Chapter 1

Configuring BGP Routing

This chapter contains the following sections:

Overview

The Border Gateway Protocol (BGP) provides loop-free interdomain routing between autonomous systems (ASs). This section describes some of the main concepts of BGP.

Conventions in This Chapter

Certain terms used with BGP, such as the names of attributes and messages, are typically expressed in all uppercase letters in the RFCs. For improved readability, those terms are represented in lowercase in this chapter. Table 4 lists the terms and their variant spellings.

Table 4: Conventions for BGP Terms

In This Chapter	In RFCs
aggregator	AGGREGATOR
AS-confed-set	AS_CONFED_SET
AS-path or AS path	AS_PATH
AS-sequence	AS_SEQUENCE
AS-set	AS_SET
atomic-aggregate	ATOMIC_AGGREGATE
cluster-list	CLUSTER_LIST
keepalive	KEEPALIVE
local-pref	LOCAL_PREF
multiexit discriminator or MED	MULTI_EXIT_DISC
new-as-path	NEW_AS_PATH
new-aggregator	NEW_AGGREGATOR
next-hop or next hop	NEXT_HOP
no-advertise	NO_ADVERTISE
no-export	NO_EXPORT
no-export-subconfed	NO_EXPORT_SUBCONFED
notification	NOTIFICATION
open	OPEN
origin	ORIGIN
originator-ID	ORIGINATOR_ID
route-refresh	ROUTE-REFRESH
update	UPDATE

Autonomous Systems

An autonomous system (AS) is a set of routers that use the same routing policy while running under a single technical administration. An AS runs interior gateway protocols (IGPs) such as RIP, OSPF, and IS-IS within its boundaries. ASs use exterior gateway protocols (EGPs) to exchange routing information with other ASs. BGP is an EGP.

The outside world views an AS as a single entity, even though it can be a collection of IGPs working together to provide routing within its interior.

Each AS has an identification number provided by an Internet registry or by an Internet service provider (ISP) that uniquely identifies it to the outside world.

BGP Speaker

A router that has been configured to run the BGP routing protocol is called a BGP speaker.

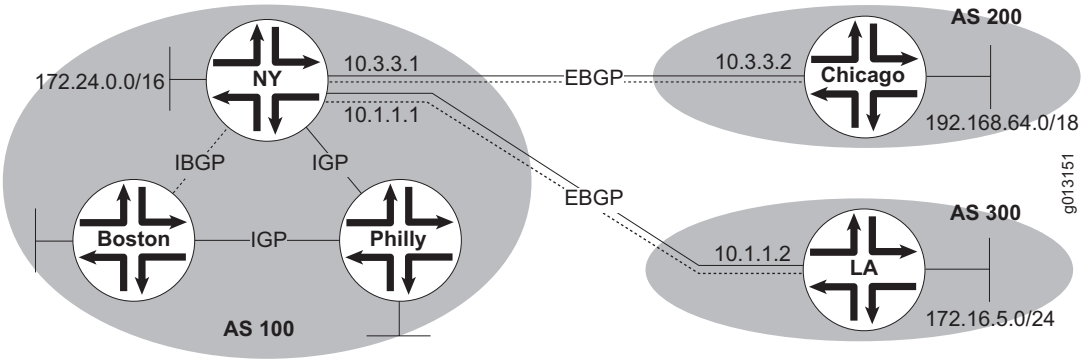
BGP Peers and Neighbors

Unlike some other routing protocols, BGP speakers do not automatically discover each other and begin exchanging information. Instead, each BGP speaker must be explicitly configured with a set of BGP peers with which it exchanges routing information. BGP peers do not have to be directly connected to each other in order to share a BGP session. Another term for BGP peer is BGP neighbor. A BGP *peer group* consists of two or more BGP peers that share a common set of update policies.

In Figure 1, router NY and router Chicago are peers. Router NY and router LA are peers. Router NY and router Boston are peers. Router NY and router Philly are not peers. Router Chicago and router LA are not peers.

NOTE: The figures in this chapter indicate a BGP session with a dotted line. A physical link is represented by a solid line.

Figure 1: BGP Peers



BGP Session

When two BGP speakers have both been configured to be BGP peers of each other, they will establish a BGP session to exchange routing information. A BGP session is simply a TCP connection over which routing information is exchanged according to the rules of the BGP protocol.

Because BGP relies on TCP to provide reliable and flow-controlled transmission of routing information, the BGP protocol itself is very simple. However it also implies that two routers can be BGP peers of each other only if they are reachable from each other in the sense that they can exchange IP packets.

In practice this means that either of the following must be true:

- The BGP peers must be connected to a common IP subnet.
- The BGP peers must be in the same AS, which runs an IGP enabling the BGP peers to reach each other.

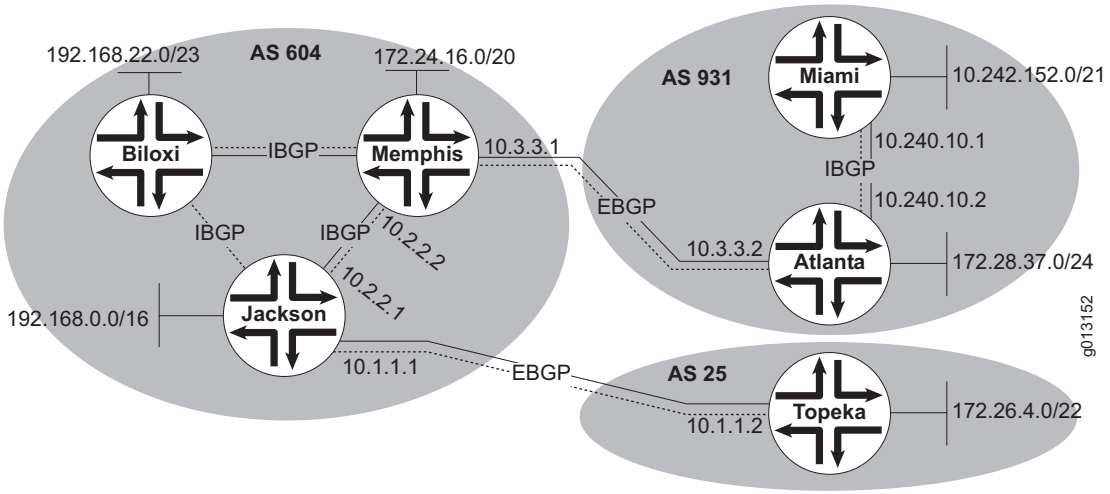
IBGP and EBGP

When two BGP speakers are in the same autonomous system, the BGP session is called an *internal* BGP session, or IBGP session. When two BGP speakers are in different autonomous systems, the BGP session is called an *external* BGP session, or EBGP session. BGP uses the same types of message on IBGP and EBGP sessions, but the rules for when to send which message and how to interpret each message differ slightly; for this reason some people refer to IBGP and EBGP as two separate protocols.

IBGP requires that BGP speakers within an autonomous system be fully meshed, meaning that there must be a BGP session between each pair of peers within the AS. IBGP does not require that all the peers be physically connected. EBGP does not require full meshing of BGP speakers. EBGP sessions typically exist between peers that are physically connected.

Figure 2 shows an example of the exchange of information between routers running IBGP and EBGP across multiple ASs.

Figure 2: Internal and External BGP



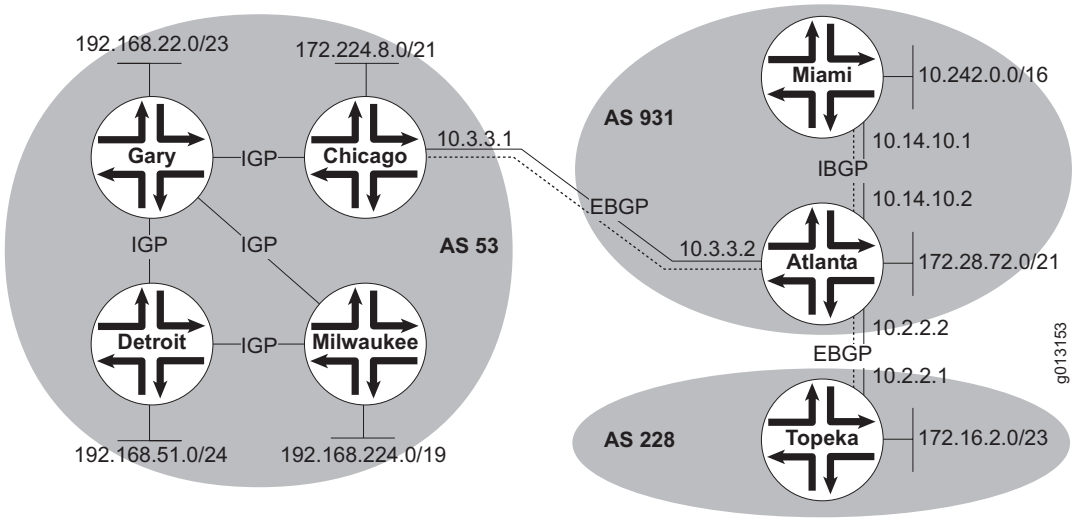
Interior Gateway Protocols

Not all the routers within an AS have to be BGP peers. For example, in some large enterprise networks, ASs generally have many more non-BGP routers. These routers communicate using an interior gateway protocol (IGP) such as the following:

- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Routing Information Protocol (RIP)

Figure 3 shows that the routers in AS 53 all communicate with each other using an IGP. Routing information internal to AS 53 is redistributed from the IGP into BGP at router Chicago. Router Chicago redistributes into the IGP the routing information it receives from its external BGP peer, router Atlanta. Router Atlanta has an internal BGP link within its AS, and an external BGP link to router Topeka.

Figure 3: Interior Gateway Protocols



BGP Messages

BGP speakers exchange routing information with each other by exchanging BGP messages over a BGP session. BGP uses the following five message types:

- Open BGP messages—When two BGP speakers establish a BGP session with each other, the first message they exchange after the underlying TCP session has been established is an *open message*. This message contains various bits of information that enable the two BGP peers to determine whether they want to establish a BGP session with each other—for example, the AS number of the BGP speaker—and to negotiate certain parameters for the BGP session—for example, how often to send a keepalive message.

- Update messages—The update message is the most important message in the BGP protocol. A BGP speaker sends update messages to announce routes to prefixes that it can reach and to withdraw routes to prefixes that it can no longer reach.

- Keepalive messages—BGP speakers periodically exchange keepalive messages to determine whether the underlying TCP connection is still up.

- Notification messages—If a BGP speaker wishes to terminate a BGP session (either because it has been configured to do so or because it has detected some error condition), it will send a notification message to its peer specifying the reason for terminating the BGP session.

  If the session is being terminated for a nonfatal error, the notification messages includes the error code cease. Subcodes sent in the notification message can inform network operators about peering problems and help them better understand network events. Table 5 lists the subcodes defined for BGP notification messages bearing the cease code.

Table 5: Cease Notification Message Subcodes

Subcode	Reason	Symbolic Name
1	The number of address prefixes received from the peer has exceeded the upper bound configured with the **neighbor maximum-prefix** command. The notification message can include address family and upper bound information in the data field.	Maximum Number of Prefixes Reached
2	The BGP speaker is administratively shutting down the session.	Administratively Shutdown
3	The BGP speaker is removing the peer configuration.	Peer Unconfigured
4	The BGP speaker is administratively resetting the session.	Administratively Reset
5	The BGP speaker is rejecting the connection (for example, because the peer is not configured locally on the speaker) after accepting a transport protocol connection.	Connection Rejected
6	The BGP speaker is administratively resetting the session for some other configuration.	Other Configuration Change

- Route-refresh messages—BGP speakers can send route-refresh messages to peers that advertise the route-refresh capability. The messages contain a request for the peer to resend its routes to the router. This feature enables the BGP speaker to apply modified or new policies to the routes when it receives them again.

BGP Route

A *BGP route* consists of two parts, a prefix and a set of path attributes. It is not uncommon to use the term *path* to refer to a BGP route, although that term technically refers to one of the path attributes of that route.

Routing Information Base

BGP routes are stored in a BGP speaker's routing information base (RIB), also known as its routing table, which conceptually consists of the following three parts:

- Adj-RIBs-In store unprocessed routes learned from update messages received by the BGP speaker.
- Loc-RIB contains local routes resulting from the BGP speaker applying its local policies to the routes contained in its Adj-RIBs-In.
- Adj-RIBs-Out store routes that the BGP speaker will advertise to its peers in the update messages it sends.

Prefixes and CIDR

A *prefix* describes a set of IP addresses that can be reached using the route. For example, the prefix 10.1.1.0/24 indicates all IP addresses whose first 24 bits contain the value 10.1.1. The term *network* is sometimes used instead of *prefix* to describe a set of addresses. To reduce confusion, this chapter restricts *network* to its more common usage, to refer to a physical structure of routers and links.

Prefixes are made possible by classless interdomain routing (CIDR). CIDR addresses have largely replaced the concept of classful addresses (such as Class A, Class B, and Class C) in the Internet. Classful addresses have an implicit, fixed-length mask corresponding to the predefined class boundaries. For example, 192.56.0.0 is a Class B address with an implicit (or natural) mask of 255.255.0.0.

CIDR uses network prefixes and explicit masks, represented by a prefix length, enabling network prefixes of arbitrary lengths. CIDR represents the sample address above as 192.56.0.0/16. The /16 indicates that the high-order 16 bits (the first 16 bits counting from left to right) in the address mask are all 1s.

CIDR enables you to aggregate multiple classful addresses into a single classless advertisement, reducing the number of advertisements that must be made to provide full access to all the addresses. Suppose an ISP has customers with the following addresses:

192.168.128.0

192.168.129.0

192.168.130.0

192.168.131.0

192.168.132.0

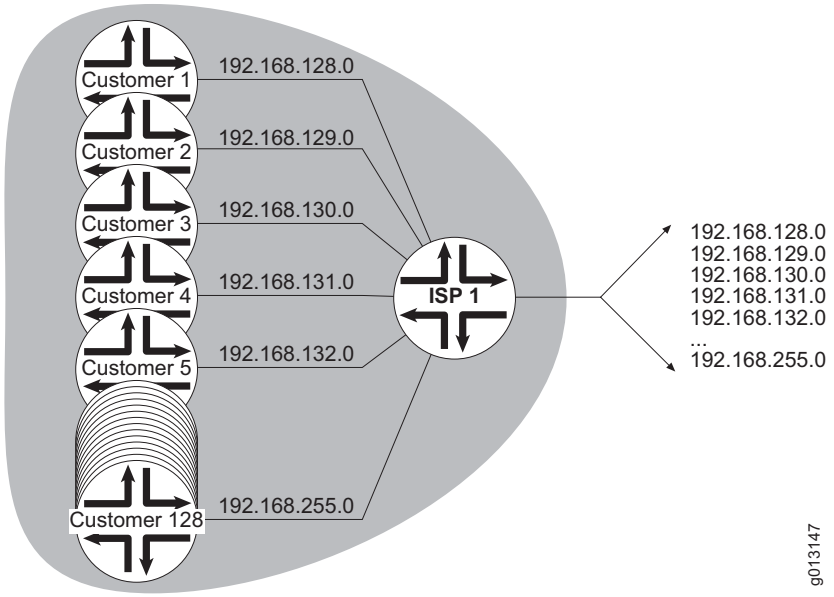
192.168.133.0

...

192.168.255.0

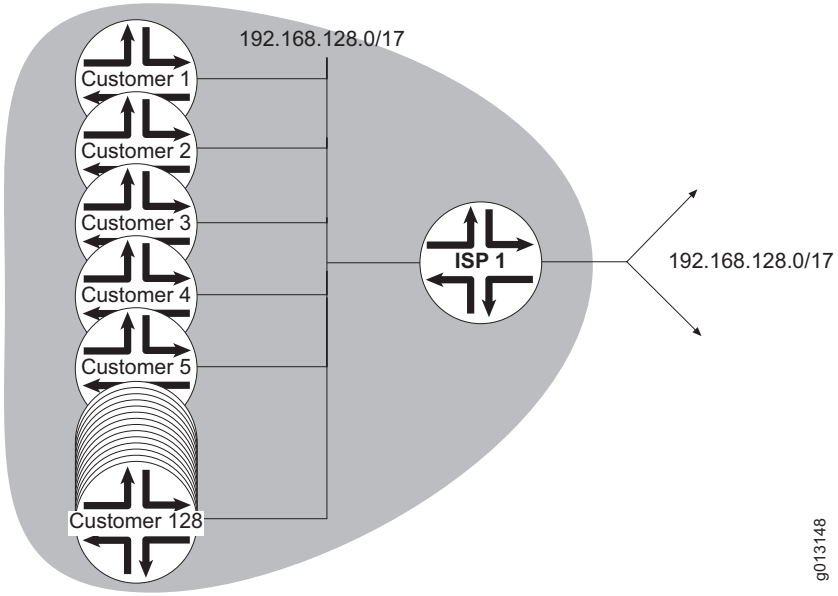
Without CIDR, the ISP has to advertise a route to each address, as shown in Figure 4.

Figure 4: Routing Without CIDR



With CIDR, the ISP can aggregate the routes as 192.168.128.0/17 and advertise a single address to that prefix, as shown in Figure 5.

Figure 5: Routing with CIDR



Path Attributes

A path attribute provides some additional information about a route. If a BGP speaker has more than one route to the same destination prefix, it selects one of those routes to use (the “best” route) based on the path attributes. BGP as implemented on the E-series router specifies detailed and complex criteria for picking the best route; this helps ensure that all routers will converge to the same routing table, a necessary behavior to avoid routing loops. See *Selecting the Best Path* on page 103 for more information.

The following are some of the most important path attributes:

- *AS-path* specifies the sequence of autonomous systems that must be crossed to reach a certain destination. This path attribute is used to avoid routing loops and to prefer shorter routes over longer routes.

- *Next-hop* specifies the IP address of the ingress router in the next autonomous system on the path to the destination.

- *Local-pref* and *multiexit discriminator* (MED) are metrics that administrators can tune to ensure that certain routes are more attractive over other routes. The local-pref attribute specifies a degree of preference that enables a router to select among multiple routes to the same prefix. The MED is used for ASs that have more than one connection to each other. The administrator of one AS sets the MED to express a degree of preference for one link versus another; the BGP peer in the other AS uses this MED to optimize traffic.

- *Originator-ID* specifies the IP address of the router that originates the route. The router ignores updates that have this attribute set to its own IP address.

- *Atomic-aggregate* and *aggregator* inform peers about actions taken by a BGP speaker regarding aggregation of routes. If a BGP speaker aggregates routes that have differing path attributes, it includes the atomic-aggregate attribute with the aggregated prefix to inform update recipients that they must not deaggregate the prefix. A BGP speaker aggregating routes can include the aggregator attribute to indicate the router and AS where the aggregation was performed.

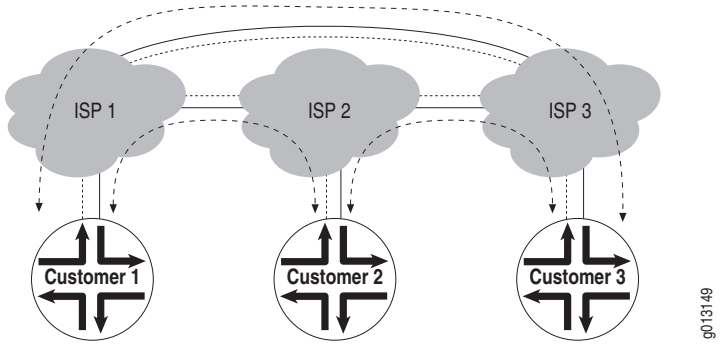
- *Community* and *extended community* identify prefixes as sharing some common attribute, providing a means of grouping prefixes and enacting routing policies on the group of prefixes. A prefix can belong to more than one community. You can specify a community name as a 32-bit string, a standards-defined well-known community, or an AS number combined with a 32-bit number to create a unique identifier. An extended community name consists of either an IP address or an AS number, combined with a 32-bit or 16-bit number to create a unique identifier.

Transit and Nontransit Service

While an ISP provides connectivity to its customers, it also provides connectivity to customers of other ISPs. In doing this, an ISP must be able to ensure the appropriate use of its resources.

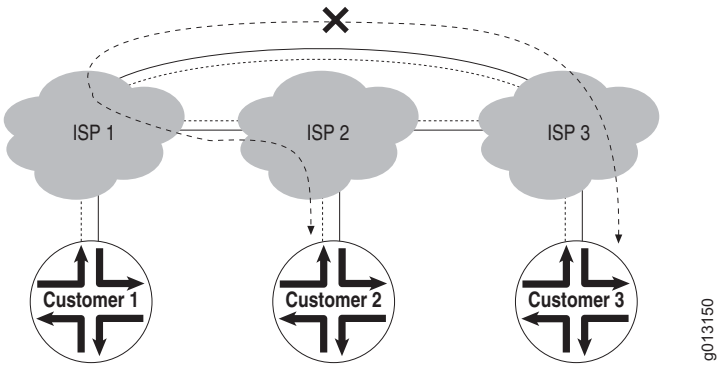
For example, Figure 6 shows three ISPs and three customers. ISP 1, ISP 2, and ISP 3 are directly connected to one another through a physical link and a corresponding EBGP session (represented here by a single line). Customer 1 is connected to ISP 1 through a physical link and corresponding EBGP session. Customer 2 is similarly connected to ISP 2, and Customer 3 is similarly connected to ISP 3. Each ISP provides *transit* service to its own customers. Figure 6 illustrates how the ISP permits traffic to transit across its backbone from its own customers or to its own customers.

Figure 6: Transit Service



Each ISP provides *nontransit* service to other ISPs. For example, Figure 7 shows that ISP 1 does not permit traffic between ISP 2 and ISP 3 to cross its backbone. If ISP 1 permits such traffic, it squanders its own resources with no benefit to its customers or itself.

