit the following: remote-as (if configured), version, update-source, out-route-map, out-filter-list, out-dist-list, minimum-advertisement-interval, and next-hop-self. All the peer group members will inherit changes made to the peer group.

Note Limitation on EBGP peer group: Because all members of a peer group each receive one identical copy of an update, all the members (peering addresses) must be in the same logical IP subnet (lis), so that the update is not invalidated or dropped.

To assign configuration options to an individual neighbor, specify any of the following commands using the IP address. To assign the options to a peer group, specify any of the commands using the peer group name. Perform any of these tasks in router configuration mode.

Task	Command
Specify a BGP neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number*
Associate a description with a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **description** *text*
Allow a BGP speaker (the local router) to send the default route 0.0.0.0 to a neighbor for use as a default route.	**neighbor** {*ip-address* | *peer-group-name*} **default-originate** [**route-map** *map-name*]
Specify that the COMMUNITIES attribute be sent to the neighbor at this IP address.	**neighbor** {*ip-address* | *peer-group-name*} **send-community**
Allow internal BGP sessions to use any operational interface for TCP connections.	**neighbor** {*ip-address* | *peer-group-name*} **update-source** *interface*
Allow BGP sessions, even when the neighbor is not on a directly connected segment.	**neighbor** {*ip-address* | *peer-group-name*} **ebgp-multihop**
Set the minimum interval between sending BGP routing updates.	**neighbor** {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds*
Limit the number of prefixes allowed from a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **maximum-prefix** *maximum* [*threshold*] [**warning-only**]
Invoke MD5 authentication on a TCP connection to a BGP peer.	**neighbor** {*ip-address* | *peer-group-name*} **password** *string*
Specify a weight for all routes from a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **weight** *weight*
Filter BGP routing updates to/from neighbors, as specified in an access list.	**neighbor** {*ip-address* | *peer-group-name*} **distribute-list** {*access-list-number* | *name*} {**in** | **out**}
Establish a BGP filter.	**neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number* {**in** | **out** | **weight** *weight*}
Disable next-hop processing on the BGP updates to a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **next-hop-self**
Specify the BGP version to use when communicating with a neighbor.	**neighbor** {*ip-address* | *peer-group-name*} **version** *value*
Apply a route map to incoming or outgoing routes.	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name* {**in** | **out**}
Configure the software to start storing received updates.	**neighbor** {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**

If a peer group is not configured with a remote-as, the members can be configured with the **neighbor remote-as** command. This allows you to create peer groups containing EBGP neighbors.

You can customize inbound policies for peer group members, (using, for example, a distribute list, route map, or filter list) because one identical copy of an update is sent to every member of a group. Therefore, neighbor options related to outgoing updates cannot be customized for peer group members.

External BGP peers normally must reside on a directly connected network. Sometimes it is useful to relax this restriction in order to test BGP; do so by specifying the **neighbor ebgp-multihop** command.

For internal BGP, you might want to allow your BGP connections to stay up regardless of which interface is used to reach a neighbor. To do this, you first configure a *loopback* interface and assign it an IP address. Next, configure the BGP update source to be the loopback interface. Finally, configure your neighbor to use the address on the loopback interface. Now the IBGP session will be up as long as there is a route, regardless of any interface.

You can set the minimum interval of time between BGP routing updates.

You can invoke MD5 authentication between two BGP peers, meaning that each segment sent on the TCP connection between them is verified. This feature must be configured with the same password on both BGP peers; otherwise, the connection between them will not be made. The authentication feature uses the MD5 algorithm. Invoking authentication causes the Cisco IOS software to generate and check the MD5 digest of every segment sent on the TCP connection. If authentication is invoked and a segment fails authentication, then a message appears on the console.

See the "TCP MD5 Authentication for BGP Example" at the end of this chapter for an example of enabling MD5 authentication.

Make Neighbors Members of the Peer Group

Finally, to configure a BGP neighbor to be a member of that BGP peer group, perform the following task in router configuration mode, using the same peer group name:

Task	Command
Make a BGP neighbor a member of the peer group.	**neighbor** *ip-address* **peer-group** *peer-group-name*

See the "BGP Peer Group Examples" section at the end of this chapter for examples of IBGP and EBGP peer groups.