Figure 7: Nontransit Service



IPv6 BGP Support

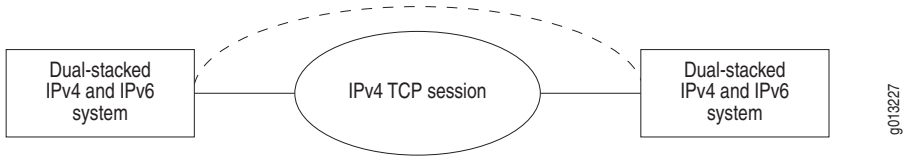
Most of the extensions and features available in BGP for IPv4 are also available for the IPv6 address family, such as policy-based routing, redistributing routes to and from other protocols, route aggregation, route flap dampening, and confederations. For a description of IPv6, see *JUNOSe IP, IPv6, and IGP Configuration Guide, Chapter 2, Configuring IPv6*.

Multiprotocol Extensions for BGP-4 (MP-BGP) allow the exchange of IPv6 routing information over TCP IPv4 (Figure 8) or TCP IPv6 transport (Figure 9).

Exchange of IPv6 Routing Information over TCP IPv4

Figure 8 illustrates the exchange of IPv6 routing information over a TCP IPv4 connection.

Figure 8: IPv6 Routing over TCP IPv4



The E-series router's MP-BGP implementation uses BGP update messages to announce the feasible routes to an associated IPv6 BGP next hop and also to announce the nonfeasible routes that need to be withdrawn from the peer. The E-series router announces only IPv6 global addresses as the BGP next-hop address; it does not use the optional link-local IPv6 address as the BGP next hop.

BGP determines the next-hop addresses to be announced by using the IPv4-compatible IPv6 address. For example, the following table shows the translation of an IPv4 address.

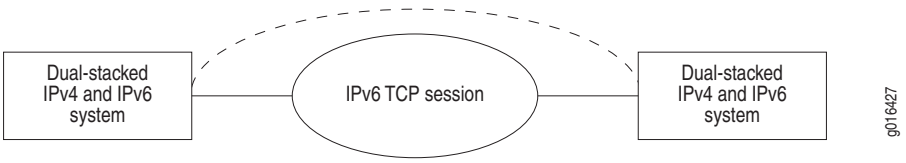
IPv4 address	IPv6 address
10.1.1.1	::10.1.1.1

When a BGP speaker receives a BGP update message carrying IPv6 feasible routes, the speaker resolves the announced IPv6 BGP next hop by performing a route lookup to the IPv6 address in the IPv6 route table.

Exchange of IPv6 Routing Information over TCP IPv6

Figure 9 illustrates the exchange of IPv6 routing information over a TCP IPv6 connection.

Figure 9: IPv6 Routing over TCP IPv6



Link-Local Next Hops in MP-BGP Packets

When the router has an external directly connected (non-multihop) BGP peer, the router advertises two next hops. It advertises the global next hop and a next hop with a link-local address. The link-local next hop is advertised even when the router has been configured with the next-hop self feature. Advertising the link-local next hop enables the configuration of single-hop EBGP sessions for IPv6 next hops.

For all other types of peers, the router advertises only the global BGP IPv6 next hop.

You can overwrite the global and link-local IPv6 next-hop addresses by configuring and applying a route map that sets the addresses. The **set ipv6 next-hop** clause in the route map can specify a global address, a link-local address, or both for the next hop.

However, a neighbor outbound route map that adds a link-local IPv6 address for peers where the router should not advertise a link-local next hop is considered an invalid configuration.

The router accepts both global and link-local BGP IPv6 next-hop addresses received from its BGP IPv6 peers. As a consequence, when advertising a route to an internal peer, the router can modify the network address of the next-hop field by removing the link-local IPv6 address of the next hop.

For static BGP peers, the JUNOSe software does not support the use of link-local addresses when you configure BGP peers. You cannot configure the local interface for a neighbor that has been configured with a link-local address. Although you can configure a neighbor with a link-local address, a BGP session to that peer over TCP IPv6 does not come up.

For dynamic BGP peers, an E-series router can accept incoming TCP sessions with the link-local address as the source address. However, the BGP peering does not come up for such a connection.

Platform Considerations

For information about modules that support BGP on the ERX-7xx models, ERX-14xx models, and the ERX-310 router:

- See *ERX Module Guide, Table 1, Module Combinations* for detailed module specifications.
- See *ERX Module Guide, Appendix A, Module Protocol Support* for information about the modules that support BGP.

For information about modules that support BGP on E120 routers and E320 routers:

- See *E120 and E320 Module Guide, Table 1, Modules and IOAs* for detailed module specifications.
- See *E120 and E320 Module Guide, Appendix A, IOA Protocol Support* for information about the modules that support BGP.

References

For more information about the BGP protocol, consult the following resources:

- Address Prefix Based Outbound Route Filter for BGP-4—draft-chen-bgp-prefix-orf-07.txt (March 2004 expiration)
- BGP Extended Communities Attribute—draft-ietf-idr-bgp-ext-communities-07.txt (February 2004 expiration)
- BGP/MPLS IP VPNs—draft-ietf-l3vpn-rfc2547bis-03.txt (April 2005 expiration)
- BGP support for four-octet AS number space—draft-ietf-idr-as4bytes-08.txt (February 2004 expiration)
- Connecting IPv6 Islands across IPv4 Clouds with BGP—draft-ietf-ngtrans-bgp-tunnel-04.txt (July 2002 expiration)
- Cooperative Route Filtering Capability for BGP-4—draft-ietf-idr-route-filter-09.txt (February 2003 expiration)
- Dynamic Capability for BGP-4—draft-ietf-idr-dynamic-cap-04.txt (February 2004 expiration)
- *JUNOSe Release Notes*, *Appendix A*, *System Maximums*—Refer to the Release Notes corresponding to your software release for information about maximum values.
- Graceful Restart Mechanism for BGP—draft-ietf-idr-restart-10.txt (March 2004 expiration)
- RFC 1657—Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2 (July 1997)
- RFC 1745—BGP4/IDRP for IP—OSPF Interaction (December 1994)
- RFC 1771—A Border Gateway Protocol 4 (BGP-4) (March 1995)
- RFC 1772—Application of the Border Gateway Protocol in the Internet (March 1995)
- RFC 1773—Experience with the BGP-4 protocol (March 1995)
- RFC 1774—BGP-4 Protocol Analysis (March 1995)
- RFC 1863—A BGP/IDRP Route Server alternative to a full mesh routing (October 1995)
- RFC 1930—Guidelines for creation, selection, and registration of an Autonomous System (AS) (March 1996)
- RFC 1965—Autonomous System Confederations for BGP (June 1996)
- RFC 1966—BGP Route Reflection An alternative to full mesh IBGP (June 1996)

- RFC 1997—BGP Communities Attribute (August 1996)
- RFC 1998—An Application of the BGP Community Attribute in Multi-home Routing (August 1996)
- RFC 2858—Multiprotocol Extensions for BGP-4 (June 2000)
- RFC 2270—Using a Dedicated AS for Sites Homed to a Single Provider (January 1998)
- RFC 2385—Protection of BGP Sessions via the TCP MD5 Signature Option (August 1998)
- RFC 2439—BGP Route Flap Damping (November 1998)
- RFC 2519—A Framework for Inter-Domain Route Aggregation (February 1999)
- RFC 2545—Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing (March 1999)
- RFC 2547—BGP/MPLS VPNs (March 1999)
- RFC 2796—BGP Route Reflection—An Alternative to Full Mesh IBGP (April 2000)
- RFC 2842—Capabilities Advertisement with BGP-4 (May 2000)
- RFC 2858—Multiprotocol Extensions for BGP-4 (June 2000)
- RFC 2918—Route Refresh Capability for BGP-4 (September 2000)
- RFC 3032—MPLS Label Stack Encoding (January 2001)
- RFC 3065—Autonomous System Confederations for BGP (February 2001)
- RFC 3392—Capabilities Advertisement with BGP-4 (November 2002)
- Subcodes for BGP Cease Notification Message—draft-ietf-idr-cease-subcode-05.txt (March 2004 expiration)

NOTE: IETF drafts are valid for only 6 months from the date of issuance. They must be considered as works in progress. Please refer to the IETF Web site at http://www.ietf.org for the latest drafts.

Features

Some of the more important BGP features supported by the E-series router are the following:

- Access lists
- Advertisement intervals
- Aggregation
- BGP/MPLS VPNs
- Communities
- Confederations
- EBGP multihop
- IBGP single hop
- Highly scalable BGP-4 architecture
- Multicast
- Next-hop self
- Peer groups
- Route dampening (also referred to as route damping)
- Route mapping and attribute manipulation
- Route origins
- Route redistribution
- Route reflectors
- Soft-reconfiguration inbound
- Synchronization enabling and disabling
- Update source

Before You Configure BGP

Before you attempt to configure BGP, ensure that you have TCP/IP reachability to the BGP peers with which you want your router to communicate. This may include tasks such as setting up interfaces and creating routes.

See the *JUNOSe Link Layer Configuration Guide* and *JUNOSe Physical Layer Configuration Guide* for information about how to configure appropriate interfaces. See *JUNOSe IP Services Configuration Guide, Chapter 1, Configuring Routing Policy,* for information about setting up routing information.

Configuration Tasks

BGP is a very flexible protocol, often providing more than one way to achieve a routing goal. The configuration tasks required therefore vary depending on your needs and decisions. Read all of the following sections to determine the best method for configuring BGP for your needs.

- Basic Configuration on page 16
- Configuring BGP Peer Groups on page 25
- Advertising Routes on page 49
- Configuring BGP Routing Policy on page 68
- Selecting the Best Path on page 103
- Interactions Between BGP and IGPs on page 130
- Detecting Peer Reachability with BFD on page 138
- Managing a Large-Scale AS on page 140
- Configuring BGP Multicasting on page 150
- Using BGP Routes for Other Protocols on page 153
- Configuring BGP/MPLS VPNs on page 154

Basic Configuration

Two tasks are common to every BGP configuration: You must enable the BGP routing process, and you must configure BGP neighbors. All other basic configuration tasks are optional.

You can configure certain BGP attributes globally, for peer groups, or for individual peers. The most specific level of configuration takes precedence. For example, if you configure an attribute both globally and for a peer group, the peer group configuration takes precedence for that peer group, but does not affect other peer groups. If you configure an attribute both for a peer group and for a peer, the peer configuration takes precedence for that peer, but does not affect other members of that peer group.

Enabling BGP Routing

All BGP configurations require that you enable the BGP routing process on one or more routers.

router bgp

- Use to enable the BGP routing protocol and to specify the local AS—the AS to which this BGP speaker belongs.
- All subsequent BGP configuration commands are placed within the context of this router and AS; you can have only a single BGP instance per virtual router.
- Specify only one BGP AS per virtual router.
- This command takes effect immediately.
- Example

  host1(config)#**router bgp 100**

- Use the **no** version to remove the BGP process.

Understanding BGP Command Scope

BGP commands can be sorted into the following categories, each of which has a different scope; that is, each configures parameters within a different area of applicability. Individual command descriptions in this chapter and in *Chapter 3, Configuring BGP-MPLS Applications*, provide more information about command behavior.

- The commands listed in Table 6 configure parameters for the BGP process globally, regardless of address family.

Table 6: Commands Affecting BGP Globally

bgp advertise-inactive	bgp graceful-restart path-selection-defer-time
bgp advertise best-external-to-internal	bgp graceful-restart restart-time
bgp always-compare-med	bgp graceful-restart stalepaths-time
bgp bestpath med confed	bgp log-neighbor-changes
bgp bestpath missing-as-worst	bgp maxas-limit
bgp client-to-client reflection	bgp redistribute-internal
bgp cluster-id	bgp router-id
bgp confederation identifier	bgp shutdown
bgp confederation peers	ip bgp-community new-format
bgp default local-preference	overload shutdown
bgp default route-target filter	rib-out disable
bgp enforce-first-as	router bgp
bgp fast-external-fallover	timers bgp
bgp graceful-restart	

- The commands listed in Table 7 configure parameters for all address families within the current VRF context.

Table 7: Commands Affecting All Address Families in a VRF

distance bgp	synchronization

- The commands listed in Table 8 configure parameters only for the current address family context.

Table 8: Commands Affecting the Current Address Family

address family	disable-dynamic-redistribute
aggregate-address	external-paths
auto-summary	ip route-type
bgp dampening	maximum-paths
bgp wait-on-end-of-rib	network
check-vpn-next-hops	redistribute
default-information originate	table-map

- The commands listed in Table 9 configure parameters for a peer or peer group, regardless of address family. If the peer or peer group is activated in more than one address family, the values are changed in all those address families. These commands are said to apply on a per-VRF basis. In the following example, EBGP multihop is configured for the session, but when you configure an address family, it is not available—that is, EBGP multihop is not configurable per address family:

```
host1(config-router)#neighbor 10.1.3.4 remote-as 1234
host1(config-router)#neighbor 10.2.3.4 ebgp-multihop 5
host1(config-router)#address-family ipv4 multicast
host1(config-router-af)#neighbor 10.2.3.4 ebgp-multihop ?
                                          ^
% Invalid input detected at '^' marker.
host1(config-router-af)#exit-address-family
```

Table 9: Commands Affecting All Address Families for the Specified Peer or Peer Group

neighbor advertisement-interval	neighbor maximum-update-size
neighbor allow	neighbor passive
neighbor bfd-liveness-detection	neighbor password
neighbor capability	neighbor peer-type
neighbor description	neighbor remote-as
neighbor ebgp-multihop	neighbor rib-out disable
neighbor graceful-restart	neighbor shutdown
neighbor graceful-restart restart-time	neighbor site-of-origin
neighbor graceful-restart stalepaths-time	neighbor timers
neighbor ibgp-singlehop	neighbor update-source
neighbor lenient	neighbor weight

- The commands listed in Table 10 configure parameters separately for each address family exchanged over the BGP session. If you configure these parameters for a peer or peer group that is activated in more than one address family, the values are affected only for the current address family. The inbound route map is such a parameter; the following example demonstrates that a BGP session can have a different inbound route map for each address family.

  host1(config-router)#**neighbor 1.1.3.4 remote-as 1234**
  host1(config-router)#**neighbor 1.2.3.4 route-map ucast-map in**
  host1(config-router)#**address-family ipv4 multicast**
  host1(config-router-af)#**neighbor 1.2.3.4 activate**
  host1(config-router-af)#**neighbor 1.2.3.4 route-map mcast-map in**
  host1(config-router-af)#**exit-address-family**

Table 10: Commands Affecting Only the Current Address Family for the Specified Peer or Peer Group

neighbor activate	neighbor peer-group
neighbor advertise-map	neighbor prefix-list
neighbor allowas-in	neighbor prefix-tree
neighbor as-override	neighbor remote-private-as
neighbor default-originate	neighbor route-map
neighbor distribute-list	neighbor route-reflector-client
neighbor filter-list	neighbor send-community
neighbor local-as	neighbor send-label
neighbor maximum-prefix	neighbor soft-reconfiguration inbound
neighbor next-hop-self	neighbor unsuppress-map
neighbor next-hop-unchanged	

Inheritance of Configuration Values

Peer groups inherit all configuration values that are globally configured. However, attributes configured for a peer group override inherited global configuration values. Individual peers that are members of peer groups inherit all configuration values from the peer group. However, attributes configured on a peer override values inherited from the peer group of which it is a member.

The **neighbor** commands enable you to control features or set parameters for individual peers or for peer groups. These commands can be classified into the four categories shown in Table 11, based on whether the command enables a feature or sets parameters, the levels at which it behaves, and how the **no** version of the command compares with the **default** version.

Table 11: Behavior of Neighbor Commands

Category A: **Enable or disable a feature that can be configured for a peer or for a peer group**	**Category B:** **Enable or disable a feature that can be configured for a peer, for a peer group, or globally**	**Category C:** **Set parameters for a peer or for a peer group**	**Category D:** **Set parameters for a peer, for a peer group, or globally**
■ neighbor activate ■ neighbor advertise-map ■ neighbor as-override ■ neighbor bfd-liveness-detection ■ neighbor capability ■ neighbor ebgp-multihop ■ neighbor ibgp-singlehop ■ neighbor lenient ■ neighbor next-hop-self ■ neighbor next-hop-unchanged ■ neighbor passive ■ neighbor remove-private-as ■ neighbor route-reflector-client ■ neighbor send-community ■ neighbor soft-reconfiguration inbound	■ neighbor default-originate ■ neighbor graceful-restart ■ neighbor rib-out disable ■ neighbor shutdown	■ neighbor advertisement-interval ■ neighbor allow ■ neighbor allowas-in ■ neighbor description ■ neighbor distribute-list ■ neighbor filter-list ■ neighbor graceful-restart restart-time ■ neighbor graceful-restart stalepaths-time ■ neighbor local-as ■ neighbor maximum-orf-entries ■ neighbor maximum-prefix ■ neighbor maximum-update-size ■ neighbor password ■ neighbor peer-group ■ neighbor peer-type ■ neighbor prefix-list ■ neighbor prefix-tree ■ neighbor remote-as ■ neighbor route-map ■ neighbor send-label ■ neighbor site-of-origin ■ neighbor unsuppress-map ■ neighbor update-source ■ neighbor weight	■ neighbor timers

Some of the commands in Table 11 inherit global values set by other commands. Table 12 describes the relationship between these commands.

Table 12: Inheritance from Other Commands

Category B Command	Inherits Global Values Set By
neighbor default-originate	default-information originate
neighbor graceful-restart	bgp graceful-restart
neighbor rib-out disable	rib-out disable
neighbor shutdown	bgp shutdown
neighbor graceful-restart restart-time	bgp graceful-restart restart-time
neighbor graceful-restart stalepaths-time	bgp graceful-restart stalepaths-time

Example 1 For category A and B commands, the behavior of the **no** version of the command is different from the behavior of the **default** version of the command. The **no** version explicitly disables the feature:

- Applied to a peer, the **no** version disables the feature regardless of whether the feature is enabled for any peer group to which it belongs.
- Applied to a peer group, the **no** version disables the feature regardless of whether the feature is enabled for BGP globally or by default.

The **default** version simply unconfigures the feature for the peer or peer group.

- Applied to a peer, the **default** version causes the peer to inherit the state of the feature (enabled or disabled) from any peer group to which it belongs.
- Applied to a peer group, the **default** version causes the peer group to inherit the state of the feature (enabled or disabled) from the BGP global configuration.

The following example illustrates this difference and the inheritance concept with the **neighbor soft-reconfiguration inbound** command.

```
host1(config-router)#neighbor lisbon peer-group
host1(config-router)#neighbor 10.19.7.8 peer-group lisbon
```

Inbound soft-reconfiguration is disabled by default, hence it is currently disabled for both the lisbon peer group and peer 10.19.7.8.

```
host1(config-router)#neighbor lisbon soft-reconfiguration inbound
```

Inbound soft-reconfiguration is now enabled for the lisbon peer group. Because the peer inherits values from the peer group, inbound soft-reconfiguration is now also enabled for peer 10.19.7.8.

```
host1(config-router)#no neighbor 10.19.7.8 soft-reconfiguration inbound
```

The **no** command disables inbound soft-reconfiguration for peer 10.19.7.8, overriding the configuration of the peer group to which the peer 10.19.7.8 belongs. The configuration of an individual peer takes precedence over the configuration of the peer group to which the peer belongs.

```
host1(config-router)#default neighbor 10.19.7.8 soft-reconfiguration inbound
```

The **default** version returns the peer to inheriting the peer group configuration. Because inbound soft-reconfiguration is still enabled for lisbon, it is now also enabled for peer 10.19.7.8.

```
host1(config-router)#default neighbor lisbon soft-reconfiguration inbound
```

Finally, this last command returns the peer group configuration to the default value, disabling inbound soft-reconfiguration. The peer 10.19.7.8 inherits this value.

Example 2 For category C and D commands, the behavior of the **no** version of the command is the same as the behavior of the **default** version of the command. The following example illustrates this behavior and the inheritance concept for the **neighbor timers** command.

By default, the BGP global keepalive timer is 30 seconds and the global hold-time timer is 90 seconds.

```
host1(config-router)#neighbor eastcoast peer-group
host1(config-router)#neighbor 10.10.21.23 peer-group eastcoast
```

Peer group eastcoast and peer 10.10.21.23 both have the default timer values. The peer group inherits the global timer values; the peer is a member of eastcoast and inherits the timer values from the peer group.

```
host1(config-router)#neighbor eastcoast timers 15 40
```

Now peer group eastcoast has a keepalive timer of 15 seconds and a hold-time timer of 40 seconds. Peer 10.10.21.23 inherits these values from the peer group.

```
host1(config-router)#no neighbor 10.10.21.23 timers
```

Now peer 10.10.21.23 has its timers reset to the global values of 30 and 90 seconds. The configuration of an individual peer takes precedence over the configuration of the peer group to which the peer belongs, which in turn takes precedence over the global configuration.

```
host1(config-router)#default neighbor 10.10.21.23 timers
```

Nothing changes. For commands in categories C and D, the behavior of the **default** version is the same as the **no** version. Peer 10.10.21.23 still has the global timer values.

```
host1(config-router)#neighbor eastcoast timers 20 20
```

The eastcoast peer group now has timer values of 20 seconds. Peer 10.10.21.23 still has the global timer values.

Limitations on Inheritance

All BGP peers that are members of the same peer group must send essentially the same updates. Accordingly, all members of a peer group must be the same kind of peer; that is, all must be internal peers, all must be external peers, or all must be confederation peers.

Outbound policies configured for peer groups are still inherited by peer group members, but you cannot override this inherited outbound policy by configuring a different outbound policy on individual members of that peer group with the following commands:

Table 13: Commands That Do Not Override Inherited Outbound Policy

neighbor as-override	neighbor next-hop-unchanged	neighbor route-map out
neighbor default-originate	neighbor prefix-list out	neighbor route-reflector-client
neighbor distribute-list out	neighbor prefix-tree out	neighbor send-community
neighbor filter-list out	neighbor remove-private-as	neighbor unsuppress-map
neighbor next-hop-self		

NOTE: This restriction does not apply to inbound policy, which you can still override per peer.

The update messages can vary for members of a peer group as follows:

- The next hop can be different for each update sent to peer group members if the members are all external peers.
- The AS path can be different for each update sent to peer group members if the members are all external peers if you have enabled AS override with the **neighbor as-override** command.

Setting the BGP Identifier

By default, the router ID of the router is used as the BGP identifier. You can use the **bgp router-id** command to configure an IP address as the BGP identifier.

bgp router-id

- Use to configure an IP address as the BGP identifier.
- Example

  host1(config-router)#**bgp router-id 10.25.1.1**

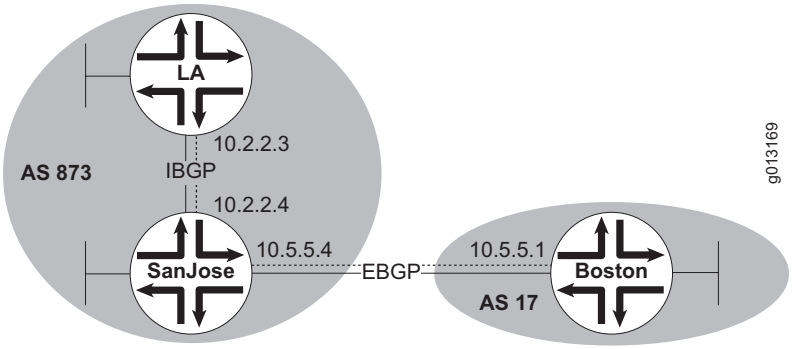
- The new BGP identifier is used in open messages sent after you issue the command. To use the new BGP identifier for sessions already in the established state, you must use the **clear ip bgp** command to perform a hard clear.
- Use the **no** version to restore the router ID as the BGP identifier.

Configuring Neighbors

Use the **neighbor remote-as** command to create a BGP peering session with a given BGP peer—identified by its IP address—in a given AS. Note that the **neighbor remote-as** command must be issued on both routers on either side of a BGP session for the BGP session to become established.

Consider the simple network structure shown in Figure 10. Routers LA and SanJose are IBGP peers within AS 873. Router SanJose has an EBGP peer, router Boston, in AS 17.

Figure 10: Configuring Neighbors



The following commands configure router Boston with router SanJose as a peer:

```
host1(config)#router bgp 17
host1(config-router)#neighbor 10.5.5.4 remote-as 873
```

The following commands configure router SanJose with router LA and router Boston as peers:

```
host2(config)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
```

The following commands configure router LA with router SanJose as a peer:

```
host3(config)#router bgp 873
host3(config-router)#neighbor 10.2.2.4 remote-as 873
```

neighbor remote-as

- Use to add an entry to the BGP neighbor table.
- Specifying a neighbor with an AS number that matches the AS number specified in the **router bgp** command identifies the neighbor as internal to the local AS. Otherwise, the neighbor is treated as an external neighbor.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately.
- Use the **no** version to remove an entry from the table.

Configuring BGP Peer Groups

You will often want to apply the same policies to most or all of the peers of a particular BGP speaker. Update policies are usually defined by route maps, filter lists, and distribution lists. You can reduce the configuration effort by defining a peer group made up of these peers.

A peer group is defined relative to a particular BGP speaker. Figure 11 shows two peer groups, *eastcoast* and *leftcoast*. Each of these peer groups is defined for router Chicago, the hub router. Routers Boston, NY, and Miami have no knowledge of being members of Router Chicago's *eastcoast* peer group. Similarly, routers SanFran, LA, and SanDiego have no knowledge of being members of router Chicago's *leftcoast* peer group.

The following commands configure the *eastcoast* peer group on router Chicago:

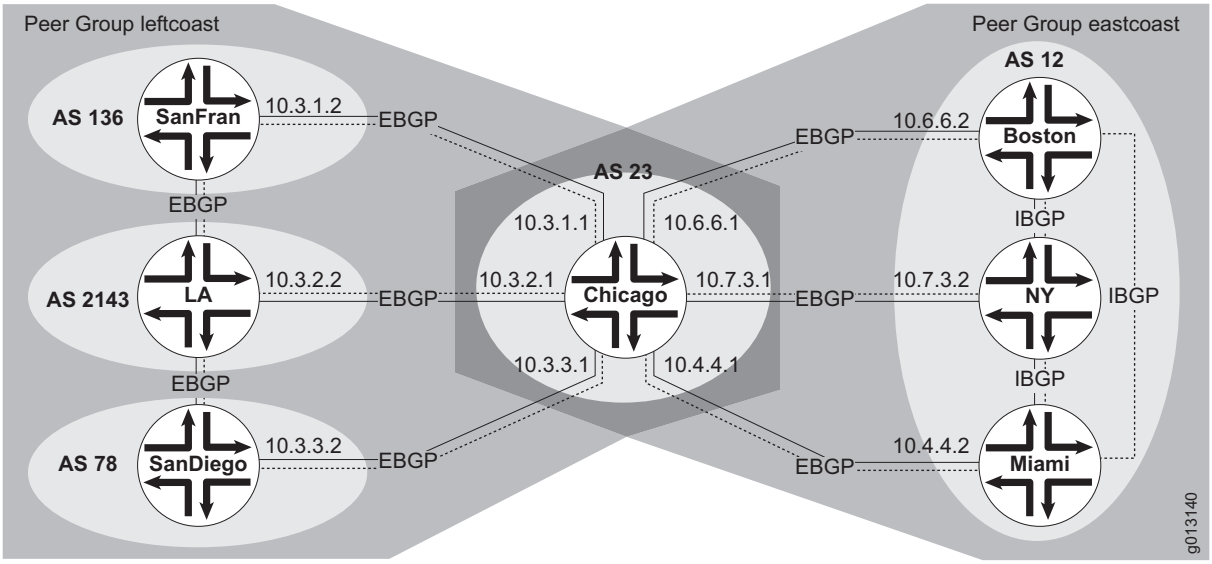
```
host1(config)#router bgp 23
host1(config)#route-map wtset permit 10
host1(config-route-map)#set weight 25
host1(config-route-map)#exit
host1(config-router)#neighbor eastcoast peer-group
host1(config-router)#neighbor eastcoast route-map wtset in
host1(config-router)#neighbor 10.6.6.2 remote-as 12
host1(config-router)#neighbor 10.6.6.2 peer-group eastcoast
host1(config-router)#neighbor 10.7.3.2 remote-as 12
host1(config-router)#neighbor 10.7.3.2 peer-group eastcoast
host1(config-router)#neighbor 10.4.4.2 remote-as 12
host1(config-router)#neighbor 10.4.4.2 peer-group eastcoast
```

The following commands configure the *leftcoast* peer group on router Chicago:

```
host1(config-router)#neighbor leftcoast peer-group
host1(config-router)#neighbor 10.3.3.2 remote-as 78
host1(config-router)#neighbor 10.3.3.2 peer-group leftcoast
host1(config-router)#neighbor 10.3.2.2 remote-as 2143
host1(config-router)#neighbor 10.3.2.2 peer-group leftcoast
host1(config-router)#neighbor 10.3.1.2 remote-as 136
host1(config-router)#neighbor 10.3.1.2 peer-group leftcoast
```

The multiprotocol extensions to BGP enable the exchange of information within different types of *address families. By* default, peers and peer groups exist in the unicast IPv4 address family and exchange unicast IPv4 addresses. For information on configuring and activating BGP peer groups within address families, see *Configuring the Address Family* on page 42.

Figure 11: BGP Peer Groups



neighbor peer-group

- Two versions of this command exist. Use to create a BGP peer group or to configure a BGP neighbor to be a member of a peer group.
- To create a BGP peer group, specify a *peerGroupName* for the new peer group. Use the **no** version to remove a peer group.
- To assign members to a peer group, specify an *ip-address* and a *peerGroupName* of a BGP neighbor that belongs to this group.
- This command takes effect immediately.
- Use the **no** version to remove a neighbor from a peer group.

NOTE: You cannot mix IPv4 and IPv6 peer members in a peer group. Only one type peer is allowed, IPv4 or IPv6. For example, the following error is generated if an IPv6 peer group member is added to a peer group that already has IPv4 members; that is, where the peer-group type is IPv4:

```
host1(config-router)#neighbor 1::1 peer-group hamburg
% Unable to set 'peer-group' for address family ipv4:unicast for peer 1::1
in core (IPv6 peer cannot be member of a peer-group of type IPv4)
```

For information about the inheritance of configuration values by peer groups and peers, see *Inheritance of Configuration Values* on page 20.

Setting the Peer Type

Each peer group must have a peer type before any BGP sessions for members of that peer group are allowed to come up and before the Adj-RIBs-Out table of that peer group can be filled. You can use the **neighbor peer-type** command to explicitly configure a peer type for a peer group.

Alternatively, you can implicitly configure the peer type of a peer group by either of the following methods:

- Configure a remote AS for the peer group.
- Assign a peer with a configured remote AS as a member of the peer group.

In both of these implicit cases, the remote AS is combined with the local AS, the configured confederation ID, and the configured confederation peers to determine the peer type of the peer group.

neighbor peer-type

- Use to specify a peer type for a peer group.
- This command is supported only for peer groups; it is not available for individual peers.
- Use the **internal** keyword to specify that peers must be in the same AS or, if confederations are employed, in the same sub-AS in the same confederation.
- Use the **external** keyword to specify that peers must be in a different AS.
- Use the **confederation** keyword to specify that peers must be in a different sub-AS in the same confederation. Use this keyword only if confederations are employed.
- This command takes effect immediately. If the command changes the peer type of the peer group, all BGP sessions for members of that peer group are automatically bounced.
- All the members of the peer group inherit the characteristic configured with this command. It cannot be overridden for a specific peer, because the command applies only to peer groups.
- Example

  host1(config-router)#**neighbor promispeers peer-type internal**

- Use the **no** version to remove the configuration from the peer group.

Assigning a Description

You can associate a description with a BGP neighbor or a peer group. This is a convenient way to store minimal pertinent information about the neighbor.

neighbor description

- Use to associate a textual description of up to 80 characters with a BGP neighbor or peer group.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately.
- Example

  host1(config-router)#**neighbor 10.11.0.5 description bostonmetropeer**

- Use the **no** version to remove the description.

Logging Neighbor State Changes

You can force BGP to log a message whenever a peer enters or leaves the Established state.

bgp log-neighbor-changes

- Use to log a notice message to the bgpNeighborChanges log when a neighbor enters or leaves the Established state for any reason.
- The severity of the log message is notice by default.
- Issue the **log destination console severity notice** command to display the messages on the console.
- This command takes effect immediately.
- Example

  host1:3(config)#**bgp log destination console severity notice**
  host1:3(config)#**router bgp 100**
  host1:3(config-router)#**bgp log-neighbor-changes**
  NOTICE 04/30/2001 21:06:22 bgpNeighborChanges (3,4.4.4.4): peer 4.4.4.4 in core leaves established state
  NOTICE 04/30/2001 21:06:22 bgpNeighborChanges (3,5.5.5.5): peer 5.5.5.5 in core leaves established state
  NOTICE 04/30/2001 21:06:22 bgpNeighborChanges (3,6.6.6.6): peer 6.6.6.6 in core leaves established state
  NOTICE 04/30/2001 21:06:22 bgpNeighborChanges (3,13.13.13.1): peer 13.13.13.1 in core leaves established state
  NOTICE 04/30/2001 21:06:31 bgpNeighborChanges (3,4.4.4.4): peer 4.4.4.4 in core enters established state
  NOTICE 04/30/2001 21:06:31 bgpNeighborChanges (3,5.5.5.5): peer 5.5.5.5 in core enters established state
  NOTICE 04/30/2001 21:06:31 bgpNeighborChanges (3,6.6.6.6): peer 6.6.6.6 in core enters established state
  NOTICE 04/30/2001 21:06:31 bgpNeighborChanges (3,13.13.13.1): peer 13.13.13.1 in core enters established state

- Use the **no** version to stop logging.

Specifying a Source Address for a BGP Session

By default, BGP uses the IP address of the outgoing interface toward the peer as the source IP address for the TCP connection over which the BGP session runs. If the outgoing interface goes down, the BGP session is dropped because the IP source address is no longer valid. This is appropriate behavior for EBGP sessions because the EBGP peers typically can reach each other only by virtue of being connected to a common subnet.

For IBGP sessions, however, you typically want BGP sessions to be automatically rerouted around interfaces that are down. You can issue the **neighbor update-source** command to accomplish this. This command instructs BGP to use the IP address of a specified interface as the source address of the underlying TCP connection. Typically, a loopback interface is used because it is inherently stable.

For example, you can specify that BGP use loopback interface 2 as the source for messages that it sends to peer 192.50.30.1:

```
host1(config)#neighbor 192.50.30.1 update-source loopback 2
```

neighbor update-source

- Use to allow a BGP session to use the IP address of a specific operational interface as the source address for TCP connections.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately and automatically bounces the BGP session.
- If you specify an interface in this command and the interface is later removed, then this command is also removed from the router configuration.
- Use the **no** version to restore the interface assignment to the closest interface.

The source address that you specify with the **neighbor update-source** command is also used by BGP as the default value for the next hop address advertised for IPv4 or IPv6 prefixes.

The source addresses and next hop address that result from using the **neighbor update-source** command vary depending on the configuration of the command. Table 14 lists the results for different configurations.

Table 14: Source Addresses and Default Next Hop Addresses for Various Configurations

Configured Neighbor Address	Configured Update Source Address	Source Address used for TCPv4 and TCPv6 Connection	Default Next Hop Value for IPv4 Prefixes	Default Next Hop Value for IPv6 Prefixes
IPv4 neighbor address	IPv4 source address	IPv4 source address	IPv4 source address	IPv4 source address mapped to an IPv6 address
IPv4 neighbor address	IPv6 source address	Not allowed	Not allowed	Not allowed
IPv4 neighbor address	Interface name	IPv4 address of the interface. If the interface does not have an IPv4 address, then the session does not come up.	IPv4 address of the interface	IPv6 address of the interface. If the interface does not have an IPv6 address, then the IPv4 address of the interface is mapped to an IPv6 address.
IPv6 neighbor address	IPv6 source address	IPv6 source address	0.0.0.0	IPv6 source address
IPv6 neighbor address	IPv4 source address	Not allowed	Not allowed	Not allowed
IPv6 neighbor address	Interface name	IPv6 address of the interface. If the interface does not have an IPv6 address, then the session does not come up.	IPv4 address of the interface. If the interface does not have an IPv4 address, then 0.0.0.0.	IPv6 address of the interface

You can override a native IPv6 next-hop address with either the **neighbor update-source** command or an outbound route map.

When you specify an interface with the **neighbor update-source** command, the IPv4-mapped IPv6 address of the interface is used instead of the native IPv6 address for the next hop.

```
host1(config)#interface loopback 0
host1(config-if)#ip address 10.1.1.1/32
host1(config-if)#exit
host1(config)#router bgp 100
host1(config-router)#neighbor 2::2 update-source loopback 0
```

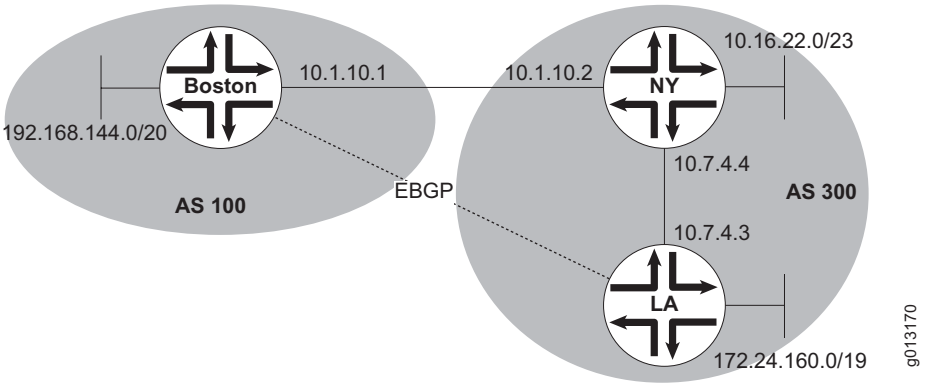
In this example, the IPv4-mapped IPv6 address of the loopback 0 interface is the next-hop address sent when IPv6 prefixes are advertised. However, if loopback 0 has an IPv6 address, then that address is used as the default next hop for advertising IPv6 prefixes.

Specifying Peers That Are Not Directly Connected

Normally, EBGP speakers are directly connected. When you cannot connect EBGP speakers directly, you can use the **neighbor ebgp-multihop** command to specify that the neighbor is more than one hop away. You generally need static routes to configure multihop connections. By default, the one-hop limitation per EBGP peers is enforced by the time-to-live attribute. You can override this default limit by using the *ttl* variable to specify the maximum number of hops to the peer.

In Figure 12, router Boston and router LA are connected together through router NY, rather than by a direct connection. Routers Boston and LA are configured as external peers with the **neighbor ebgp-multihop** command because no direct connection exists between them. Because router NY is not a BGP speaker, static routes are configured on routers Boston and LA. The configuration for router NY is not shown, because it does not involve BGP.

Figure 12: Using EBGP-Multihop



The following commands achieve the BGP configuration.

To configure router Boston:

```
host1(config)#ip route 10.7.4.0 255.255.255.0 10.1.10.2
host1(config)#router bgp 100
host1(config-router)#neighbor 10.7.4.3 remote-as 300
host1(config-router)#neighbor 10.7.4.3 ebgp-multihop
```

To configure router LA:

```
host2(config)#ip route 10.1.10.0 255.255.255.0 10.7.4.4
host2(config)#router bgp 300
host2(config-router)#neighbor 10.1.10.1 remote-as 100
host2(config-router)#neighbor 10.1.10.1 ebgp-multihop
```

neighbor ebgp-multihop

- Use to configure BGP to accept route updates from external peers in networks that are not directly connected to the local peer.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- Any external BGP peering that is not resolved by a connected route is treated as a multihop. Configurations with loopback-to-loopback external BGP peering require the **neighbor ebgp-multihop** command to work properly. In these configurations, the **neighbor remote-as** command is issued with the address of a loopback interface.
- This command takes effect immediately and automatically bounces the BGP session.

- Use the **no** version to return BGP to halt acceptance of such routers. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Specifying a Single-Hop Connection for IBGP Peers

IBGP peers are multihop by default. However, you can use the **neighbor ibgp-single-hop** command to enable single-hop connections for IBGP peers.

neighbor ibgp-singlehop

- Use to specify an internal BGP peer as a single-hop peer for IBGP sessions.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- If the neighbor session type is anything other than internal BGP, issuing this command generates an error message.
- This command takes effect immediately and automatically bounces the BGP session.
- Example

  host1(config-router)#**neighbor 192.168.32.15 ibgp-singlehop**

- Use the **no** version to restore the default behavior, wherein the internal peer cannot be a single-hop peer. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Controlling the Number of Prefixes

As the routing table increases in size, the processor and memory resources required to process routing information increases. Some peers send so much routing information that a BGP speaker can be overwhelmed by the updates. You can use the **neighbor maximum-prefix** command to limit how many prefixes can be received from a neighbor.

The router resets the BGP connection when the specified maximum is exceeded. You can use the **warning-only** keyword to log a warning rather than reset the connection. You can also configure the router so that a warning is logged when a specified percentage of the maximum is exceeded.

In the following example, the router is configured to reset the BGP connection when it receives more than 1,000 prefixes from its neighbor at 2.2.2.2:

```
host1(config)#router bgp 100
host1(config-router)#neighbor 2.2.2.2 maximum-prefix 1000
```

neighbor maximum-prefix

- Use to control how many prefixes can be received from a neighbor.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.

- By default, BGP checks the maximum prefix limit only against accepted routes. You can specify the **strict** keyword to force BGP to check the maximum prefix against all received routes. The accepted and received routes will likely differ when you have configured inbound soft reconfiguration and route filters for incoming traffic.
- This command takes effect immediately. To prevent a peer from continually flapping, when it goes to state idle because the maximum number of prefixes has been reached, the peer stays in state idle until you use the **clear ip bgp** command to issue a hard clear.
- Use the **no** version to remove the maximum number of prefixes.

Removing Private AS Numbers from Updates

You might choose to conserve AS numbers by assigning private AS numbers to some customers. You can assign private AS numbers from the range 64,512 to 65,535. However, when BGP advertises prefixes to other ISPs, it is undesirable to include the private AS numbers in the path. Configure the external neighbors to drop the numbers with the **neighbor remove-private-as** command.

neighbor remove-private-as

- Use to remove private AS numbers only in updates sent to external peers.
- All private AS numbers are removed regardless of their position in the AS-path attribute and regardless of the presence of public AS numbers.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override the characteristic for a specific member of the peer group.
- Example

  host1(config-router)#**neighbor 10.10.128.52 remove-private-as**

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to halt the removal of private AS numbers in updates sent to external peers. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Checking AS Path Length

You can use the **bgp maxas-limit** command to prevent the forwarding of routes having AS paths longer than a specified limit.

bgp maxas-limit

- Use to require BGP to check the AS path in all received update messages.
- If a received AS path is longer than the specified limit:
  - The route is stored in the BGP routing table and therefore is displayed by the **show ip bgp** commands.
  - The route is not a candidate for being selected as a best path, is not stored in the forwarding information base, and is not propagated to external or internal peers.
- Changes in the limit do not affect routes previously received. Clearing the BGP sessions (**clear ip bgp**) forces a resend of all routes; the new limits are then applied on receipt of the routes.
- Example

  host1(config-router)#**bgp maxas-limit 42**

- Causes BGP to check the AS path of all routes received after you issue the command.