Indicate Backdoor Routes

You can indicate which networks are reachable by using a *backdoor* route that the border router should use. A backdoor network is treated as a local network, except that it is not advertised. To configure backdoor routes, perform the following task in router configuration mode:

Task	Command
Indicate reachable networks through backdoor routes.	**network** *address* **backdoor**

Modify Parameters While Updating the IP Routing Table

By default, when a BGP route is put into the IP routing table, the MED is converted to an IP route metric, the BGP next hop is used as the next hop for the IP route, and the tag is not set. However, you can use a route map to perform mapping. To modify metric and tag information when the IP routing table is updated with BGP learned routes, perform the following task in router configuration mode:

Task	Command
Apply a route map to routes when updating the IP routing table.	**table-map** *route-map name*

Set Administrative Distance

Administrative distance is a measure of the preference of different routing protocols. BGP uses three different administrative distances: external, internal, and local. Routes learned through external BGP are given the external distance, routes learned with internal BGP are given the internal distance, and routes that are part of this autonomous system are given the local distance. To assign a BGP administrative distance, perform the following task in router configuration mode:

Task	Command
Assign a BGP administrative distance.	**distance bgp** *external-distance internal-distance local-distance*

Changing the administrative distance of BGP routes is considered dangerous and generally is not recommended. The external distance should be lower than any other dynamic routing protocol, and the internal and local distances should be higher than any other dynamic routing protocol.

Adjust BGP Timers

BGP uses certain timers to control periodic activities such as the sending of keepalive messages, and the interval after not receiving a keepalive message after which the Cisco IOS software declares a peer dead. By default, the keepalive timer is 60 seconds, and the holdtime timer is 180 seconds. You can adjust these timers. When a connection is started, BGP will negotiate the hold time with the neighbor. The smaller of the two hold times will be chosen. The keepalive timer is then set based on the negotiated hold time and the configured keepalive time. To adjust BGP timers for all neighbors, use the following command in router configuration mode:

Task	Command
Adjust BGP timers.	**timers bgp** *keepalive holdtime*

To adjust BTP keepalive and holdtime timeres for a specific neighbor, use the following command in router configuration mode:

Command	Purpose
neighbor [*ip-address* | *peer group-name*] **timers** *keepalive holdtime*	Sets the keepalive and holdtime timers (in seconds) for the specified peer or peer group.

Note The timers configured for a specific neighbor or peer group override the timers configured for all BGP neighbors using the **timers bgp** command.

To clear the timers for a BGP neighbor or peer group, use the **no** form of the **neighbor timers** command.

Change the Local Preference Value

You can define a particular path as more preferable or less preferable than other paths by changing the default local preference value of 100. To assign a different default local preference value, perform the following task in router configuration mode:

Task	Command
Change the default local preference value.	**bgp default local-preference** *value*

You can use route maps to change the default local preference of specific paths. See the "BGP Route Map Examples" section at the end of this chapter for examples when used with BGP route maps.

Redistribute Network 0.0.0.0

By default, you are not allowed to redistribute network 0.0.0.0. To permit the redistribution of network 0.0.0.0, perform the following task in router configuration mode:

Task	Command
Allow the redistribution of network 0.0.0.0 into BGP.	**default-information originate**

Select Path Based on MEDs from Other Autonomous Systems

The MED is one of the parameters that is considered when selecting the best path among many alternative paths. The path with a lower MED is preferred over a path with a higher MED.

By default, during the best-path selection process, MED comparison is done only among paths from the same autonomous system. You can allow comparison of MEDs among paths regardless of the autonomous system from which the paths are received. To do so, perform the following task in router configuration mode:

Task	Command
Allow the comparison of MEDs for paths from neighbors in different autonomous systems.	**bgp always-compare-med**

Configure Route Dampening

Route 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.

For example, consider a network with three BGP autonomous systems: AS 1, AS 2, and AS 3. Suppose the route to network A in AS 1 flaps (it becomes unavailable). Under circumstances without route dampening, AS 1's EBGP neighbor to AS 2 sends a Withdraw message to AS 2. The border router in AS 2, in turn, propagates the Withdraw to AS 3. When the route to network A reappears, AS 1 sends an Advertisement to AS 2, which sends it to AS 3. If the route to network A repeatedly becomes unavailable, then available, many Withdrawals and Advertisements are sent. This is a problem in an internetwork connected to the Internet because a route flap in the Internet backbone usually involves many routes.