  To apply the new behavior to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

- Use the **no** version to halt checking of received AS path lengths.

If you use the **fields as-path** option with the **show ip bgp** command, the display indicates routes whose AS path exceeds the limit. The following display illustrates the result of setting the AS path length limit to 5:

```
host1:3#show ip bgp fields intro best peer loc-pref as-path
Local router ID 13.13.13.3, local AS 200
 10 paths, 5 distinct prefixes (520 bytes used)
 6 paths selected for route table installation
 14 path attribute entries (1943 bytes used)

Status codes: > best

 Prefix             Peer            LocPrf AS-path
   10.23.40.1/32    192.168.13.1    200    100 211 32 15 67 44 (too long)
> 10.23.40.1/32    172.123.23.2    100    100 211
> 10.23.40.2/32    192.168.13.1    200    100 211 32 15 67
   10.23.40.2/32    172.123.23.2    100    100 211 32
> 10.23.40.3/32    192.168.13.1           100 211 32 15
   10.23.40.3/32    172.123.23.2           100 211 32 15
   10.23.40.4/32    192.168.13.1    100    100 211 32
> 10.23.40.4/32    172.123.23.2    200    100 211 32 15 67
> 10.23.40.5/32    192.168.13.1    100    100 211
   10.23.40.5/32    172.123.23.2    200    100 211 32 15 67 44 (too long)
```

Enabling MD5 Authentication on a TCP Connection

You can use the **neighbor password** command to enable MD5 authentication on a TCP connection between two BGP peers. Enabling MD5 authentication causes each segment sent on the TCP connection between them to be verified.

You must configure MD5 authentication with the same password on both BGP peers; otherwise, the router does not make the connection between the BGP peers.

The MD5 authentication feature uses the MD5 algorithm. When you specify this command, the router generates and checks the MD5 digest on every segment sent on the TCP connection.

In the following example, the password is set to "opensesame":

```
host1(config)#router bgp 100
host1(config-router)#neighbor 2.2.2.2 password opensesame
```

The **show ip bgp neighbors** command does not reveal the password, but does indicate whether MD5 authentication is configured for the session. The output of the **show configuration** command varies as follows:

- If you use the **8** keyword to specify that the password is encrypted, then the output of the **show configuration** command displays the text that you entered (the ciphertext password).
- If you do not use the **8** keyword (that is, you use the **0** keyword or no encryption keyword), and if the **service password-encryption** command has not been issued, then the output of the **show configuration** command displays the text that you entered (the plaintext password).
- If you do not use the **8** keyword (that is, you use the **0** keyword or no encryption keyword) but the **service password-encryption** command has been issued, then the output of the **show configuration** command displays an encrypted password that is equivalent to the cleartext password that you entered.

neighbor password

- Use to enable MD5 authentication on a TCP connection between two BGP peers.
- If you configure a password for a neighbor, an existing session is torn down and a new one established.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- If a router has a password configured for a neighbor, but the neighbor router does not, a message indicating this condition appears on the console while the routers attempt to establish a BGP session between them.
- Similarly, if the two routers have different passwords configured, a message appears on the console indicating that this condition exists.

- Use the **8** keyword to indicate that the password is encrypted (entered in ciphertext). Use the **0** keyword to indicate that the password is unencrypted (entered in plaintext).
- This command takes effect immediately and automatically bounces the BGP session.
- Use the **no** version to disable MD5 authentication.

Setting the Maximum Size of Update Messages

You can use the **neighbor maximum-update-size** command to set the maximum size of update messages transmitted to a BGP peer.

For example, to set the maximum update size to 2,000 octets:

```
host1(config)#router bgp 100
host1(config-router)#neighbor 10.12.2.5 maximum-update-size 2000
```

neighbor maximum-update-size

- Use to set the maximum size for transmitted BGP update messages.
- Set the maximum-update-size to a range: 256–4096.
- The default is 1024 octets.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- BGP always *accepts* updates of up to 4096 octets, regardless of the setting for transmitted updated messages.
- Applies to all update messages sent after you issue the command.
- Use the **no** version to restore the default value.

Setting Automatic Fallover

You can use the **bgp fast-external-fallover** command to specify that in the event of the failure of a link to any adjacent external peer, the BGP session is immediately and automatically brought down rather than waiting for the TCP connection to fail or for the hold timer to expire.

bgp fast-external-fallover

- Use to immediately bring down a BGP session if the link to an adjacent external peer fails.
- If you do not issue this command, the BGP session is not brought down in the event of a link failure until the TCP connection fails or the hold timer expires.
- This command takes effect immediately.
- Use the **no** version to stop automatically bringing down the session in the event of link failure.

Setting Timers

BGP uses a keepalive timer to control the interval at which keepalive messages are sent. A hold-time timer controls how long BGP waits for a keepalive message before declaring a peer not available.

BGP negotiates the hold time with each neighbor when establishing the BGP connection. The peers use the lower of the two configured hold times. BGP sets the keepalive timer based on this negotiated hold time and the configured keepalive time.

neighbor timers

- Use to set the keepalive and hold-time timers for the specified neighbor or peer group.
- Overrides timer values set with the **timers bgp** command.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- If you set the keepalive timer to 0, BGP does not send any keepalive messages.
- If you do not expect the peer to send any keepalives, set the hold-time timer to 0.
- This command takes effect immediately and automatically bounces the session to force BGP to send a new open message to renegotiate the new timer values.
- Example

  host1(config-router)#**neighbor 192.168.21.5 timers 90 240**

- Use the **no** version to restore the default values on the specified neighbor or peer group—30 seconds for the keepalive timer and 90 seconds for the hold-time timer.

timers bgp

- Use to set the keepalive and hold-time timers for all neighbors.
- If you set the keepalive timer to 0, BGP does not send any keepalive messages.
- If you do not expect the peer to send any keepalives, set the hold-time timer to 0.
- Example

  host1(config-router)#**timers bgp 75 300**

- The new timer values are used by every session that comes up after you issue the command; timers configured specifically for the sessions take precedence over these values.

  To force sessions that are already established to use the new timer values, you must use the **clear ip bgp** command to perform a hard clear.

- Use the **no** version to restore the default values on all neighbors—30 seconds for the keepalive timer and 90 seconds for the hold-time timer.

Automatic Summarization of Routes

By default, all routes redistributed into BGP from an IGP are automatically summarized to their natural network masks.

auto-summary

- Use to reenable automatic summarization of routes redistributed into BGP.
- Automatic summarization is enabled by default. However, creating an address family for a VRF automatically disables automatic summarization for that address family.
- This command takes effect immediately.
- Use the **no** version to disable automatic summarization of redistributed routes.

Administrative Shutdown

You can administratively shut down particular BGP neighbors or peer groups without removing their configuration from BGP by using the **neighbor shutdown** command.

You can also administratively shut down BGP globally by using the **bgp shutdown** command.

bgp shutdown

- Use to shut down BGP globally.
- This command takes effect immediately.
- Example

  host1(config-router)#**bgp shutdown**

- Use the **no** version to reenable BGP.

neighbor shutdown

- Use to shut down a neighbor or peer group without removing their configuration.
- This command takes effect immediately.
- Use the **no** version to reenable a neighbor or peer group that was previously shut down. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Configuring BGP for Overload Conditions

You can specify how you want BGP to behave when it is running out of memory in an overload condition. You can have BGP either shut itself down or continue running; in the latter case, BGP performance might be altered because of the lack of resources.

overload shutdown

- Use to shut down BGP if it runs out of memory.
- The default behavior is for BGP to transition from the Up state to the Overload state and continue running.
- This command takes effect immediately.
- Example

  host1(config-router)#**overload shutdown**

- Use the **no** version to restore the default behavior.

The following partial outputs show how the BGP state is indicated by the **show ip bgp summary** command:

```
host1#show ip bgp summary
Local router ID 10.1.0.1, local AS 1
  Administrative state is Start
  Operational state is Overload
  Shutdown in overload state is disabled
  Default local preference is 100
...
host1#show ip bgp summary
Local router ID 10.1.0.1, local AS 1
  Administrative state is Start
  Operational state is Down due to transition from Overload state
  Shutdown in overload state is enabled
  Default local preference is 100
...
```

Enabling Route Storage in Adj-RIBs-Out Tables

By default, a BGP speaker does not store a copy of each route it sends to a BGP peer in the Adj-RIBs-Out table for that peer. However, you can force BGP to store a copy of routes in the Adj-RIBs-Out table for a particular peer or peer group by enabling that Adj-RIBs-Out table ("enabling rib-out") with the **no neighbor rib-out disable** command. Alternatively, you can use the **no rib-out disable** command to affect all BGP peers. The details of route storage vary between peers and peer groups.

For peers, BGP stores a single bit with each route in the table to indicate whether it has previously advertised the route to the peer, enabling the avoidance of spurious withdrawals. The full set of attributes for each route is not stored in the peer Adj-RIBs-Out table.

After enabling rib-out for a peer, you can issue the **show ip bgp neighbors advertised-routes** command to display the routes that have been advertised to the peer. The attributes displayed for the routes are those from the local routing table, not those that were advertised. In other words, BGP stores the attributes prior to the application of any outbound policy.

For peer groups, BGP stores the full set of attributes associated with the route after the application of any outbound policy; that is, it stores the attributes as they will be advertised. BGP does not store a bit to track whether a route was advertised to the peer group. Storing the full attribute set for each peer group route is memory intensive but acceptable for peer groups, because the number of peer groups is relatively small. An advantage of enabling rib-out for peer groups is that convergence is accelerated because the attributes for each route are already determined for all routes to be advertised to the peer group. BGP has to apply outbound policy only once for each route rather than once for each peer for each route.

After enabling rib-out for a peer group, you can issue the **show ip bgp advertised-routes** command to display the routes that will be advertised to the peer group and the attributes (after the application of any outbound policy) that will be advertised with the routes.

When you have enabled rib-out for individual peers or a peer group, before sending an advertisement or withdrawal the router compares the route it is about to send with the last route sent for the same prefix (and stored in the Adj-RIBs-Out table for the peer or peer group) and sends the update message only if the new information is different from the old.

The comparison prevents the sending of unnecessary withdrawals for both peers and peer groups, because the BGP speaker will not send a withdrawal if the table indicates it has not previously advertised that route to the peer. However, because the route attributes are no longer stored with the routes in peer Adj-RIBs-Out tables, BGP cannot compare them with the attributes in the new update message. Consequently, BGP cannot determine whether the update contains new attributes or the same attributes as those previously advertised, and might send superfluous advertisements to peers. This circumstance does not happen for peer groups, because their Adj-RIBs-Out tables store the full attribute set.

Effects of Changing Outbound Policies

After you change the outbound policy for a peer or peer group, the policy changes do not take effect until you issue either a hard clear or an outbound soft clear. (See *Resetting a BGP Connection* on page 95 for information about performing clears with the **clear ip bgp** command.) The clear action causes BGP to reapply the outbound policy of the peer or peer group to each route in the BGP routing table. BGP then stores the results in the Adj-RIBs-Out table for that peer or peer group. The BGP session with each peer or peer group member takes the routes from the appropriate Adj-RIBs-Out table and sends them in update messages to the peer or peer group member.

NOTE: You cannot change outbound policy for an individual peer group member. You can change outbound policy only for a peer group as a whole or for peers that are not members of a peer group.

neighbor rib-out disable

- Use to disable storage of routes (disable rib-out) in the specified neighbor's Adj-RIBs-Out table or in a single Adj-RIBs-Out table for the entire specified peer group.
- Route storage is disabled by default.
- If you enable storage for a peer, the peer's Adj-RIBs-Out table contains all routes actually sent to the peer. By contrast, if you enable storage for a peer group, the peer group's Adj-RIBs-Out table contains those routes that the BGP speaker intends to send to the peer group members; individual members might or might not have already received updates that advertise the routes.
- If you specify a BGP peer group by using the *peerGroupName* argument, a single Adj-RIBs-Out table is enabled for the entire peer group. You can override this configuration for a member of the peer group by issuing the command for that peer.
- Limit the number of Adj-RIBs-Out tables to no more than ten for peer groups to conserve memory resources. No limit applies to peers.
- This command takes effect immediately and automatically bounces the BGP session(s) if the command changes the current configuration.
- Example

  host1(config-router)#**no neighbor 10.15.24.5 rib-out disable**

- Use the **no** version to enable the route storage. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

rib-out disable

- Use to disable storage of routes in the Adj-RIBs-Out tables (disable rib-out) for all BGP peers.
- Route storage is disabled by default.
- This command takes effect immediately and automatically bounces the BGP session if the command changes the current configuration.
- Example

  host1(config)#**rib-out disable**

- Use the **no** version to enable the route storage. Use the **default** version to remove the explicit global configuration from all peers and reestablish inheritance of the feature configuration.

Configuring the Address Family

The BGP multiprotocol extensions specify that BGP can exchange information within different types of *address families*. The JUNOSe BGP implementation defines the following different types of address families:

- Unicast IPv4—If you do not explicitly specify the address family, the router is configured to exchange unicast IPv4 addresses by default. You can also configure the router to exchange unicast IPv4 routes in a specified VRF.

- Multicast IPv4—If you specify the multicast IPv4 address family, you can use BGP to exchange routing information about how to reach a multicast source instead of a unicast destination. For information about BGP multicasting commands, see *Chapter 1, Configuring BGP Routing*. For a general description of multicasting, see *JUNOSe Multicast Routing Configuration Guide, Chapter 5, Configuring IPv4 Multicast*.

- VPN IPv4—If you specify the VPN-IPv4 (also known as VPNv4) address family, you can configure the router to provide IPv4 VPN services over an MPLS backbone. These VPNs are often referred to as BGP/MPLS VPNs. For detailed information, see *Chapter 3, Configuring BGP-MPLS Applications*.

- Unicast IPv6—If you specify the IPv6 unicast address family, you can configure the router to exchange unicast IPv6 routes or unicast IPv6 routes in a specified VRF. For a description of IPv6, see *JUNOSe IP, IPv6, and IGP Configuration Guide, Chapter 2, Configuring IPv6*.

- Multicast IPv6—If you specify the multicast IPv6 address family, you can use BGP to exchange routing information about how to reach an IPv6 multicast source instead of an IPv6 unicast destination. For a general description of multicasting, see *JUNOSe Multicast Routing Configuration Guide, Chapter 5, Configuring IPv4 Multicast*.

- VPN IPv6—If you specify the VPN-IPv6 address family, you can configure the router to provide IPv6 VPN services over an MPLS backbone. These VPNs are often referred to as BGP/MPLS VPNs.

- L2VPN—If you specify the L2VPN address family, you can configure the PE router (L2VPNs) or VE router (VPLS) to exchange layer 2 network layer reachability information (NLRI) for all L2VPN (VPWS) or VPLS instances. Optionally, you can use the **signaling** keyword with the **address-family** command for the L2VPN address family to specify BGP signaling of L2VPN reachability information. Currently, you can omit the **signaling** keyword with no adverse effects. For a description of L2VPNs (VPWS), see *Chapter 11, Configuring L2VPNs*. For a description of VPLS, see *Chapter 8, Configuring VPLS*.

- Route-target—If you specify the route-target address family, you can configure the router to exchange route-target membership information to limit the number of routes redistributed among members. For a description of route-target filtering, see *Chapter 3, Configuring BGP-MPLS Applications*.

- VPLS—If you specify the VPLS address family, you can configure the router to exchange layer 2 NLRI for a specified VPLS instance. For a description of VPLS, see *Chapter 8, Configuring VPLS*.

- VPWS—If you specify the VPWS address family, you can configure the PE router to exchange layer 2 NLRI for a specified L2VPN (VPWS) instance. For a description of L2VPNs (VPWS), see *Chapter 11, Configuring L2VPNs*.

Any command issued outside the context of an address family applies to the unicast IPv4 address family by default.

To limit the exchange of routes to those from within the address family and to set other desired BGP parameters:

1. Access Router Configuration mode and create peers and peer groups. These peers and peer groups are in the default IPv4 address family.

   host1(config)#**router bgp 100**
   host1(config-router)#**neighbor 10.10.2.2 remote-as 100**
   host1(config-router)#**neighbor 10.10.3.3 remote-as 100**
   host1(config-router)#**neighbor ibgp peer-group**

2. In Router Configuration mode, create the address family within which the router exchanges addresses; this creation accesses Address Family Configuration mode.

   host1(config-router)#**address-family vpn4 unicast**

3. From within the address family, activate individual neighbors or peer groups to exchange routes from within the current address family. These peers or peer groups must first be created in the IPv4 address family.

   host1(config-router-af)#**neighbor ibgp activate**

4. If you have activated a peer group, from within the address family add peers as members of the peer group. These peers must first be created in the IPv4 address family.

   host1(config-router-af)#**neighbor 10.10.2.2 peer-group ibgp**
   host1(config-router-af)#**neighbor 10.10.3.3 peer-group ibgp**

5. From within the address family, configure BGP parameters for the address family.

6. Exit Address Family Configuration mode.

   host1:vr1(config-router-af)#**exit-address-family**

address-family

- Use to configure the router or VRF to exchange IPv4 or IPv6 addresses by creating the specified address family.
- IPv4 and IPv6 addresses can be exchanged in unicast, multicast, or VPN mode.
- The default setting is to exchange IPv4 addresses in unicast mode from the default router.

- Creating an address family for a VRF automatically disables both synchronization and automatic summarization for that VRF.
- This command takes effect immediately.
- Examples

  host1:vr1(config-router)#**address-family ipv4 multicast**
  host1:vr1(config-router)#**address-family ipv4 unicast**
  host1:vr1(config-router)#**address-family ipv4 unicast vrf vr2**
  host1:vr1(config-router)#**address-family vpn4 unicast**
  host1:vr1(config-router)#**address-family ipv6 unicast**

- Use the **no** version to disable the exchange of a type of prefix.

bgp default ipv4-unicast

- Use to configure all neighbors to exchange addresses in the IPv4 unicast address family.
- All neighbors must be activated with the **neighbor activate** command in the IPv4 address family.
- Example

  host1:vr1(config-router)#**bgp default ipv4-unicast**

- Affects only neighbors created after you issue the command. To affect existing neighbors created before you issued the command, you must use the **neighbor activate** command in the context of the IPv4 unicast address family.
- Use the **no** version to disable the exchange of IPv4 addresses on all neighbors.

exit-address-family

- Use to exit Address Family Configuration mode and access Router Configuration mode.
- Example

  host1:vr1(config-router-af)#**exit-address-family**

- There is no **no** version.

neighbor activate

- Use to specify a peer or peer group with which routes of the current address family are exchanged.
- A peer or peer group can be activated in more than one address family. By default, a peer is activated only for the IPv4 unicast address family.
- The peer or peer group must be created in unicast IPv4 before you can activate it in another address family.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- The address families that are actively exchanged over a BGP session are negotiated when the session is established.

- This command takes effect immediately. If dynamic capability negotiation was not negotiated with the peer, the session is automatically bounced so that the exchanged address families can be renegotiated in the open messages when the session comes back up.

  If dynamic capability negotiation was negotiated with the peer, BGP sends a capability message to the peer to advertise or withdraw the multiprotocol capability for the address family in which this command is issued. If a neighbor is activated, BGP also sends the full contents of the BGP routing table of the newly activated address family.

- Example

  host1:vr1(config-router-af)#**neighbor 192.168.1.158 activate**

- Use the **no** version to indicate that routes of the current address family are not to be exchanged with the peer. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

If you have configured some or all neighbors to be in the multicast or VPN-IPv4 address families, you can quickly configure all neighbors to be part of the IPv4 unicast address family by issuing the **bgp default ipv4-unicast** command.

Enabling Lenient Behavior

You can use the **neighbor lenient** command to enable the BGP speaker to attempt to recover from malformed packet errors and finite state machine errors generated by a peer. If BGP can recover from the error, it logs a warning message and attempts to maintain the session with the peer. The normal, nonlenient behavior is for the BGP speaker to send a notification message to the peer generating the error and to terminate the session. By default, lenient behavior is disabled.

neighbor lenient

- Use to enable a BGP speaker to be more tolerant of some errors generated by a peer, such as malformed BGP messages or finite state machine errors.
- The speaker attempts to recover from the errors and avoid bringing down the BGP session with the peer.
- Lenient behavior is disabled by default.
- Example

  host1(router-config)#**neighbor 10.12.45.23 lenient**

- Use the **no** version to restore the default condition, disabling lenient behavior.

Configuring Promiscuous Peers and Dynamic Peering

You can use the **neighbor allow** command to enable a peer group to accept incoming BGP connections from any remote address that matches an access list. Such a peer group is known as a promiscuous peer group; the member peers are sometimes referred to as promiscuous peers.

Promiscuous peers are useful when the remote address of the peer is not known ahead of time. An example is in B-RAS applications, in which interfaces for subscribers are created dynamically and the remote address of the subscriber is assigned dynamically from a local pool or by using RADIUS or some other method.

BGP automatically creates a dynamic peer when a peer group member accepts the incoming BGP connection. Dynamic peers are passive, meaning that when they are not in the established state, they will accept inbound connections but they will not initiate outbound connections. You cannot configure any attributes for the dynamic peers. You cannot remove a dynamic peer with the **no neighbor ip-address** command.

When a dynamic peer goes from the established state to the idle state for any reason, BGP removes the dynamic peer only if it does not go back to the established state within 1 minute. This delay enables you to see the dynamic peer in **show** command output; for example, you might want to see the reason for the last reset or how many times the session flapped.

While a dynamic peer is not in the established state, the **show ip bgp neighbor** command displays the number of seconds remaining until the dynamic peer will be removed.

If you have configured the **neighbor allow** command for multiple peer groups, when an incoming BGP connection matches the access list of more than one of these peer groups, the dynamic peer is created only in the first peer group. (BGP orders peer groups alphabetically by name.)

When the BGP speaker receives an open message from a dynamic peer, the remote AS number must match one of the following criteria; the connection is closed if it does not:

- If the peer group has a configured remote AS number, then the received AS number must be the same as the configured remote AS number.
- If the peer group does not have a configured AS number, then the received AS number must be consistent with the peer type of the peer group. Use the **neighbor peer-type** command to configure the type of the peer-group.

If a peer group has been configured with a peer type but not a remote AS, then the remote AS for dynamic peers is not known until an open message has been received from the peer. Until then, **show** commands display the remote AS as “?” or “unknown.”

Static peers that you configure with the **neighbor remote-as** or **neighbor peer-group** commands take precedence over the dynamic peers created as a result of the **neighbor allow** command. If the remote address of an incoming BGP connection matches both a static peer and the access list, the static peer is used and no dynamic peer is created. If you configure a new static peer while a dynamic peer for the same remote address already exists, BGP automatically removes the dynamic peer.

You can optionally specify the maximum number of dynamic peers that BGP can create for the peer group. There is no default maximum. In the absence of a specified maximum, the number of dynamic peers allowed is determined by the available memory and CPU. Dynamic peers consume about the same resources as static peers.

When the maximum number of dynamic peers has been created for a peer group, BGP rejects all subsequent connection attempts for that group. This behavior means that you can specify a maximum to help protect against denial-of-service attacks that attempt to create many dynamic peers to overwhelm your router resources.

BGP generates a log message whenever a dynamic peer is created, rejected because the maximum has been reached, or removed. BGP maintains counters for each peer group for the current number of dynamic peers, the highest number of concurrent dynamic peers ever reached, and the number of times a dynamic peer was rejected because the maximum was reached.

Because dynamic peers always fully inherit their configuration from a peer group, any features that are available for peers but not for peer group members are not supported for the dynamic peers. Currently, only ORFs are not supported for peer group members and therefore are not supported for dynamic peers.

clear bgp ipv6 dynamic-peers

clear ip bgp dynamic-peers

- Use to remove all dynamic peers in the specified scope.
- You can specify the IP address of a BGP neighbor or the name of a BGP peer group as the scope. For IPv4 only, you can also include a VRF in the scope.
- Use the asterisk (*) to remove all BGP dynamic peers.
- This command takes effect immediately.
- Example

  host1#**clear ip bgp 192.168.1.158 vrf boston5 dynamic-peers**

- There is no **no** version.

neighbor allow

- Use to configure a peer group to accept incoming BGP connections from any remote address that matches the specified access list.
- When the BGP connection is accepted, a dynamic peer is automatically created.
- This command is supported only for peer groups; it is not available for individual peers. These dynamic peers are not displayed by the **show configuration** command and are not stored in NVS. However, the dynamic peers are displayed by **show** commands that display information about BGP peers, such as **show ip bgp neighbors**, **show ip bgp summary**, and so on.
- Incoming connections that match the specified access list are rejected if no peer type has been configured for the peer group.

- This command takes effect immediately. Any existing dynamic BGP sessions that are no longer allowed by the new configuration are removed automatically and immediately. Preexisting dynamic peers that are still allowed by the new configuration are not affected.
- All the members of the peer group inherit the characteristic configured with this command. It cannot be overridden for a specific peer, because the command applies only to peer groups.
- Example

  host1(config-router)#**neighbor promispeers allow remotelist1 max-peers 1023**

- Use the **no** version to remove the configuration from the peer group.

Configuring Passive Peers

You can configure BGP to be passive regarding specific peers, meaning that the BGP speaker will accept inbound BGP connections from the peers but will never initiate an outbound BGP connection to the peers. This passive status conserves CPU and TCP connection resources when the neighbor does not exist.

For example, suppose you preprovision a router before installation with a large number of customer circuits to minimize the configuration changes you might have to make to the router. Any peers that do not exist will consume resources as BGP repeatedly attempts to establish a session with them.

If instead you initially configure the router as passive for those peers, BGP will not attempt to establish sessions to those peers but will wait until these remote peers initiate a session, thus conserving CPU resources.

If you configure both sides of a BGP session as passive, then the session can never come up because neither side can initiate the connection.

neighbor passive

- Use to configure the BGP speaker to only accept inbound BGP connections from the specified peer and never initiate outbound connections to that peer.
- This command takes effect immediately. If the session is not yet established, BGP immediately stops initiating outbound connections to the peer. If the session is already established, it is not bounced regardless of which side initiated the connection.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- Example

  host1(config-router)#**neighbor 10.12.3.5 passive**

- Use the **no** version to restore the default condition, permitting the initiation of outbound connections to the peer.

Advertising Routes

Each BGP speaker advertises to its peers the routes to prefixes that it can reach. These routes include:

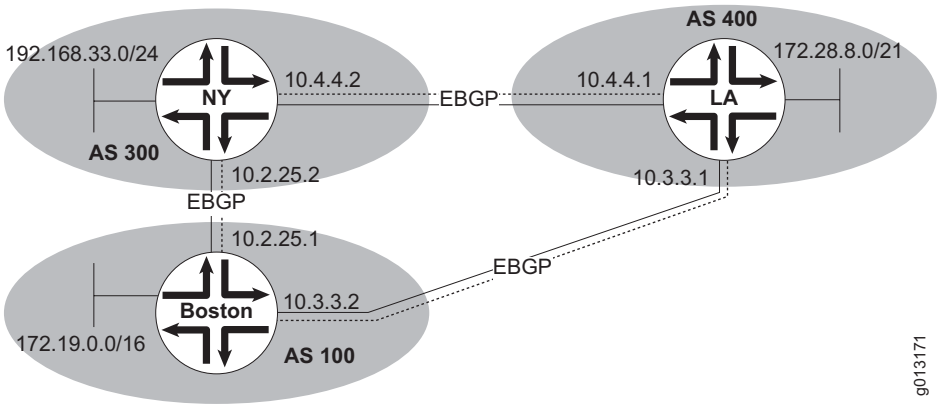
- Routes to prefixes originating within the speaker's AS
- Routes redistributed from another protocol, including static routes

By default, BGP does not advertise any route unless the router's IP routing table also contains the route.

Prefixes Originating in an AS

Use the **network** command to configure a router with the prefixes that originate within its AS. Thereafter the router advertises these configured prefixes with the origin attribute set to IGP. See *Understanding the Origin Attribute* on page 114 for more information about origins. Figure 13 shows a network structure of three autonomous systems, each with a router that originates certain prefixes.

Figure 13: Prefixes Originating in an AS



The following commands configure router NY:

```
host1(config)#router bgp 300
host1(config-router)#neighbor 10.2.25.1 remote-as 100
host1(config-router)#neighbor 10.4.4.1 remote-as 400
host1(config-router)#network 192.168.33.0 mask 255.255.255.0
```

The following commands configure router Boston:

```
host2(config)#router bgp 100
host2(config-router)#neighbor 10.2.25.2 remote-as 300
host2(config-router)#neighbor 10.3.3.1 remote-as 400
host2(config-router)#network 172.19.0.0
```

Notice that a mask was not specified for the prefix originating with router Boston. The *natural* mask is assumed for networks without a mask.

The following commands configure router LA:

```
host3(config)#router bgp 400
host3(config-router)#neighbor 10.3.3.2 remote-as 100
host3(config-router)#neighbor 10.4.4.2 remote-as 300
host3(config-router)#network 172.28.8.0 mask 255.255.248.0
```

network

- Use to specify the prefixes in its AS that the BGP speaker advertises.
- BGP advertises the specified prefix only if a non-BGP route to the prefix exists in the IP forwarding table. If the non-BGP route does not exist when you issue the **network** command, then BGP is notified as soon as the route becomes available in the IP routing table or IP tunnel routing table.
- For IPv4 addressing, specify a *network-number* and an optional *network-mask*. For IPv6 addressing, specify the IPv6 prefix.
- You can specify a route map to filter network routes or modify their path attributes.
- The default weight for network routes is 32768; use the **weight** keyword to modify the weight in the range 0–65535.
- Use the **backdoor** keyword to lower the preference of an EBGP route to the specified prefix by setting the administrative distance to that of an internal BGP route, 200. Use this option to favor an IGP backdoor route over an EBGP route to a specific network. BGP does not advertise the specified network. See *Configuring Backdoor Routes* on page 136 for more information.
- The next hop for the network is the next hop for the route contained in the routing table.
- This command takes effect immediately.
- Use the **no** version to remove the prefix.

Advertising Best Routes

By default, BGP selects from its routing table one best route to each destination. If BGP learned that best route from an internal peer, then the BGP speaker does not advertise a route to that destination to the speaker's internal peers.

In earlier software releases, the default behavior was for BGP to select two best routes to any destination. The best route learned from external (including confederation) peers was advertised to the speaker's internal peers. The best route learned from all sources was advertised to the speaker's external peers.

You can issue the **bgp advertise-external-to-internal** command to cause BGP to revert to advertising two potentially different routes to its peers. See *Selecting the Best Path* on page 103 for information about the process BGP uses to determine best routes.

bgp advertise-best-external-to-internal

- Use to cause BGP to select two best routes to every destination as follows:
  - For external peers, BGP selects the best route from the complete set of routes known to BGP.
  - For internal peers, BGP selects the best route from the set of routes BGP has received from external and confederation peers.
- Changes apply automatically whenever BGP subsequently runs the best-path decision process for a destination prefix; that is, whenever a best route is picked for a given prefix.
- The behavior enabled by this command is the default behavior for the E-series router running software releases lower than 5.0.0.
- The command is disabled by default.
- Example

  host2(config-router)#**bgp advertise-best-external-to-internal**

- Use the **no** version to restore the default condition, wherein BGP selects one best route for each destination from the complete set of routes; if the best route was received from an internal peer, no route to the destination is advertised to the internal peers.

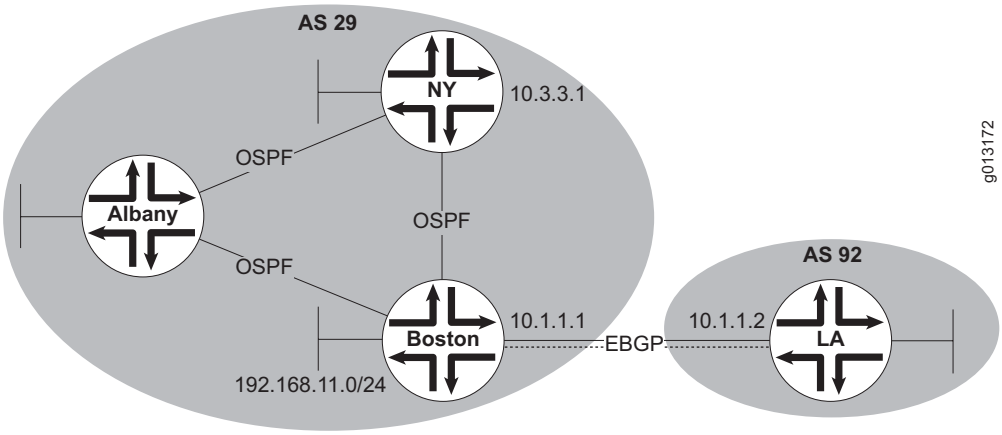
Redistributing Routes into BGP

BGP can learn about routes from sources other than BGP updates from peers. Routes known to other protocols can be *redistributed* into BGP. Similarly, routes manually configured on a router—static routes—can be redistributed into BGP. After the routes are redistributed, BGP advertises the routes. When you redistribute routes, BGP sets the origin attribute for the route to Incomplete. See *Understanding the Origin Attribute* on page 114 for more information about origins.

The following commands configure three static routes on router Boston and configure router Boston to redistribute the static routes and routes from OSPF into BGP for the network structure shown in Figure 14:

host2(config)#**ip route 172.30.0.0 255.255.0.0 192.168.10.12**
host2(config)#**ip route 172.16.8.0 255.255.248.0 10.211.5.7**
host2(config)#**ip route 192.168.4.0 255.255.254.0 10.14.147.2**
host2(config)#**router bgp 29**
host2(config-router)#**neighbor 10.1.1.2 remote-as 92**
host2(config-router)#**redistribute static**
host2(config-router)#**redistribute ospf**

Figure 14: Redistributing Routes into BGP



clear bgp ipv6 redistribution
clear ip bgp redistribution

- Use to reapply policy to routes that have been redistributed into BGP.
- This command takes effect immediately.
- There is no **no** version.

disable-dynamic-redistribute

- Use to halt the dynamic redistribution of routes that are initiated by changes to a route map.
- Dynamic redistribution is enabled by default.
- This command takes effect immediately.
- Example

  host1(config-router)#**disable-dynamic-redistribute**

- Use the **no** version to reenable dynamic redistribution.

redistribute

- Use to redistribute static routes and routes from other protocols into BGP.
- Specify the source protocol from which routes are being redistributed with one of the following keywords: **isis**, **ospf**, **static**, or **connected**. Use the **static** keyword to redistribute IP static routes. Use the **connected** keyword to redistribute routes that are established automatically by virtue of having enabled IP on an interface.
- You can specify a route map to filter the redistribution of routes from the source routing protocol into BGP. If you do not specify the **route-map** option, all routes are redistributed.
- Use the **metric** keyword to set the multiexit discriminator (MED) for routes redistributed into BGP. The default MED is the value of the IGP metric for the redistributed route.

- Use the **weight** keyword to set the weight for routes redistributed into BGP in the range 0–65535. The default weight is 32768.
- You can specify the type(s) of OSPF routes to redistribute into BGP: internal routes (**ospf match internal**), external routes of metric type 1 (**ospf match external 1**), or external routes of metric type 2 (**ospf match external 2**).
- This command takes effect immediately.
- Use the **no** version to end the redistribution of routes into BGP.

Redistributing Routes from BGP

If you have redistributed routes from BGP into an IGP, by default only EBGP routes are redistributed. You can issue the **bgp redistribute-internal** command followed by clearing all BGP sessions to permit the redistribution of IBGP routes in addition to EBGP routes.

NOTE: This default behavior does not apply to VPN routes. Redistribution of IBGP routes (routes received from an internal BGP peer) in a VRF is always enabled. You do not have to issue this command to enable redistribution of internal BGP routes in a VRF.

bgp redistribute-internal

- Use to enable the redistribution of IBGP routes in addition to EBGP routes into IGPs configured for BGP route redistribution.
- Redistribution of IBGP routes is disabled by default, except within a VRF where IBGP routes are always redistributed.
- You must clear all BGP sessions after issuing this command for it to take effect.
- Example

```
host1(config-router)#bgp redistribute-internal
host1(config-router)#exit
host1(config)#exit
host1(config)#clear ip bgp *
```

- All IBGP and EBGP routes subsequently placed in the IP routing table are redistributed to IGPs that have route redistribution enabled.

  To authorize redistribution of routes that are already present in the IP routing table, you must use the **clear ip bgp *** command (this command will bounce the BGP sessions) or the **clear ip routes *** command to reinstall BGP routes in the IP routing table.

- Use the **no** version to restore the default of permitting the redistribution only of EBGP routes.

Configuring a Default Route

Default routes can provide backup routes if primary connections fail or if the route information for a destination is unknown. A router uses the default route in its IP forwarding table to route traffic toward a destination for which no routing entry exists. The accepted BGP convention is to represent a default route by the network prefix 0.0.0.0/0.

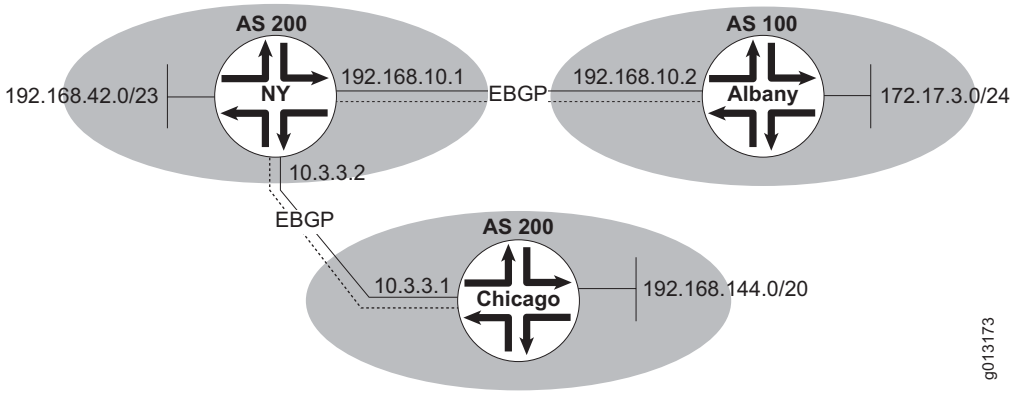
Advertising Default Routes

If you want a router to serve as a default destination for traffic from other routers that do not know where to forward traffic, you can configure the router to advertise a default route. Use the **neighbor default-originate** command to specify the neighbors to which this router will advertise the default route. Said another way, these neighbors will dynamically learn the default route from the router you configure.

If you issue the **neighbor default-originate** command, BGP sends the default route to that neighbor regardless of whether the default route exists in the IP forwarding table.

In Figure 15, router NY originates the default route 0.0.0.0/0 to router Albany only. Router Chicago does not receive the default route.

Figure 15: Advertising a Default Route



To configure router NY:

```
host1(config)#router bgp 200
host1(config-router)#network 192.168.42.0 mask 255.255.254.0
host1(config-router)#neighbor 10.3.3.1 remote-as 300
host1(config-router)#neighbor 192.168.10.2 remote-as 100
host1(config-router)#neighbor 192.168.10.2 default-originate
```

You can also specify a route map to modify the attributes of the default route. If the default route does not match the route map, then the default route is not advertised.

Redistributing Default Routes

By default, the **redistribute** command does not permit a default route to be redistributed into BGP. You can use the **default-information originate** command to override this behavior and permit the redistribution of default routes into BGP.

default-information originate

- Use to enable the redistribution of default routes into BGP.
- Use the **route-map** keyword to specify outbound route maps to apply to the default route. The route map can modify the attributes of the default route.

- This command takes effect immediately. However, if the contents of the route map specified with this command change, the new route map may or may not take effect immediately. If the **disable-dynamic-redistribute** command has been configured, you must issue the **clear ip bgp redistribution** command to apply the changed route map.
- Outbound policy configured for the neighbor (using the **neighbor route-map out** command) is applied to default routes that are advertised because of the **default-information originate** command.
- Policy specified by a route map with the **default-information originate** command is applied at the same time as the policy for redistributed routes, before any outbound policy for peers.
- Example

  host1(config)#**router bgp 100**
  host1(config-router)#**default-information originate**

- Use the **no** version to restore the default, preventing the redistribution of default routes.

Setting a Static Default Route

You might not want your routers to rely on dynamically learned default routes. Instead, you might prefer to specify a static default route that your routers use to forward traffic when they do not have a routing entry for a destination. Use the **ip route** command to configure a default route on a router. The static route can point to a network number, an IP address, or a physical interface. You can add a distance value to give preference to a specific static route when multiple entries exist for the same route.

Suppose that in Figure 16, router KC has been configured to advertise a default route to router Chicago:

host1(config)#**router bgp 62**
host1(config-router)#**network 172.17.24.0 mask 255.255.248.0**
host1(config-router)#**neighbor 10.8.3.1 remote-as 21**
host1(config-router)#**neighbor 10.8.3.1 default-originate**

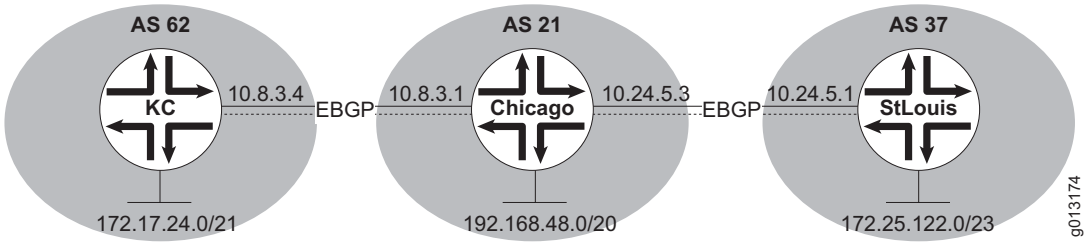
You prefer that router Chicago send traffic with unknown destinations to router StLouis, so you configure a static default route on router Chicago:

host2(config)#**router bgp 21**
host2(config-router)#**network 192.168.48.0 mask 255.255.240.0**
host2(config-router)#**neighbor 10.8.3.4 remote-as 62**
host2(config-router)#**neighbor 10.24.5.1 remote-as 37**
host2(config-router)#**exit**
host2(config)#**ip route 0.0.0.0 0.0.0.0 172.25.122.0**

Router StLouis is configured to advertise network 172.25.122.0/23 to router Chicago:

host3(config)#**router bgp 37**
host3(config-router)#**network 172.25.122.0 mask 255.255.254.0**
host3(config-router)#**neighbor 10.24.5.3 remote-as 21**

Figure 16: Setting a Static Default Route



ip route

- Use to establish static routes.
- Use the **no** version to remove static routes.

neighbor default-originate

- Use to cause a BGP speaker (the local router) to send the default route 0.0.0.0/0 to a neighbor for use as a default route.
- Use the **route-map** keyword to specify outbound route maps to apply to the default route. The route map can modify the attributes of the default route.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override the characteristic for a specific member of the peer group.
- Outbound policy configured for the neighbor (using the **neighbor route-map out** command) is not applied to default routes that are advertised because of the **neighbor default-originate** command.
- This command takes effect immediately.
- Use the **no** version to prevent the default route from being advertised by BGP. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Setting the Minimum Interval Between Routing Updates

You can use the **neighbor advertisement-interval** command to set the minimum interval between the sending of BGP updates. Lower values for the advertisement interval cause route changes to be reported more quickly, but may cause routers to use more bandwidth and processor time.