Note No penalty is applied to a BGP peer reset when route dampening is enabled. Although the reset withdraws the route, there is no penalty applied in this instance, even though route flap dampening is enabled.

Minimize Flapping

The route dampening feature minimizes the flapping problem as follows. Suppose again that the route to network A flaps. The router in AS 2 (where route dampening is enabled) assigns network A a penalty of 1000 and moves it to “history” state. The router in AS 2 continues to advertise the status of the route to neighbors. The penalties are cumulative. When the route flaps so often that the penalty exceeds a configurable suppress limit, the router stops advertising the route to network A, regardless of how many times it flaps. Thus, the route is dampened.

The penalty placed on network A is decayed until the reuse limit is reached, upon which the route is once again advertised. At half of the reuse limit, the dampening information for the route to network A is removed.

Understand Route Dampening Terms

The following terms are used when describing route dampening:

- Flap—A route is available, then unavailable, or vice versa.
- History state—After a route flaps once, it is assigned a penalty and put into “history state,” meaning the router does not have the best path, based on historical information.
- Penalty—Each time a route flaps, the router configured for route dampening in another AS assigns the route a penalty of 1000. Penalties are cumulative. The penalty for the route is stored in the BGP routing table until the penalty exceeds the suppress limit. At that point, the route state changes from “history” to “damp.”
- Damp state—In this state, the route has flapped so often that the router will not advertise this route to BGP neighbors.
- Suppress limit—A route is suppressed when its penalty exceeds this limit. The default value is 2000.
- Half-life—Once the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default). The process of reducing the penalty happens every 5 seconds.
- Reuse limit—As the penalty for a flapping route decreases and falls below this reuse limit, the route is unsuppressed. That is, the route is added back to the BGP table and once again used for forwarding. The default reuse limit is 750. The process of unsuppressing routes occurs at 10-second increments. Every 10 seconds, the router finds out which routes are now unsuppressed and advertises them to the world.
- Maximum suppress limit—This value is the maximum amount of time a route can be suppressed. The default value is 4 times the half-life.

The routes external to an AS learned via IBGP are not dampened. This policy prevent the IBGP peers from having a higher penalty for routes external to the AS.

Enable Route Dampening

To enable BGP route dampening, perform the following task in global configuration mode:

Task	**Command**
Enable BGP route dampening.	**bgp dampening**

To change the default values of various dampening factors, perform the following task in global configuration mode:

Task	**Command**
Change the default values of route dampening factors.	**bgp dampening** *half-life reuse suppress max-suppress* [**route-map** *map*]

Monitor and Maintain BGP Route Dampening

You can monitor the flaps of all the paths that are flapping. The statistics will be deleted once the route is not suppressed and is stable for at least one half-life. To display flap statistics, perform the following tasks in EXEC mode:

Task	**Command**
Display BGP flap statistics for all paths.	**show ip bgp flap-statistics**
Display BGP flap statistics for all paths that match the regular expression.	**show ip bgp flap-statistics regexp** *regexp*
Display BGP flap statistics for all paths that pass the filter.	**show ip bgp flap-statistics filter-list** *list*
Display BGP flap statistics for a single entry.	**show ip bgp flap-statistics** *address mask*
Display BGP flap statistics for more specific entries.	**show ip bgp flap-statistics** *address mask* **longer-prefix**

To clear BGP flap statistics (thus making it less likely that the route will be dampened), perform the following tasks in EXEC mode:

Task	**Command**
Clear BGP flap statistics for all routes.	**clear ip bgp flap-statistics**
Clear BGP flap statistics for all paths that match the regular expression.	**clear ip bgp flap-statistics regexp** *regexp*
Clear BGP flap statistics for all paths that pass the filter.	**clear ip bgp flap-statistics filter-list** *list*
Clear BGP flap statistics for a single entry.	**clear ip bgp flap-statistics** *address mask*
Clear BGP flap statistics for all paths from a neighbor.	**clear ip bgp** *address* **flap-statistics**

Note The flap statistics for a route are also cleared when a BGP peer is reset. Although the reset withdraws the route, there is no penalty applied in this instance, even though route flap dampening is enabled.