In the following example, the minimum time between sending BGP routing updates is set to 5 seconds:

```
host1(config)#router bgp 100
host1(config-router)#neighbor 10.2.2.2 advertisement-interval 5
```

neighbor advertisement-interval

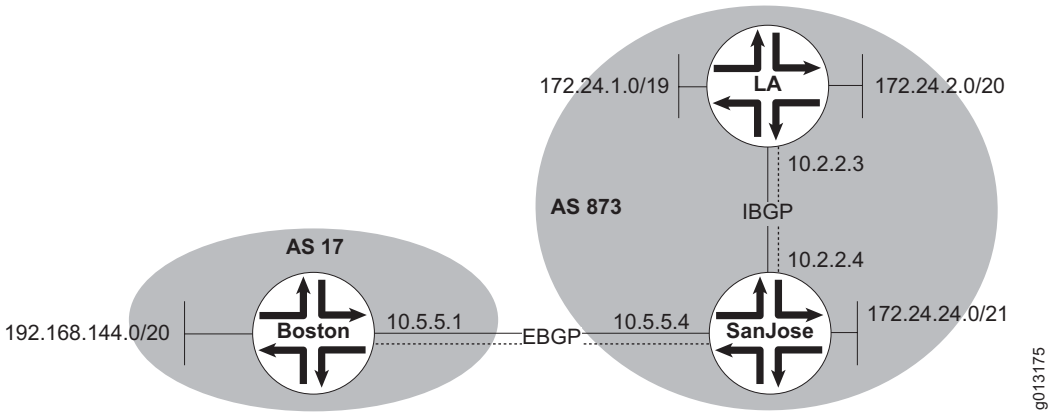
- Use to set the minimum interval between the sending of BGP updates for a given prefix.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately.
- Use the **no** version to restore the default, 30 seconds for external peers and 5 seconds for internal peers.

Aggregating Routes

Aggregation applies only to routes that are present in the BGP routing table. BGP advertises an aggregate route only if the routing table contains at least one prefix that is more specific than the aggregate. You aggregate IPv4 routes by specifying the aggregate IP address, and IPv6 routes by specifying the aggregate IPv6 prefix.

Figure 17 illustrates an IPv4 network structure where you might use aggregation. The following commands configure router LA and router Snakes so that router Snakes advertises an IPv4 aggregate route, 172.24.0.0/16, for the more specific prefixes 172.24.1.0/24, 172.24.2.0/24, and 172.24.24.0/21.

Figure 17: Configuring Aggregate Addresses



To configure router LA:

```
host1(config)#router bgp 873
host1(config-router)#neighbor 10.2.2.4 remote-as 873
host1(config-router)#network 172.24.1.0 mask 255.255.255.0
host1(config-router)#network 172.24.2.0 mask 255.255.255.0
```

To configure router SanJose:

```
host2(config)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#network 172.24.24.0 mask 255.255.248.0
host2(config-router)#aggregate-address 172.24.0.0 255.255.224.0
```

As configured above, router SanJose advertises the more specific routes as well as the aggregate route to router Boston. Alternatively, you can use the **summary-only** option to configure router SanJose to suppress the more specific routes and advertise only the aggregate route:

```
host2(config)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#network 172.24.24.0 mask 255.255.248.0
host2(config-router)#aggregate-address 172.24.0.0 255.255.224.0
summary-only
```

Each of these configurations sets the atomic-aggregate attribute in the aggregate route. This attribute informs recipients that the route *is* an aggregate and must not be deaggregated into more specific routes.

Aggregate routes discard the path information carried in the original routes. To preserve the paths, you must use the **as-set** option. This option creates an AS-Set that consists of all the AS numbers traversed by the summarized paths. The AS-Set is enclosed within curly brackets; for example, {3, 2}. Each AS number appears only once, even if it appears in more than one of the original paths. If you use the **as-set** option, the atomic-aggregate attribute is not set for the aggregated route. The following commands configure router SanJose to aggregate the routes while preserving the path information:

```
host2(config)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#network 172.24.24.0 mask 255.255.248.0
host2(config-router)#aggregate-address 172.24.0.0 255.255.224.0
summary-only as-set
```

If you do not want to aggregate all more specific routes, you can use a route map to limit aggregation. Consider Figure 17 again. Suppose you do not want router SanJose to aggregate prefix 172.24.48.0/20. The following commands show how you can configure a route map on router SanJose to match this prefix, and how to invoke the route map with the **advertise-map** option:

```
host2(config)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#neighbor 10.2.2.3 route-map lmt_agg in
host2(config-router)#network 172.24.24.0 mask 255.255.248.0
host2(config-router)#aggregate-address 172.24.0.0 255.255.224.0
advertise-map lmt_agg
host2(config-router)#exit
host2(config)#route-map lmt_agg permit 10
host2(config-route-map)#match ip address 1
host2(config-route-map)#exit
host2(config)#access-list 1 permit 172.24.48.0 0.240.255.255
```

You can use the **attribute-map** option to configure attributes for the aggregated route. In Figure 17, suppose that router LA has been configured to set the community attribute for route 172.24.160.0/19 to no-export. This attribute is passed along to router SanJose and preserved when the aggregate route is created. As a result, the aggregate route is not advertised outside the AS. The following commands demonstrate how to configure router SanJose to prevent the aggregate from not being advertised:

```
host2(config)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#network 172.24.24.0 mask 255.255.248.0
host2(config-router)#aggregate-address 172.24.0.0 255.255.224.0
attribute-map conf_agg_att
host2(config-router)#exit
host2(config)#route-map conf_agg_att permit 10
host2(config-route-map)#set community no-export
```

aggregate-address

- Use to create an aggregate entry in a BGP routing table that summarizes more specific routes.
- For IPv4 routes, you must specify an aggregate IP address (*address*) and aggregate IP mask (*mask*). For IPv6 routes, you must specify an aggregate IPv6 prefix (*ipv6Prefix*).
- The optional **as-set** keyword preserves path information by creating an AS-Set that contains all the AS numbers traversed by the aggregated routes.

NOTE: Do not use the **as-set** keyword when you have many paths to aggregate. If you do, the aggregated route is continually withdrawn and reupdated as AS-path reachability information changes for the summarized routes.

- The **summary-only** keyword advertises only the aggregate route; it suppresses the advertisement of all more specific routes. Contrast with the **suppress-map** keyword.
- The **suppress-map** keyword enables you to specify a route map to filter particular routes covered by the aggregate that will be suppressed. Contrast with the **summary-only** keyword.

NOTE: If you want to suppress advertisements only to certain neighbors, you can—with caution—use the **neighbor distribute-list** command. If a more specific route leaks out, all BGP speakers will prefer that route over the less specific aggregate you are generating (using longest-match routing).

- The **advertise-map** keyword enables you to specify the advertise-map-tag, a string of up to 32 characters that identifies the route map that sets the routes to create AS-Set origin communities.
- The **attribute-map** keyword enables you to specify the attribute-map-tag, a string of up to 32 characters that identifies the route map that sets the attributes of the aggregate route.

- This command takes effect immediately.
- Use the **no** version to remove the aggregate route entry from the routing table.

Advertising Inactive Routes

Under normal circumstances, routes that are not being used to forward traffic—*inactive* routes—are not advertised to peers unless synchronization is enabled. For example, suppose a BGP speaker receives a route to a particular prefix, determines that it is the best route to the prefix, and stores the route in the IP routing table (sometimes known as the forwarding information base, or FIB). This route might not be used for forwarding to that prefix; for example, if you have configured a static route to the same destination prefix. Because static routes have better administrative distances than BGP received routes, IP will use the static route rather than the BGP received route for forwarding traffic to that prefix. The BGP received route is inactive and is not advertised to peers. You can use the **bgp advertise-inactive** command to enable the advertisement of inactive received routes.

bgp advertise-inactive

- Use to enable the BGP speaker to advertise inactive routes—best routes in the IP forwarding table that are not being used to forward traffic. This feature is disabled by default.
- Issuing this command does not affect the BGP rules for best route selection, or how BGP populates the IP forwarding table.
- Example

  host1(config-router)#**bgp advertise-inactive**

- The new value is applied to all routes that are subsequently placed in the IP routing table.

  To apply the new value to routes that are already present in the IP routing table, you must use the **clear ip bgp *** command (this command will bounce the BGP sessions).

- Use the **no** version to prevent the advertising of received BGP routes unless one or both of the following are true:
  - The received route is in the BGP forwarding table and is being used to forward traffic (the route is active).
  - Synchronization is enabled.

Verifying an AS Path

You can use the **bgp enforce-first-as** command to cause BGP to compare the first AS in the AS-path of a received route with the configured remote AS number of that EBGP peer. If the check fails, BGP returns a notification message to the peer.

bgp enforce-first-as

- Use to cause BGP to determine whether the first AS in the AS path of a route received from an EBGP peer matches the remote AS number of that peer.
- If the AS does not match, BGP sends a notification to the peer with the error code "update message error" and error subcode "malformed as-path."
- This feature is disabled by default.
- Example

  host1(config-router)#**bgp enforce-first-as**

- Causes BGP to check the AS path of all routes received after you issue the command.

  To apply the new behavior to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

- Use the **no** version to prevent the AS comparison from taking place.

Advertising IPv4 Routes Between IPv6 BGP Peers

When an IPv6 network connects two separate IPv4 networks, you can use IPv6 to advertise the IPv4 routes over the BGP session, using TCP IPv6 as the transport mechanism. Similarly, you can advertise IPv6 routes between two IPv4 peers over their BGP session.

Configure the peers by using IPv6 addresses within the IPv4 unicast address family. You can set the IPv4 next hop with a static route or by configuring an inbound or outbound route map. This action overrides the IPv4 next hop that is advertised to the peer for IPv4 routes over BGP IPv6 peers.

If you do not use the route map, then the advertised IPv4 next hop is set to the BGP router ID. That value generally makes the next hop unreachable by the other BGP IPv6 peer.

```
host1(config)#router bgp 100
host1(config-router)#neighbor 21:1 remote-as 200
host1(config-router)#route-map my-v4-nexthop
host1(config-router)#set ip next-hop 10.13.5.1
host1(config-router)#address-family ipv4 unicast
host1(config-router-af)#neighbor 21:1 activate
host1(config-router-af)#neighbor 21:1 route-map my-v4-nexthop out
```

Advertising Routes Conditionally

By default, a BGP speaker advertises the best routes in its routing table to its peers. However, in some circumstances, you might prefer that some routes be advertised to a peer or peer group only when another route is in the BGP routing table, or only when that route is not in the routing table. BGP conditional advertisement enables you to control route advertisement without having to rely on only the best routes.

For example, in a multi-homed network, you might want to advertise certain prefixes to one of the providers when a failure occurs in the peering session with a different provider, or when there is only partial reachability to that peer.

In other cases, the advertisement to a peer of certain routes might be useful only in the event that some other routes are present in the BGP routing table.

You can use the **neighbor advertise-map** command with route maps to configure conditional advertisement of BGP routes to a peer or peer group within an address family. BGP conditional advertisement does not create routes. The routes specified by the route map in the **neighbor advertise-map** command must already be present in the BGP routing table.

BGP conditional advertisement is supported in only the following address families:

- Unicast IPv4
- Unicast IPv6
- Multicast IPv4
- Multicast IPv6
- VPNv4 unicast
- VPNv6 unicast

NOTE: For VPNv4 unicast and VPNv6 unicast address families, we recommend that you include a **match extcommunity** clause to match a route with a specific route target. However, conditional advertisement in these address families can sometimes result in unintended behaviors: advertisement of or based on an incorrect VPN route or a non-VPN route.

BGP conditional advertisement is not supported in the following address families:

- L2VPN
- Route-target
- VPLS
- VPWS

Use the **exist-map** keyword when you want a route advertised only when another route is present. The determining route must match the specified route map. If the route map you specify with the **exist-map** keyword references multiple routes, only one of those routes needs to be in the routing table to trigger the conditional advertisement.

Use the **non-exist-map** keyword when you want a route advertised only when another route is absent. The determining route must match the specified route map. If the route map you specify with the **non-exist-map** keyword references multiple routes, all of those routes must be absent to trigger the conditional advertisement.

You can optionally specify a sequence number for the advertise route map that matches the determining route. The sequence number specifies the order in which the advertise route maps are processed. It indicates the position the specified advertise route map has in the list of all advertise route maps that are configured for a particular neighbor within the same address family.

If you do not specify a sequence number, the position of the advertise route map is considered to be the sum of the current largest sequence number plus five. An advertise route map with a lower sequence number has a higher priority and is processed before one with a higher sequence number.

If the route matches more than one advertise route map, only the first matching advertise route map, based on the sequence, controls the advertisement of a BGP route.

You can configure a maximum of 50 advertise maps for a given peer or peer group in an address family. However, the name and sequence number for the advertise route map must be unique for each entry. BGP applies any policy specified by the advertise map to the conditionally advertised routes before outbound policy specified for the neighbor is applied.

The route maps referenced by the **neighbor advertise-map** command must include a **match ip-address** clause. You can also include additional match clauses. All **match** commands supported by existing outbound policies are supported. The additional clauses are useful when you want to match only on a specific route with a specific set of attributes. Only the **permit** keyword is acted on in a match clause. The **deny** keyword is ignored. Only exact matching of a prefix referenced by exist maps or non-exist maps is supported. Consequently a range specified by the **ge** or **le** keyword in the prefix list referenced by these route maps is ignored.

Clauses in a route map that include **set** commands or the **match-set summary prefix-tree** command are ignored. To change the attributes of conditionally advertised routes, you must use outbound routing policy.

If the contents of a referenced route map are changed, the new route map takes effect automatically.

neighbor advertise-map

- Use to specify a peer or peer group within the current address family to which routes specified by a route map are advertised conditionally, depending on whether a second route map is matched by some other routes in the BGP routing table.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. This characteristic cannot be overridden for individual members of the peer group.
- This command takes effect immediately.

- Example

  host1(config-router-af)#**neighbor 192.168.2.2 advertise-map advertiseroutes exist-map matchroute sequence 10**
- Use the **no** version to remove the conditions set for advertising to the peer or peer group the routes specified by the route map. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

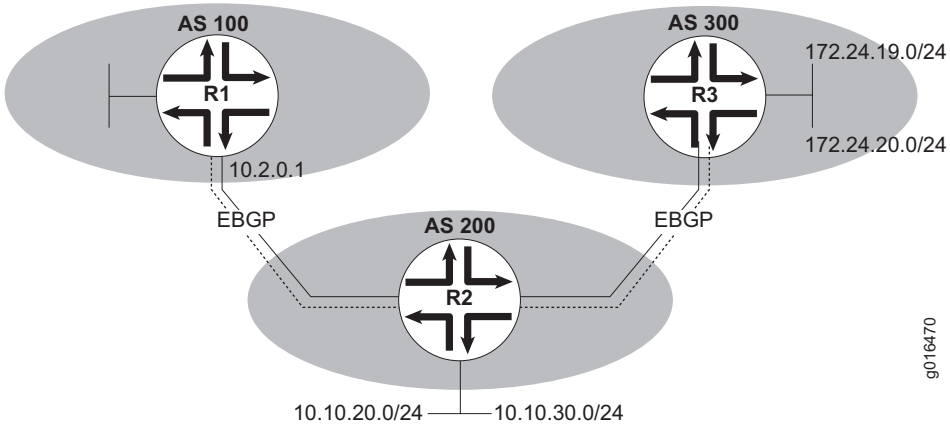
Advertising a Route Only When Another Route is Present

You can use the **exist-map** keyword with the **neighbor advertise-map** command to advertise a route only when the routing table contains some other particular route.

In the network shown in Figure 11, router 2 (R2) has established BGP sessions with both router 1 (R1) and router 3 (R3). The plan is for router 2 to send router 1 an advertisement for the route to prefix 10.10.20.0/24 only if router 2 has received a route to prefix 172.24.19.0/24 from router 3.

Alternatively, if the route to prefix 172.24.20.0 has been installed in the BGP routing table on router 2, then router 2 advertises to router 1 the route to prefix 10.10.30.0. In this case, the route does not have to be learned from router 3.

Figure 18: Advertising a Route When Another Route is Present



The following commands represent a partial configuration of router R2:

```
host1(config)#router bgp 200
host1(config-router)#address-family ipv4 unicast
host1(config-router-af)#neighbor 10.2.0.1 remote-as 100
host1(config-router-af)#neighbor 10.2.0.1 advertise-map advertisetoR1 exist-map
trigger1 sequence 10
host1(config-router-af)#neighbor 10.2.0.1 advertise-map alternatetoR1 exist-map
trigger2
host1(config-router-af)#exit
host1(config-router)#exit
!
```

```
!Configure route map to send one route to R1
!
host1(config)#access-list 77 permit 10.10.20.0 0.0.0.255
host1(config)#route-map advertisetoR1 permit 10
host1(config-route-map)#match ip address 77
host1(config-route-map)#exit
!
!Configure route map to match one trigger route from R3
!
host1(config)#ip as-path access-list 1 permit ^300
host1(config)#access-list 70 permit 172.24.19.0 0.0.0.255
host1(config)#route-map trigger1 permit 10
host1(config-route-map)#match ip address 70
host1(config-route-map)#match as-path 1
host1(config-route-map)#exit
!
!Configure route map to send alternate route to R1
!
host1(config)#access-list test permit 10.10.30.0 0.0.0.255
host1(config)#route-map alternatetoR1 permit 10
host1(config-route-map)#match ip address test
host1(config-route-map)#exit
!
!Configure route map to match alternate route from R3
!
host1(config)#access-list check permit 172.24.20.0 0.0.0.255
host1(config)#route-map trigger2 permit 10
host1(config-route-map)#match ip address check
host1(config-route-map)#exit
```

The **match as-path** clause in the route map referenced by the **exist-map** keyword ensures that router 2 sends router 1 the route to prefix 10.10.20.0 only if a route to 172.24.19.0/24 with an AS path of 300 is present in the BGP routing table. Similarly, you can impose additional restraints by including any other **match** clause that is supported by an existing outbound policy.

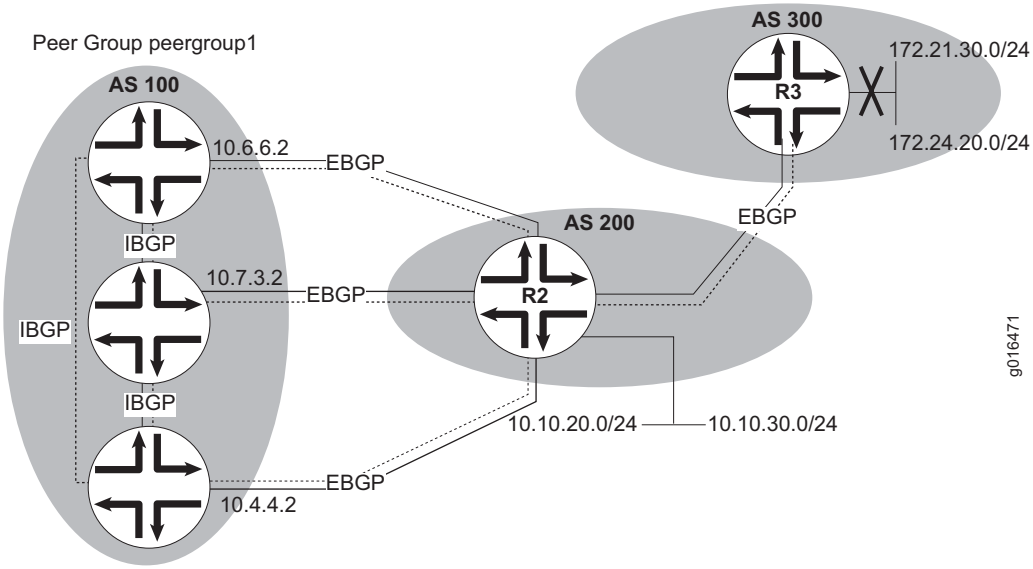
In this configuration, the condition1 route map has a sequence number of ten. Advertise route maps configured for this peer within the same address family and a lower sequence number are processed before the condition1 route map. The condition2 route map has no sequence number configured, thus giving the route map a sequence number of 15 and ensuring that condition2 is processed after the condition1 route map.

Advertising a Route Only When Another Route is Absent

You can use the **non-exist-map** keyword with the **neighbor advertise-map** command to advertise a route only when the BGP routing table does not contain some other particular route.

In the network shown in Figure 19, router R2 has established BGP sessions with both router R1 and router R3. The plan is for router R2 to send peergroup1 an advertisement for the route to prefix 10.10.30.0/24 only if the route to prefix 172.24.20.0/24 is not present in the BGP routing table. Alternatively, if router R2 has not received a route to prefix 172.21.30.0 from router R3, then router R2 advertises to peergroup1 the route to prefix 10.10.20.0. In this sample network, router R3 advertises neither of the routes to router R2. Consequently, router R2 advertises both 10.10.20.0/24 and 10.10.30.0/24 to peergroup1.

Figure 19: Advertising a Route When Another Route is Absent



The following commands configure router R2:

```
host1(config)#router bgp 200
host1(config-router)#neighbor peergroup1 peer-group
host1(config-router)#neighbor peergroup1 remote-as 100
host1(config-router)#neighbor 10.6.6.2 peer-group peergroup1
host1(config-router)#neighbor 10.7.3.2 peer-group peergroup1
host1(config-router)#neighbor 10.4.4.2 peer-group peergroup1
host1(config-router)#neighbor peergroup1 advertise-map advertisetoPG1
non-exist-map condition1 sequence 5
host1(config-router)#neighbor peer-group1 advertise-map alternatetoPG1
non-exist-map condition2
host1(config-router)#exit
host1(config)#ip as-path access-list 1 permit ^300
!
!Configure route map to send one route to peergroup1
!
host1(config)#access-list 77 permit 10.10.30.0 0.0.0.255
host1(config)#route-map advertisetoPG1 permit 10
host1(config-route-map)#match ip address 77
host1(config-route-map)#exit
!
!Configure route map to match one trigger route
!
host1(config)#access-list 70 permit 172.24.20.0 0.0.0.255
```

```
host1(config)#route-map condition1 permit 10
host1(config-route-map)#match ip address 70
host1(config-route-map)#exit
!
!Configure route map to send alternate route to peergroup1
!
host1(config)#access-list allow permit 10.10.20.0 0.0.0.255
host1(config)#route-map alternatetoPG1 permit 10
host1(config-route-map)#match ip address allow
host1(config-route-map)#exit
!
!Configure route map to match an alternate trigger route
!
host1(config)#access-list test permit 172.21.30.0 0.0.0.255
host1(config)#route-map condition2 permit 10
host1(config-route-map)#match ip address test
host1(config-route-map)#match as-path 1
host1(config-route-map)#exit
```

In this configuration, the condition1 route map has a sequence number of five, placing it high in the list of all configured advertise route maps for this peer group within the same address family. The condition2 route map has no sequence number configured, thus placing it at the bottom of the route map list.

In this configuration, the condition1 route map has a sequence number of ten. Route maps configured for this peer group within the same address family and a lower sequence number are processed before the condition1 route map. The condition2 route map has no sequence number configured, thus giving the route map the sequence number of ten and ensuring that condition2 is processed after the condition1 route map.

Advertising a Default Route Only When Another Route Is Present

In some circumstances, you might want to control the advertisement of a default route based on the reachability of an IGP prefix. Because conditional advertisement tracks the BGP routing table rather than the IP routing table, the prefixes that govern the advertisement (the conditional prefixes) must be present in the BGP routing table. In order to use the IGP prefix as a condition, you must import the IGP prefixes into the BGP routing table. You must also configure the origination of the default route.

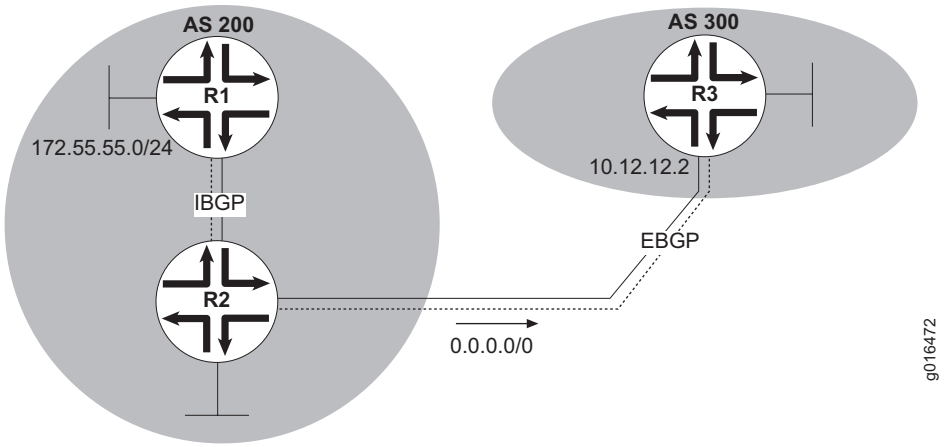
In the network shown in Figure 20, router R2 has an EBGP session with router R3 and has an IGP session with router R1. Suppose you want to advertise the default route to router R3 based on the reachability of an IGP prefix, 172.55.55.0/24, on router R2.

On router R2, configure a conditional advertisement entry for the neighbor R3. The advertise map must match the default route and the route map referenced by the **exist-map** keyword must match the imported IGP prefix.

In case router R3 must not learn about the IGP prefix 172.55.55.0/24, you must configure an additional outbound route map to deny this prefix so that it is not advertised to router R3.

With this configuration, the default route is advertised to router R3 only when the IGP prefix 172.55.55.0/24 is reachable on router R2. The default route is withdrawn if this prefix becomes unreachable.

Figure 20: Advertising a Default Route When Another Route is Present



The following commands configure router R2:

```
host1(config)#ip prefix-list default permit 0.0.0.0/0
host1(config)#route-map default permit 10
host1(config-route-map)#match ip address prefix-list default
host1(config-route-map)#exit
host1(config)#ip prefix-list test-default permit 172.55.0.0/16
host1(config)#route-map test permit 10
host1(config-route-map)#match ip address prefix-list test-default
host1(config-route-map)#exit
host1(config)#route-map outbound deny 10
host1(config-route-map)#match ip address prefix-list test-default
host1(config-route-map)#exit
host1(config)#route-map outbound permit 20
host1(config-route-map)#exit
host1(config)#router bgp 200
host1(config-router)#neighbor 10.12.12.2 remote-as 300
host1(config-router)#network 172.55.55.0/24
host1(config-router)#aggregate-address 172.55.0.0/16 summary-only
host1(config-router)#neighbor 10.12.12.2 advertise-map default exist-map test
host1(config-router)#neighbor 10.12.12.2 default-originate
host1(config-router)#neighbor 10.12.12.2 route-map outbound out
host1(config-router)#exit
```

Configuring BGP Routing Policy

Routing policy determines how the router handles the routes it receives from and sends to BGP peers or other routing protocols. In many cases, routing policy consists of filtering routes, accepting certain routes, accepting and modifying other routes, and rejecting some routes. You can think of routing policy as a way to control the flow of routes into and out of the router.

You can use one or more of the following mechanisms to configure routing policy:

- Access lists
- Community lists
- Prefix lists
- Prefix trees
- Route maps

The remainder of this section provides detailed information about using these features with BGP. Before proceeding, please see *JUNOSe IP Services Configuration Guide, Chapter 1, Configuring Routing Policy*, for a thorough background on how these features work in general.

Types of BGP Route Maps

A route map consists of *match* clauses and *set* clauses. Match clauses, which consist of a **match** command, specify the attribute values that determine whether a route matches the route map. Set clauses, which consist of a **set** command, modify the specified attributes of routes that pass all match clauses in the route map.

BGP route maps can be applied to inbound routes, outbound routes, and redistributed routes. BGP route maps are of two types, those that support both **match** and **set** clauses, and those that support only **match** clauses.

The match-and-set route maps consist of the route maps configured with any of the commands listed in Table 15.

Table 15: Commands That Create Match-and-Set Route Maps

aggregate-address attribute-map	global import map
bgp dampening route-map	neighbor route-map in
export map	neighbor route-map out
import map	redistribute route-map
global export map	table-map

BGP supports the clauses listed in Table 16 for match-and-set route maps.

Table 16: Clauses Supported in BGP Match-and-Set Route Maps

match as-path	set as-path prepend
match community	set comm-list delete
match distance	set community
match extcommunity	set dampening
match ip address	set extcommunity
match ip next-hop	set ip next-hop
match level	set local-preference

Table 16: Clauses Supported in BGP Match-and-Set Route Maps (continued)

match metric	set metric
match metric-type	set metric-type
match route-type	set origin
match tag	set tag
	set weight

The match-only route maps consist of the route maps configured with any of the commands listed in Table 17. You can use any of the match clauses listed in Table 16 in these route maps. Set clauses have no effect in these route maps.

Table 17: Commands That Create Match-Only Route Maps

aggregate advertise-map	aggregate support-map

BGP does not support the clauses listed in Table 18. However, see *Applying Table Maps* on page 78 for exceptions for route maps applied with the **table-map** command.

Table 18: Clauses Not Supported in BGP Route Maps

set automatic-tag	set level
set distance	set route-type

match as-path

- Use to match an AS-path access list.
- The implemented weight is based on the first matched AS path.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match as-path pathlist5**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match community

- Use to match a community list.
- Supported for inbound and outbound route maps.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match community comm5**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match distance

- Use to match any routes that have the specified administrative distance.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match distance 25**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match extcommunity

- Use to match an extended community list in a route map.
- You can specify one or more extended community list names in a match clause. If you specify more than one extended community list, the lists are logical ORed.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match extcommunity topeka10**

- Use the **no** version to remove the match clause from a route map or a specified value from the match clause.

match ip address

- Use to match any route that has a destination network number that is permitted by an access list, a prefix list, or a prefix tree, or performs policy routing on packets.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match ip address prefix-tree boston**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match ip next-hop

- Use to match any routes that have a next-hop router address passed by the specified access list, prefix list, or prefix tree.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match ip next-hop 5 192.54.24.1**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match level

- Use to match routes for the specified type.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match level level-1**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match metric

- Use to match a route for the specified metric value.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match metric 10**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match metric-type

- Use to match a route for the specified metric type.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match metric-type external**

- Use the **no** version to delete the match clause from a route map.

match route-type

- Use to match a route for the specified route type.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**match route-type level-1**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

match tag

- Use to match the tag value of the destination routing protocol.
- Example

  host1(config)#**route-map 1**
  host1(config-route-map)#**match tag 25**

- Use the **no** version to delete the match clause from a route map or a specified value from the match clause.

neighbor route-map

- Use to apply a route map to incoming or outgoing routes.
- If you specify an outbound route map, BGP advertises only routes that match at least one section of the route map. For routes that do not match, no further processing takes place with respect to this peer, and those routes are not advertised to this peer. The nonmatching route is still in the BGP RIB and can be sent to other peers depending on the outbound policy applied to those peers.
- If you specify an inbound route map, BGP processes only the received routes that match at least one section of the route map. The nonmatching routes are rejected from entering the local BGP RIB and no further processing takes place.
- A clause with multiple values matches a route having any of the values; that is, the multiple values are logical ORed.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.
- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Example

  host1(config)#**rneighbor 192.168.5.34 route-map nyc1 in**

- Use the **no** version to remove the route map.

route-map

- Use to define the conditions for redistributing routes from one routing protocol into another, and for filtering or modifying updates sent to or received from peers.
- Each **route-map** command has a list of **match** and **set** commands associated with it.
- The **match** commands specify the match criteria—the conditions under which redistribution is allowed for the current route map.
- The **set** commands specify the set actions—the redistribution actions to perform if the criteria enforced by the **match** commands are set.

- Use route maps when you wish to have detailed control over how routes are redistributed between routing processes.
- The destination routing protocol is the one you specify with the **router** command.
- The source routing protocol is the one you specify with the **redistribute** command.
- A clause with multiple values matches a route having any of the values; that is, the multiple values are logical ORed.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- Example

  host1(config)#**route-map nyc1 permit 10**

- Use the **no** version to delete the route map.

set as-path prepend

- Use to modify an AS path for BGP routes by prepending one or more AS numbers or a list of AS numbers to the path list.
- The only global BGP metric available to influence the best-path selection is the AS-path length. By varying the length of the AS path, a BGP speaker can influence the best-path selection by a peer farther away.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set as-path prepend list list10**

- Use the **no** version to delete the set clause from a route map.

set comm-list delete

- Use to remove communities specified by the community list from the community attribute of routes matching the route map.
- You can use this command to delete communities only if the community list was created with a single community per list entry as shown in the following sample configuration for router Test:

  host1(config)#**ip community-list 1 permit 231:10**
  host1(config)#**ip community-list 1 permit 231:20**
  host1(config)#**router bgp 45**
  host1(config-router)#**neighbor 10.6.2.5 remote-as 5**
  host1(config-router)#**neighbor 10.6.2.5 route-map indelete in**
  host1(config-router)#**route-map indelete permit 10**
  host1(config-route-map)#**set comm-list 1 delete**

Router Test receives the same route from 10.6.2.5 and applies the indelete route map. BGP compares each list entry with the community attribute. A match is found for the list entry 231:10, and this community is deleted from the community attribute. Similarly, a match is found for the list entry of 231:20, and this community is deleted from the community attribute.

- Use the **no** version to delete the set clause from a route map.

set community

- Use to set the community attribute in BGP updates.
- You can specify a community list number in the range 1–4294967295, or in the new community format of AA:NN, or one of the following well-known communities:
  - **local-as**—Prevents advertisement outside the local AS
  - **no-advertise**—Prevents advertisement to any peer
  - **no-export**—Prevents advertisement beyond the BGP confederation boundary
- Alternatively, you can use the **list** keyword to specify the name of a community list that you previously created with the **ip community-list** command.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set community no-advertise**

- Use the **none** keyword to remove the community attribute from a route.
- Use the **no** version to delete the set clause from a route map.

set dampening

- Use to enable BGP route flap dampening only on routes that pass the match clauses of, and are redistributed by, a particular route map.
- BGP creates a dampening parameter block for each unique set of dampening parameters—such as suppress threshold and reuse threshold—used by BGP. For example, if you have a route map that sets the dampening parameters to one set of values for some routes and to another set of values for the remaining routes, BGP uses and stores two dampening parameter blocks, one for each set.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set dampening 5 1000 1500 45 15**

- Use the **no** version to delete the set clause from a route map.

set extcommunity

- Use to set the extended community attributes in a route map for BGP updates.
- You can specify a site-of-origin (**soo**) extended community and a route target (**rt**) extended community at the same time in a set clause without overwriting the other.

- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set extcommunity rt 10.10.10.2:325**

- Use the **no** version to delete the set clause from a route map.

set ip next-hop

- Use to set the next hop attribute of a route that matches a route map.
- This command is not supported for route maps used by the **table-map** command.
- You can specify an IP address or an interface as the next hop.
- Use the **peer-address** keyword to have the following effect:
  - On outbound route maps, disables the next hop calculation by setting the next hop to the IP address of the BGP speaker
  - On inbound route maps, overrides any third-party next-hop configuration by setting the next hop to the IP address of the peer
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set ip next-hop 192.56.32.1**

- Use the **no** version to delete the set clause from a route map.

set local-preference

- Use to specify a preference value for the AS path.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set local-preference 200**

- Use the **no** version to delete the set clause from a route map.

set metric

- Use to set the metric value—for BGP, the MED—for a route.
- To establish an absolute metric, do not enter a plus or minus sign before the metric value.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set metric 10**

- To establish a relative metric, specify a plus or minus sign immediately preceding the metric value. The value is added to or subtracted from the metric of any routes matching the route map. The relative metric value can be in the range 0–4294967295.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set metric -25**

- You cannot use both an absolute metric and a relative metric within the same route map sequence. Setting either metric overrides any previously configured value.
- Use the **no** version to delete the set clause from a route map.

set metric-type

- Use to set the metric type for a route.
- For BGP, affects all types of route maps. If the route map contains both a **set metric-type** and a **set metric** clause, the **set metric** clause takes precedence. Specifying the **internal** metric type in a BGP outbound route map, BGP sets the MED of the advertised routes to the IGP cost of the next hop of the advertised route. If the cost of the next hop changes, BGP is not forced to readvertise the route.
- For BGP, you can specify the following:
  - **external**—Reverts to the normal BGP rules for propagating the MED; this is the BGP default
  - **internal**—Sets the MED of a received route that is being propagated to an external peer equal to the IGP cost of the indirect next hop
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set metric-type internal**

- Use the **no** version to delete the set clause from a route map.

set origin

- Use to set the BGP origin of the advertised route.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set origin egp**

- Use the **no** version to delete the set clause from a route map.

set tag

- Use to set the tag value of the destination routing protocol.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set tag 23**

- Use the **no** version to delete the set clause from a route map.

set weight

- Use to specify the BGP weight for the routing table.
- The weights assigned with the **set weight** command in a route map override the weights assigned using the **neighbor weight** and **neighbor filter-list weight** commands.

- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set weight 200**

- Use the **no** version to delete the set clause from a route map.

Applying Table Maps

You can use the **table-map** command on a per-address-family basis to apply a route map to modify IP attributes of BGP routes that are about to be added to the IP routing table. In these route maps, you can use only the set clauses in Table 19.

Table 19: Set Clauses Supported in Route Maps Applied with the Table-Map Command

set distance	set metric-type
set level	set route-type
set metric	set tag

set distance

- Use to set the administrative distance attribute on routes being installed into the routing table that match the route map.
- Distance is used to establish preference between routes to the same prefix to identify the best route to that prefix. Setting distance in any other circumstance has no effect.
- Example

  host1(config-route-map)#**set distance 5**

- Use the **no** version to delete the set clause from a route map.

set level

- Use to specify where to import routes when all of a route map's match criteria are met.
- Example

  host1(config-route-map)#**set level level-2**

- Use the **no** version to delete the set clause from a route map.

set route-type

- Use to set the routes of the specified type.
- Example

  host1(config-route-map)#**set route-type internal**

- Use the **no** version to delete the set clause from a route map.

table-map

- Use to apply a policy to BGP routes about to be added to the IP routing table.
- The route map can include any of the clauses listed in Table 19.
- The new route map is applied to all routes that are subsequently placed in the IP routing table. To apply the new table map to routes that are already present in the IP routing table, you must refresh the IP routing table with the **clear ip routes *** command or the **clear ip bgp *** command (this command will bounce the BGP sessions).
- Example

```
host1(config-router)#table-map distmet1
host1(config-router)#exit
host1(config)#exit
host1#clear ip routes *
```

- Use the **no** version to halt application of the route map.

For example, suppose you want to change the distance and metric attributes to particular values for routes advertised by a members of a particular community. The **show ip route bgp** command indicates that the routes currently in the table have a variety of values for these attributes:

```
host1#show ip route bgp
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2

  Prefix/Length     Type      Next Hop        Dist/Met        Intf
------------------ ------- --------------- -------------- ------------
10.100.3.3/32        Bgp     10.12.12.1       20/0           ATM5/1.12
10.63.42.23/32      Bgp     10.45.2.31       12/5          ATM5/1.14
```

The following commands demonstrate how you can apply the policy to change these values:

```
host1(config)#route-map distmet1 permit 5
host1(config-route-map)#match community boston42
host1(config-route-map)#set distance 33
host1(config-route-map)#set metric 44
host1(config-route-map)#exit
host1(config)#router bgp 100
host1(config-router)#table-map distmet1
host1(config-router)#exit
host1(config)#exit
host1#clear ip routes *
```

The **show ip route bgp** command reveals the new values:

```
host1#show ip route bgp
Protocol/Route type codes:
  I1- ISIS level 1, I2- ISIS level2,
  I- route type intra, IA- route type inter, E- route type external,
  i- metric type internal, e- metric type external,
  O- OSPF, E1- external type 1, E2- external type2,
  N1- NSSA external type1, N2- NSSA external type2

  Prefix/Length     Type      Next Hop        Dist/Met        Intf
------------------ ------- --------------- -------------- ------------
10.100.3.3/32        Bgp     10.12.12.1       33/44           ATM5/1.12
10.63.42.23/32      Bgp     10.45.2.31      33/44          ATM5/1.14
```

Access Lists

An access list is a sequential collection of permit and deny conditions that you can use to filter inbound or outbound routes. You can use different kinds of access lists to filter routes based on either the prefix or the AS path.

Filtering Prefixes

To filter routes based on the prefix, you can do any of the following:

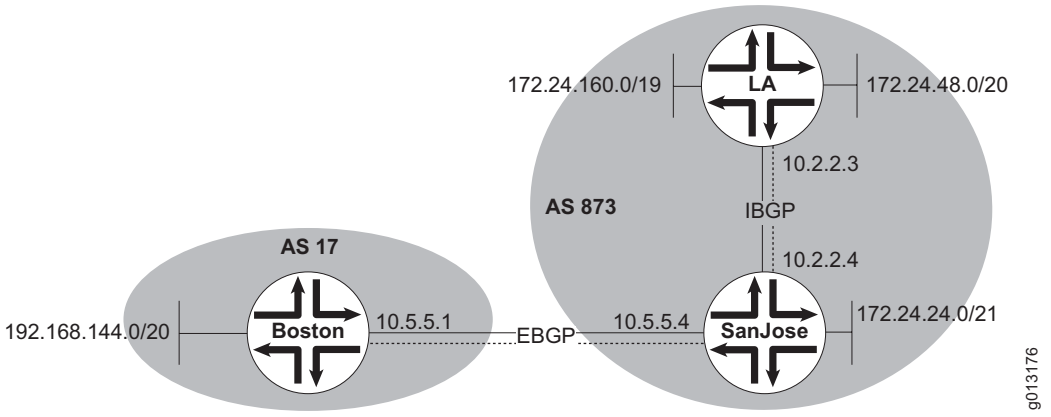
- Define an access list with the **access list** command and apply the list to routes received from or passed to a neighbor with the **neighbor distribute-list** command.
- Define a prefix list with the **ip prefix-list** command and apply the list to routes received from or passed to a neighbor with the **neighbor prefix-list** command.
- Define a prefix tree with the **ip prefix-tree** command and apply the list to routes received from or passed to a neighbor with the **neighbor prefix-tree** command.

The router compares each route's prefix against the conditions in the list or tree one by one. If the first match is for a permit condition, the route is accepted or passed. If the first match is for a deny condition, the route is rejected or blocked. The order of conditions is critical because testing stops with the first match. If no conditions match, the router rejects or blocks the address; that is, the last action of any list is an implicit deny condition for all routes. The implicit rule is displayed by **show access-list** and **show configuration** commands.

You cannot selectively place conditions in or remove conditions from an access list, prefix, list, or prefix tree. You can insert a new condition only at the end of a list or tree.

Consider the network structure in Figure 21.