Once a route is dampened, you can display BGP route dampening information, including the time remaining before the dampened routes will be unsuppressed. To display the information, perform the following task in EXEC mode:

Task	Command
Display the dampened routes, including the time remaining before they will be unsuppressed.	**show ip bgp dampened-paths**

You can clear BGP route dampening information and unsuppress any suppressed routes by performing the following task in EXEC mode:

Task	Command
Clear route dampening information and unsuppress the suppressed routes.	**clear ip bgp dampening** [*address mask*]

Monitor and Maintain BGP

You can remove all contents of a particular cache, table, or database. You also can display specific statistics. The following sections describe each of these tasks.

Clear Caches, Tables, and Databases

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

The following table lists the tasks associated with clearing caches, tables, and databases for BGP. Perform these tasks in EXEC mode:

Task	Command
Reset a particular BGP connection.	**clear ip bgp** *address*
Reset all BGP connections.	**clear ip bgp ***
Remove all members of a BGP peer group.	**clear ip bgp peer-group** *tag*

Display System and Network Statistics

You can display specific statistics such as the contents of BGP routing tables, caches, and databases. Information provided can be used to determine 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.

To display various routing statistics, perform the following tasks in EXEC mode:

Task	Command
Display all BGP routes that contain subnet and supernet network masks.	**show ip bgp cidr-only**
Display routes that belong to the specified communities.	**show ip bgp community** *community-number* [**exact**]
Display routes that are permitted by the community list.	**show ip bgp community-list** *community-list-number* [**exact**]

Task	Command
Display routes that are matched by the specified autonomous system path access list.	**show ip bgp filter-list** *access-list-number*
Display the routes with inconsistent originating autonomous systems.	**show ip bgp inconsistent-as**
Display the routes that match the specified regular expression entered on the command line.	**show ip bgp regexp** *regular-expression*
Display the contents of the BGP routing table.	**show ip bgp** [*network*] [*network-mask*] [**subnets**]
Display detailed information on the TCP and BGP connections to individual neighbors.	**show ip bgp neighbors** [*address*]
Display routes learned from a particular BGP neighbor.	**show ip bgp neighbors** [*address*] [**received-routes** | **routes** | **advertised-routes** | **paths** *regular-expression* | **dampened-routes**]
Display all BGP paths in the database.	**show ip bgp paths**
Display information about BGP peer groups.	**show ip bgp peer-group** [*tag*] [**summary**]
Display the status of all BGP connections.	**show ip bgp summary**

Log Changes in Neighbor Status

To enable the logging of messages generated when a BGP neighbor resets, comes up, or goes down, use the following command in router configuration mode:

Command	Purpose
bgp log-neighbor-changes	Log messages generated when a BGP neighbor goes up or down, or resets

BGP Configuration Examples

The following sections provide BGP configuration examples:

- BGP Route Map Examples
- BGP Neighbor Configuration Examples
- BGP Synchronization Example
- BGP Path Filtering by Neighbor Example
- BGP Aggregate Route Examples
- BGP Confederation Example
- TCP MD5 Authentication for BGP Example
- BGP Peer Group Examples
- BGP Community with Route Maps Examples

BGP Route Map Examples

The following example shows how you can use route maps to modify incoming data from a neighbor. Any route received from 140.222.1.1 that matches the filter parameters set in autonomous system access list 200 will have its weight set to 200 and its local preference set to 250, and it will be accepted.

```
router bgp 100
!
 neighbor 140.222.1.1 route-map fix-weight in
 neighbor 140.222.1.1 remote-as 1
!
route-map fix-weight permit 10
 match as-path 200
 set local-preference 250
 set weight 200
!
ip as-path access-list 200 permit ^690$
ip as-path access-list 200 permit ^1800
```

In the following example, route map *freddy* marks all paths originating from autonomous system 690 with a Multi Exit Discriminator (MED) metric attribute of 127. The second permit clause is required so that routes not matching autonomous system path list 1 will still be sent to neighbor 1.1.1.1.

```
router bgp 100
 neighbor 1.1.1.1 route-map freddy out
!
ip as-path access-list 1 permit ^690_
ip as-path access-list 2 permit .*
!
route-map freddy permit 10
 match as-path 1
 set metric 127
!
route-map freddy permit 20
 match as-path 2
```

The following example shows how you can use route maps to modify incoming data from the IP forwarding table:

```
router bgp 100
 redistribute igrp 109 route-map igrp2bgp
!
route-map igrp2bgp
 match ip address 1
 set local-preference 25
 set metric 127
 set weight 30000
 set next-hop 192.92.68.24
 set origin igp
!
access-list 1 permit 131.108.0.0 0.0.255.255
access-list 1 permit 160.89.0.0 0.0.255.255
access-list 1 permit 198.112.0.0 0.0.127.255
```

It is proper behavior to not accept any autonomous system path not matching the **match** clause of the route map. This means that you will not set the metric and the Cisco IOS software will not accept the route. However, you can configure the software to accept autonomous system paths not matched in the **match** clause of the route map command by using multiple maps of the same name, some without accompanying **set** commands.

```
route-map fnord permit 10
 match as-path 1
 set local-preference 5
!
route-map fnord permit 20
 match as-path 2
```

The following example shows how you can use route maps in a reverse operation to set the route tag (as defined by the BGP/OSPF interaction document, RFC 1403) when exporting routes from BGP into the main IP routing table:

```
router bgp 100
 table-map set_ospf_tag
!
route-map set_ospf_tag
 match as-path 1
 set automatic-tag
!
ip as-path access-list 1 permit .*
```

In the following example, the route map called *set-as-path* is applied to outbound updates to the neighbor 200.69.232.70. The route map will prepend the autonomous system path “100 100” to routes that pass access list 1. The second part of the route map is to permit the advertisement of other routes.

```
router bgp 100
 network 171.60.0.0
 network 172.60.0.0
 neighbor 200.69.232.70 remote-as 200
 neighbor 200.69.232.70 route-map set-as-path out
!
route-map set-as-path 10 permit
 match address 1
 set as-path prepend 100 100
!
route-map set-as-path 20 permit
 match address 2
!
access-list 1 permit 171.60.0.0 0.0.255.255
access-list 1 permit 172.60.0.0 0.0.255.255
!
access-list 2 permit 0.0.0.0 255.255.255.255
```

Inbound route-maps could do prefix-based matching and set various parameters of the update. Inbound prefix matching is available in addition to as-path and community-list matching. In the following example, the **set local preference** command sets the local preference of the inbound prefix 140.10.0.0/16 to 120.

```
!
router bgp 100
 network 131.108.0.0
 neighbor 131.108.1.1 remote-as 200
 neighbor 131.108.1.1 route-map set-local-pref in !
route-map set-local-pref permit 10
 match ip address 2
 set local preference 120
!
route-map set-local-pref permit 20
!
access-list 2 permit 140.10.0.0 0.0.255.255 access-list 2 deny any
```

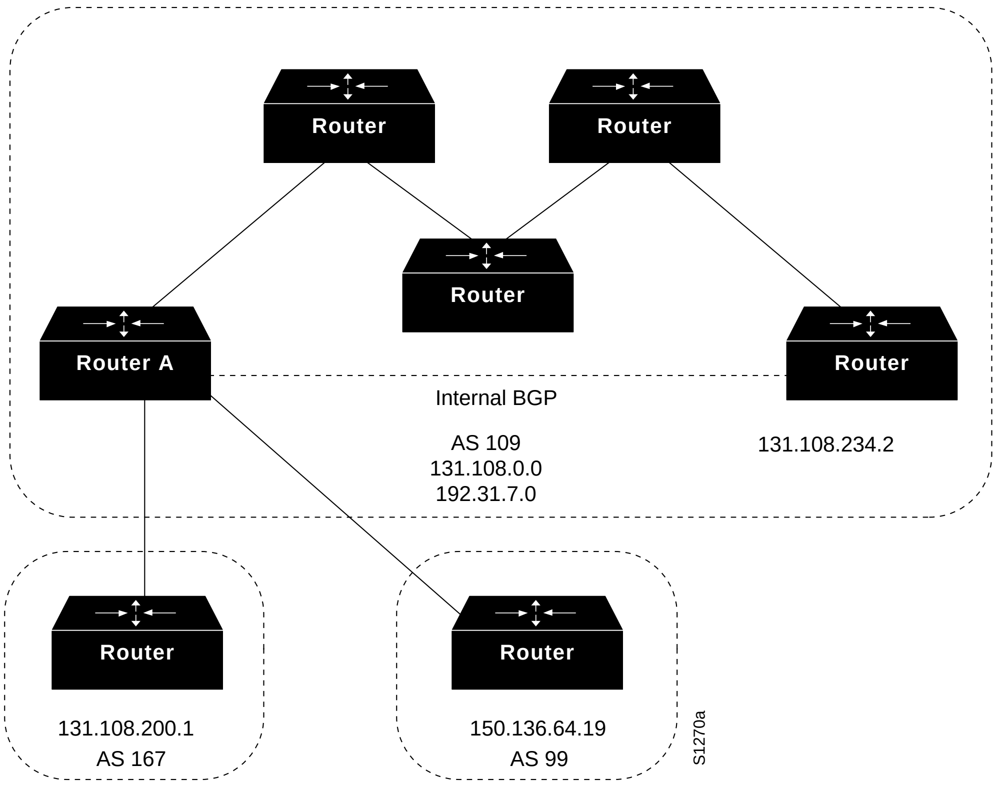
BGP Neighbor Configuration Examples

In the following example, a BGP router is assigned to autonomous system 109, and two networks are listed as originating in the autonomous system. Then the addresses of three remote routers (and their autonomous systems) are listed. The router being configured will share information about networks 131.108.0.0 and 192.31.7.0 with the neighbor routers. The first router listed is in a different autonomous system; the second **neighbor** command specifies an internal neighbor (with the same autonomous system number) at address 131.108.234.2; and the third **neighbor** command specifies a neighbor on a different autonomous system.

```
router bgp 109
 network 131.108.0.0
 network 192.31.7.0
 neighbor 131.108.200.1 remote-as 167
 neighbor 131.108.234.2 remote-as 109
 neighbor 150.136.64.19 remote-as 99
```

In Figure 28, Router A is being configured. The internal BGP neighbor is not directly linked to Router A. External neighbors (in autonomous system 167 and autonomous system 99) must be linked directly to Router A.

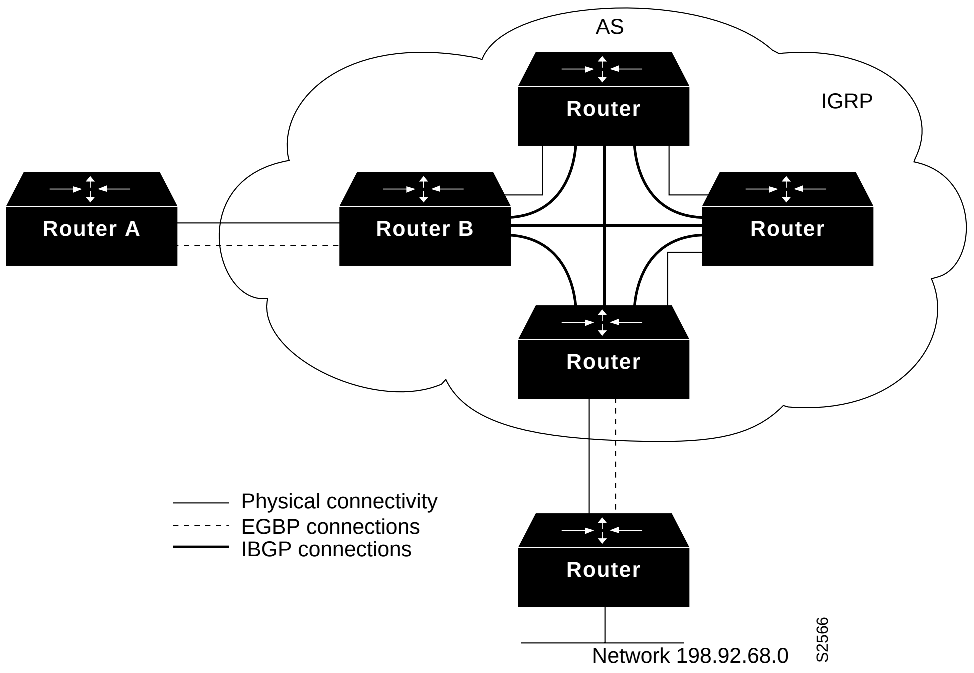
Figure 28 Assigning Internal and External BGP Neighbors



BGP Synchronization Example

In the configuration shown in Figure 29, with synchronization on, Router B does not advertise network 198.92.68.0 to Router A until an IGRP route for network 198.92.68.0 exists. If you specify the **no synchronization** router configuration command, Router B advertises network 198.92.68.0 as soon as possible. However, because routing information still must be sent to interior peers, you must configure a full internal BGP mesh.

Figure 29 BGP Synchronization Configuration



BGP Path Filtering by Neighbor Example

The following is an example of BGP path filtering by neighbor. The routes that pass as-path access list 1 will get weight 100. Only the routes that pass as-path access list 2 will be sent to 193.1.12.10. Similarly, only routes passing access list 3 will be accepted from 193.1.12.10.

```
router bgp 200
 neighbor 193.1.12.10 remote-as 100
 neighbor 193.1.12.10 filter-list 1 weight 100
 neighbor 193.1.12.10 filter-list 2 out
 neighbor 193.1.12.10 filter-list 3 in
ip as-path access-list 1 permit _109_
ip as-path access-list 2 permit _200$
ip as-path access-list 2 permit ^100$
ip as-path access-list 3 deny _690$
ip as-path access-list 3 permit .*
```

BGP Aggregate Route Examples

The following examples show how you can use aggregate routes in BGP either by redistributing an aggregate route into BGP or by using the conditional aggregate routing feature.

In the following example, the **redistribute static** command is used to redistribute aggregate route 193.*.*.*:

```
ip route 193.0.0.0 255.0.0.0 null 0
!
router bgp 100
 redistribute static
```

The following configuration creates an aggregate entry in the BGP routing table when there is at least one specific route that falls into the specified range. The aggregate route will be advertised as coming from your autonomous system and has the atomic aggregate attribute set to show that information might be missing. (By default, atomic aggregate is set unless you use the **as-set** keyword in the **aggregate-address** command.)

```
router bgp 100
 aggregate-address 193.0.0.0 255.0.0.0
```

The following example creates an aggregate entry using the same rules as in the previous example, but the path advertised for this route will be an AS_SET consisting of all elements contained in all paths that are being summarized:

```
router bgp 100
 aggregate-address 193.0.0.0 255.0.0.0 as-set
```

The following example not only creates the aggregate route for 193.*.*.*, but will also suppress advertisements of more specific routes to all neighbors:

```
router bgp 100
 aggregate-address 193.0.0.0 255.0.0.0 summary-only
```

BGP Confederation Example

The following is a sample configuration from several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 666 (specified via the **bgp confederation identifier** command).

In a BGP speaker in autonomous system 6001, the **bgp confederation peers** command marks the peers from autonomous systems 6002 and 6003 as special EBGP peers. Hence peers 171.69.232.55 and 171.69.232.56 will get the local-preference, next-hop and MED unmodified in the updates. The router at 160.69.69.1 is a normal EBGP speaker and the updates received by it from this peer will be just like a normal EBGP update from a peer in autonomous system 666.

```
router bgp 6001
 bgp confederation identifier 666
 bgp confederation peers 6002 6003
 neighbor 171.69.232.55 remote-as 6002
 neighbor 171.69.232.56 remote-as 6003
 neighbor 160.69.69.1 remote-as 777
```

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special EBGP peers. 170.70.70.1 is a normal IBGP peer and 199.99.99.2 is a normal EBGP peer from autonomous system 700.

```
router bgp 6002
 bgp confederation identifier 666
 bgp confederation peers 6001 6003
 neighbor 170.70.70.1 remote-as 6002
 neighbor 171.69.232.57 remote-as 6001
 neighbor 171.69.232.56 remote-as 6003
 neighbor 199.99.99.2 remote-as 700
```

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special EBGP peers. 200.200.200.200.200 is a normal EBGP peer from autonomous system 701.

```
router bgp 6003
 bgp confederation identifier 666
```

```
    bgp confederation peers 6001 6002
    neighbor 171.69.232.57 remote-as 6001
    neighbor 171.69.232.55 remote-as 6002
    neighbor 200.200.200.200 remote-as 701
```

The following is a part of the configuration from the BGP speaker 200.200.200.205 from autonomous system 701. Neighbor 171.69.232.56 is configured as a normal EBGP speaker from autonomous system 666. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```
router bgp 701
 neighbor 171.69.232.56 remote-as 666
 neighbor 200.200.200.205 remote-as 701
```

TCP MD5 Authentication for BGP Example

The following example specifies that the router and its BGP peer at 145.2.2.2 invoke MD5 authentication on the TCP connection between them:

```
router bgp 109
 neighbor 145.2.2.2 password v61ne0qkel33&
```

BGP Peer Group Examples

This section contains an IBGP peer group example and an EBGP peer group example.

IBGP Peer Group Example

In the following example, the peer group named *internal* configures the members of the peer group to be IBGP neighbors. By definition, this is an IBGP peer group because the **router bgp** command and the **neighbor remote-as** command indicate the same autonomous system (in this case, AS 100). All the peer group members use loopback 0 as the update source and use *set-med* as the outbound route-map. The example also shows that, except for the neighbor at address 171.69.232.55, all the neighbors have filter-list 2 as the inbound filter list.

```
router bgp 100
 neighbor internal peer-group
 neighbor internal remote-as 100
 neighbor internal update-source loopback 0
 neighbor internal route-map set-med out
 neighbor internal filter-list 1 out
 neighbor internal filter-list 2 in
 neighbor 171.69.232.53 peer-group internal
 neighbor 171.69.232.54 peer-group internal
 neighbor 171.69.232.55 peer-group internal
 neighbor 171.69.232.55 filter-list 3 in
```

EBGP Peer Group Example

In the following example, the peer group *external-peers* is defined without the **neighbor remote-as** command. This is what makes it an EBGP peer group. Each member of the peer group is configured with its respective autonomous system number separately. Thus, the peer group consists of members from autonomous systems 200, 300, and 400. All the peer group members have *set-metric* route map as an outbound route map and filter-list 99 as an outbound filter list. Except for neighbor 171.69.232.110, all have 101 as the inbound filter list.

```
router bgp 100
 neighbor external-peers peer-group
 neighbor external-peers route-map set-metric out
 neighbor external-peers filter-list 99 out
 neighbor external-peers filter-list 101 in
 neighbor 171.69.232.90 remote-as 200
 neighbor 171.69.232.90 peer-group external-peers
 neighbor 171.69.232.100 remote-as 300
 neighbor 171.69.232.100 peer-group external-peers
 neighbor 171.69.232.110 remote-as 400
 neighbor 171.69.232.110 peer-group external-peers
 neighbor 171.69.232.110 filter-list 400 in
```

BGP Community with Route Maps Examples

This section contains three examples of the use of BGP communities with route maps.

In the first example, the route map *set-community* is applied to the outbound updates to the neighbor 171.69.232.50. The routes that pass access list 1 have the special community attribute value “no-export.” The remaining routes are advertised normally. This special community value automatically prevents the advertisement of those routes by the BGP speakers in autonomous system 200.

```
router bgp 100
 neighbor 171.69.232.50 remote-as 200
 neighbor 171.69.232.50 send-community
 neighbor 171.69.232.50 route-map set-community out
!
route-map set-community 10 permit
 match address 1
 set community no-export
!
route-map set-community 20 permit
 match address 2
```

In the second example, the route map *set-community* is applied to the outbound updates to neighbor 171.69.232.90. All the routes that originate from AS 70 have the community values 200 200 added to their already existing values. All other routes are advertised as normal.

```
route-map bgp 200
 neighbor 171.69.232.90 remote-as 100
 neighbor 171.69.232.90 send-community
 neighbor 171.69.232.90 route-map set-community out
!
route-map set-community 10 permit
 match as-path 1
 set community 200 200 additive
!
route-map set-community 20 permit
 match as-path 2
!
ip as-path access-list 1 permit 70$
ip as-path access-list 2 permit .*
```

In the third example, community-based matching is used to selectively set MED and local-preference for routes from neighbor 171.69.232.55. All the routes that match community list 1 get the MED set to 8000. This includes any routes that have the communities “100 200 300” or “900 901.” These routes could have other community values also.

All the routes that pass community list 2 get the local preference set to 500. This includes the routes that have community values 88 or 90. If they belong to any other community, they will not be matched by community list 2.

All the routes that match community list 3 get the local-preference set to 50. Community list 3 will match all the routes because all the routes are members of the Internet community. Thus, all the remaining routes from neighbor 171.69.232.55 get a local preference 50.

```
router bgp 200
 neighbor 171.69.232.55 remote-as 100
 neighbor 171.69.232.55 route-map filter-on-community in
!
route-map filter-on-community 10 permit
 match community 1
 set metric 8000
!
route-map filter-on-community 20 permit
 match community 2 exact-match
 set local-preference 500
!
route-map filter-on-community 30 permit
 match community 3
 set local-preference 50
!
ip community-list 1 permit 100 200 300
ip community-list 1 permit 900 901
!
ip community-list 2 permit 88
ip community-list 2 permit 90
!
ip community-list 3 permit internet
```