Figure 21: Filtering with Access Lists

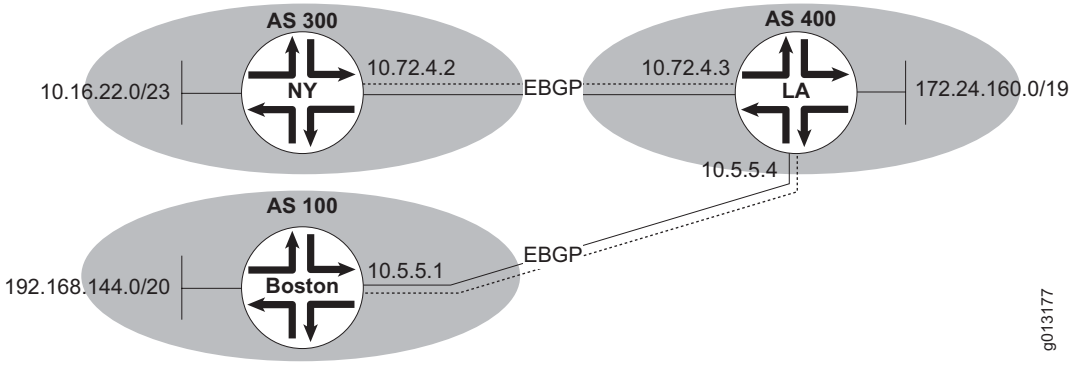


The following commands configure router Boston to apply access list reject1 to routes inbound from router SanJose. Access list reject1 rejects routes matching 172.24.160.0/19.

```
host3(config)#router bgp 17
host3(config-router)#neighbor 10.5.5.4 remote-as 873
host3(config-router)#neighbor 10.5.5.4 distribute-list reject1 in
host3(config-router)#exit
host3(config)#access-list reject1 permit 172.24.48.0 0.0.255
host3(config)#access-list reject1 deny 172.24.160.0 0.0.255
host3(config)#access-list reject1 permit 172.24.24.0 0.0.255
```

Consider the network shown in Figure 22. Router NY originates network 10.16.22.0/23 and advertises it to router LA. Suppose you do not want router LA to advertise that network to router Boston. You can apply an access list to updates from router LA to router Boston that prevents router LA from propagating updates for network 10.16.22.0/23.

Figure 22: Filtering Routes with an Access List



The following commands configure router LA:

```
host2(config)#router bgp 400
host2(config-router)#network 172.24.160.0 mask 255.255.224.0
host2(config-router)#neighbor 10.72.4.2 remote-as 300
host2(config-router)#neighbor 10.5.5.1 remote-as 100
host2(config-router)#neighbor 10.5.5.1 distribute-list 1 out
host2(config-router)#exit
host2(config)#access-list 1 deny 10.16.22.0 0.254.255.255
```

access-list

- Use to define an IP access list to permit or deny routes based on the prefix.
- Each access list is a set of permit or deny conditions for routes based on matching a route's prefix.
- Use the **neighbor distribute-list** command to apply the access list to routes received from or forwarded to a neighbor.
- Use the **log** keyword to log an Info event in the ipAccessList log whenever an access-list rule is matched.
- Use the **no** version to delete an IP access list or the specified entry in the access list.

clear access-list

- Use to clear IP access list counters.
- Each access list has a counter for its entries.
- Example

  ```
  host1#clear access-list reject1
  ```

- There is no **no** version.

neighbor distribute-list

- Use to filter routes to selected prefixes as specified in an access list.
- Using distribute lists is one of three ways to filter BGP advertisements. The other ways are as follows:
  - Use AS-path filters with the **ip as-path access-list** and the **neighbor filter-list** commands.
  - If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.
  - Use filters with route maps with the **route-map** and the **neighbor route-map** commands.

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to disassociate the access list from a neighbor.

neighbor prefix-list

- Use to assign an inbound or outbound prefix list.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.
- Example

  host1(config-router)#**neighbor 192.168.1.158 prefix-list seoul19 in**
- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to remove the prefix list.

neighbor prefix-tree

- Use to assign an inbound or outbound prefix tree.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.

- Example

  host1(config-router)#**neighbor 192.168.1.158 prefix-tree newyork out**

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- IPv6 prefix trees are not supported, Therefore you can specify an IPv6 address with this command only within the IPv4 address family and when you want to advertise IPv4 routes to IPv6 peers.
- Use the **no** version to remove the prefix tree.

Filtering AS Paths with a Filter List

You can use a filter list to filter incoming and outgoing routes based on the value of the AS-path attribute. Whenever a BGP route passes through an AS, BGP prepends its AS number to the AS-path attribute. The AS-path attribute is the list of ASs that a route has passed through to reach a destination.

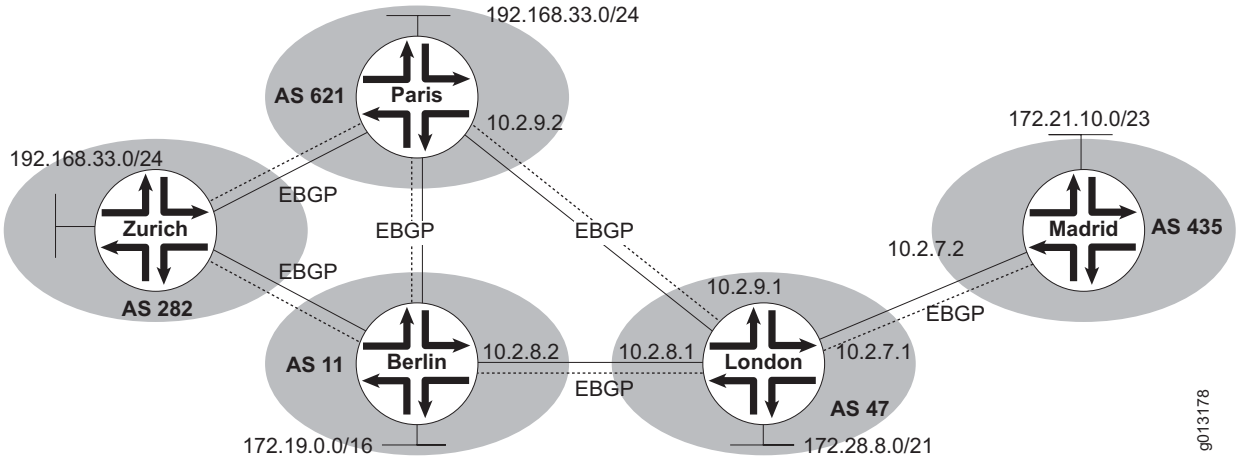
To filter routes based on the AS-path, define the access list with the **ip as-path access-list** command, and apply the list to routes received from or passed to a neighbor with the **neighbor filter-list** command. AS-path access lists use regular expressions to describe the AS path to be matched. A regular expression uses special characters—often referred to as metacharacters—to define a pattern that is compared with an input string. For a full discussion of regular expressions, with examples on how to use them, see *Using Regular Expressions* in *JUNOSe IP Services Configuration Guide, Chapter 1, Configuring Routing Policy*.

The router compares each route's AS path against the conditions in the access list one by one. If the first match is for a permit condition, the route is accepted or passed. If the first match is for a deny condition, the route is rejected or blocked. The order of conditions is critical because testing stops with the first match. If no conditions match, the router rejects or blocks the route; that is, the last action of any list is an implicit deny condition for all routes.

You cannot selectively place conditions in or remove conditions from an AS-path access list. You can insert a new condition only at the end of an AS-path access list.

Example 1 Consider the network structure in Figure 23.

Figure 23: Filtering with AS-Path Access Lists



Suppose you want router London to behave in the following way:

- Accept routes originated in AS 621 only if they pass directly to router London
- Accept routes originated in AS 11 only if they pass directly to router London
- Forward routes from AS 282 to AS 435 only if they pass through either AS 621 or AS 11, but not both AS 621 and AS 11

The following commands configure router London to apply filters based on the AS path to routes received from router Berlin and router Paris and to routes forwarded to router Madrid.

```
host1(config)#router bgp 47
host1(config-router)#neighbor 10.2.9.2 remote-as 621
host1(config-router)#neighbor 10.2.9.2 filter-list 1 in
host1(config-router)#neighbor 10.2.8.2 remote-as 11
host1(config-router)#neighbor 10.2.8.2 filter-list 2 in
host1(config-router)#neighbor 10.2.7.2 remote-as 435
host1(config-router)#neighbor 10.2.7.2 filter-list 3 out
host1(config-router)#exit
host1(config)#ip as-path access-list 1 deny ^621_11$
host1(config)#ip as-path access-list 1 permit .*
host1(config)#ip as-path access-list 2 deny ^11_621$
host1(config)#ip as-path access-list 2 permit .*
host1(config)#ip as-path access-list 3 deny ^11_621_282
host1(config)#ip as-path access-list 3 deny ^621_11_282
host1(config)#ip as-path access-list 3 permit .*
```

AS-path access list 1 is applied to routes that router London receives from router Paris. Router London rejects routes with the AS path (621 11).

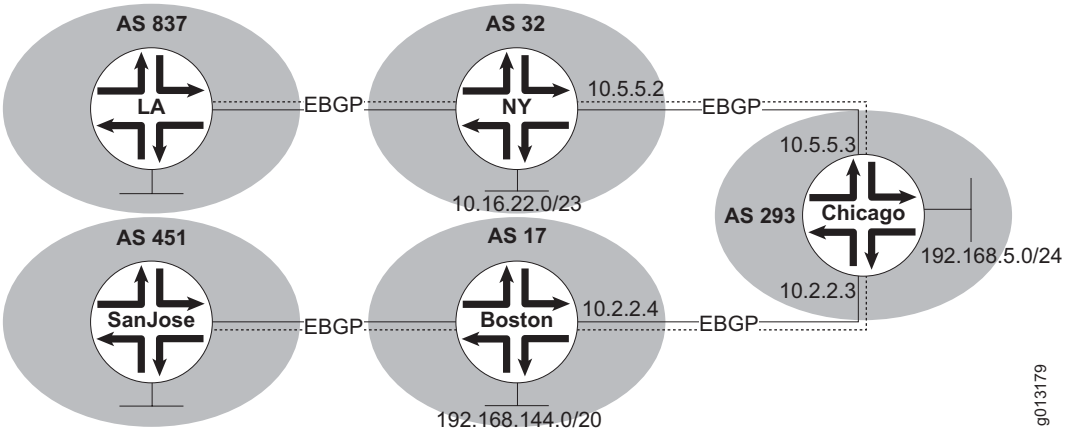
AS-path access list 2 is applied to routes that router London receives from router Berlin. Router London rejects routes with the AS path (11 621) or (621 282 11).

Router London accepts routes with the AS path (11 282), (621 282), (621 11 282), or (11 621 282). However, it applies AS-path access list 3 to routes it forwards to router Madrid, and filters out routes with the AS path (621 11 282) or (11 621 282).

Example 2 Consider the following commands used to configure router Chicago in Figure 24:

```
host1(config)#router bgp 293
host1(config-router)#neighbor 10.5.5.2 remote-as 32
host1(config-router)#neighbor 10.5.5.2 filter-list 1 in
host1(config-router)#neighbor 10.2.2.4 remote-as 17
host1(config-router)#exit
host1(config)#ip as-path access-list 1 deny ^32$
```

Figure 24: Assigning a Filter List



Access list 1 denies routes that originate in AS 32—and therefore routes originated by router NY—because the AS-path attribute for these routes begins with (and indeed consists only of) the value 32.

Routes originating anywhere else—such as in AS 837, AS 17, or AS 451—are permitted, because their AS-path attributes do not begin with 32.

ip as-path access-list

- Use to define an AS-path access list to permit or deny routes based on the AS path.
- Each access list is a set of permit or deny conditions for routes based on matching a route's AS path with a regular expression. If the regular expression matches the representation of the AS path of the route as an ASCII string, then the permit or deny condition applies. The AS path does not contain the local AS number.
- Use the **neighbor filter-list** command to apply the AS-path access list. You can apply access list filters to inbound and outbound BGP routes. You can assign weights to routes matching the AS-path access list.
- Use the **no** version to remove a single access list entry if **permit** or **deny** and a *path-expression* are specified. Otherwise, the entire access list is removed.

neighbor filter-list

- Use to assign an AS-path access list to matching inbound or outbound routes with the **in** or **out** keywords.
- You can specify an optional weight value with the **weight** keyword to assign a relative importance to incoming routes matching the AS-path access list.
- The name of the access list is a string of up to 32 characters.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.
- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

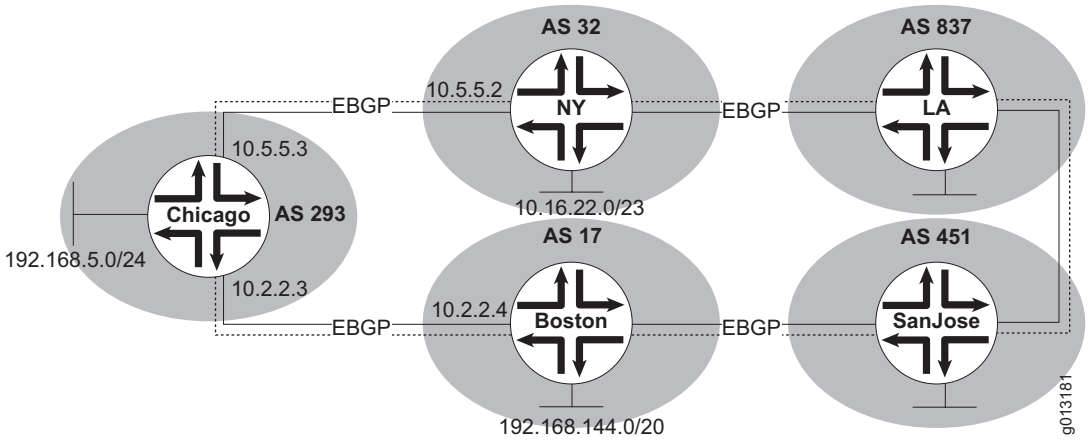
- Use the **no** version to disassociate the access list from a neighbor.

Filtering AS Paths with a Route Map

You can use a route map instead of the **neighbor filter-list** command to apply access lists for filtering routes. In Figure 25, suppose router Chicago is configured as follows:

```
host1(config)#router bgp 293
host1(config-router)#network 192.168.5.0 mask 255.255.255.0
host1(config-router)#neighbor 10.2.2.4 remote-as 17
host1(config-router)#neighbor 10.2.2.4 weight 150
host1(config-router)#neighbor 10.5.5.2 remote-as 32
host1(config-router)#neighbor 10.5.5.2 weight 50
```

Figure 25: Route Map Filtering



Routes learned from router Boston have a weight of 150, whereas those learned from router NY have a weight of 50. Router Chicago therefore prefers all routes learned from router Boston to those learned from router NY. Based on this configuration, router Chicago prefers routes to prefixes originating in AS 837 or originating in AS 32 that pass through router Boston over routes to those same prefixes that pass through router NY.

This is a longer path than you might desire. You can avoid this result by configuring a route map to modify the weight of certain routes learned by router Chicago:

```
host1(config-router)#neighbor 10.5.5.2 route-map alpha in
host1(config-router)#exit
host1(config)#route-map alpha permit 10
host1(config-route-map)#match as-path dog1
host1(config-route-map)#set weight 175
host1(config-route-map)#exit
host1(config)#ip as-path access-list dog1 permit _32$
host1(config)#ip as-path access-list dog1 permit _837$
host1(config)#route-map alpha permit 20
host1(config-route-map)#match as-path dog2
host1(config-route-map)#exit
host1(config)#ip as-path access-list dog2 permit .*
```

BGP applies route map alpha to all routes learned from 10.5.5.2 (router NY). Instance 10 of route map alpha matches routes with access list dog1. This access list permits any route whose AS-path attribute ends in 32 or 837—that is, routes that originate in AS 32 or AS 837. It sets their weight to 175, overriding the neighbor weight (50) set for updates received from 10.5.5.2. Then, instance 20 of route map alpha permits all other routes with no modification.

The result of this improved configuration is the following:

- Router Chicago prefers routes learned from router Boston (weight 150) over routes learned from router NY (weight 50), except that
- Router Chicago prefers routes learned from router NY that originate in AS 837 or AS 32 (weight 175 as a result of route map alpha) over the same routes learned from router Boston (weight 150).

Refer to the commands and guidelines in the section *Types of BGP Route Maps* on page 69 for more information about configuring route maps.

Configuring the Community Attribute

A community is a logical group of prefixes that share some common attribute. Community members can be on different networks and in different autonomous systems. BGP allows you to define the community to which a prefix belongs. A prefix can belong to more than one community. The community attribute lists the communities to which a prefix belongs.

You can use communities to simplify routing policies by configuring which routing information a BGP speaker will accept, prefer, or distribute to other neighbors according to community membership. When a route is learned, advertised, or redistributed, a BGP speaker can set, append, or modify the community of a route. When routes are aggregated, the resulting BGP update contains a community attribute that contains all communities from all of the aggregated routes (if the aggregate is an AS-set aggregate).

Several well-known communities have been predefined. Table 20 describes how a BGP speaker handles a route based on the setting of its community attribute.

Table 20: Action Based on Well-Known Community Membership

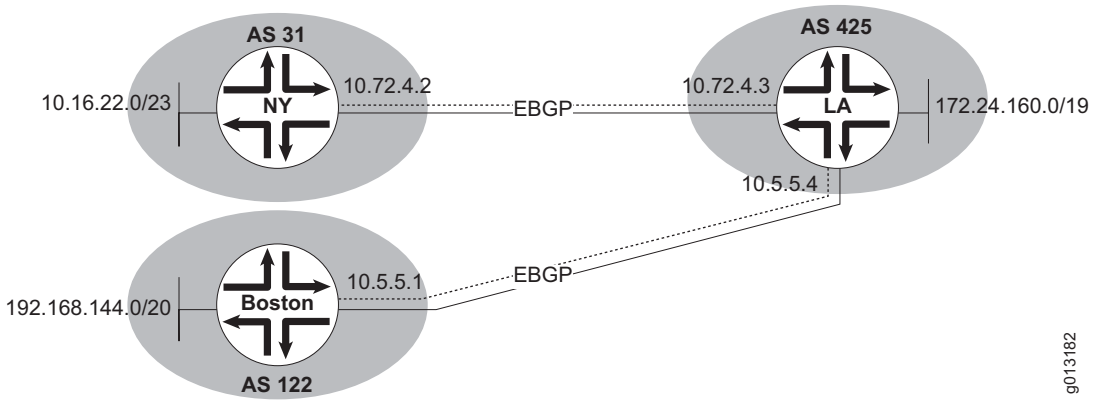
Well-Known Community	BGP Speaker Action
no-export	Does not advertise the route to any EBGP peers (does not advertise the route beyond the local AS)
no-advertise	Does not advertise the route to any peers, IBGP or EBGP
local-as (also known as no-export-subconfed)	Advertises the route only to peers within the local confederation
internet	Advertises this route to the Internet community; by default, all prefixes are members of the Internet community

In addition to the well-known communities, you can define local-use communities, also known as private communities or general communities. These communities serve as a convenient way to categorize groups of routes to facilitate the use of routing policies. The community attribute consists of four octets, but it is common practice to designate communities in the *AA*:*NN* format. The autonomous system number (*AA*) comprises the higher two octets, and the community number (*NN*) comprises the lower two octets. Both are expressed as decimal numbers. For example, if a prefix in AS 23 belongs to community 411, the attribute can be expressed as 23:411. Use the **ip bgp-community new-format** command to specify that the **show** commands display communities in this format.

Use the **set community** command in route maps to configure the community attributes. By default, the community attribute is not sent to BGP peers. To send the community attribute to a neighbor, use the **neighbor send-community** command.

Consider the network structure shown in Figure 26. The following sample configurations illustrate some of the capabilities of using the community attribute.

Figure 26: Communities



The following commands configure router NY to apply route map *setcomm* to routes going out to 10.72.4.3. If the community attribute of such a route matches instance 10 of the route map, router NY sets the community attribute to 31:15. All locally originated routes will match this instance of the route map.

```
host1(config)#router bgp 31
host1(config-router)#network 10.16.22.0 mask 255.255.254.0
host1(config-router)#neighbor 10.72.4.3 remote-as 425
host1(config-router)#neighbor 10.72.4.3 send-community
host1(config-router)#neighbor 10.72.4.3 route-map setcomm out
host1(config-router)#exit
host1(config)#ip as-path access-list 1 permit ^$
host1(config)#route-map setcomm permit 10
host1(config-route-map)#match as-path 1
host1(config-route-map)#set community 31:15
```

The following commands configure router LA to apply route map *matchcomm* to routes coming in from 10.72.4.2. If the community attribute of such a route matches instance 10 of the route map, router LA sets the weight of the route to 25.

```
host2(config)#router bgp 425
host2(config-router)#network 172.24.160 mask 255.255.224.0
host2(config-router)#neighbor 10.72.4.2 remote-as 31
host2(config-router)#neighbor 10.72.4.2 send-community
host2(config-router)#neighbor 10.72.4.2 route-map matchcomm in
host2(config-router)#neighbor 10.5.5.1 remote-as 122
host2(config-router)#neighbor 10.5.5.1 send-community
host2(config-router)#exit
host2(config)#ip community-list 1 permit 31:15
host2(config)#route-map matchcomm permit 10
host2(config-route-map)#match community 1
host2(config-route-map)#set weight 25
```

The following commands configure router Boston to apply route map 5 to routes going out to 10.5.5.4. If the destination IP address of such a route matches instance 10 of the route map, router Boston sets the community attribute of the route to no-export.

```
host3(config)#router bgp 122
host3(config-router)#network 192.168.144.0 mask 255.255.240.0
host3(config-router)#neighbor 10.5.5.4 remote-as 425
host3(config-router)#neighbor 10.5.5.4 send-community
host3(config-router)#neighbor 10.5.5.4 route-map 5 out
host3(config-router)#exit
host3(config)#route-map 5 permit 10
host3(config-route-map)#match ip address access5
host3(config-route-map)#set community no-export
host3(config-route-map)#exit
host3(config)#access-list access5 permit 10.16.22.112
```

Suppose router Boston forwards a route destined for 10.16.22.112 through router LA. Route map 5 matches and sets the community attribute to no-export. As a consequence router LA does not export the route to router NY; the route does not reach its destination.

ip bgp-community new-format

- Use to specify that communities must be displayed in *AA*:*NN* format, where *AA* is a number that identifies the autonomous system and *NN* is a number that identifies the community within the autonomous system.
- Use the **no** version to restore the default display.

neighbor send-community

- Use to specify that a community attribute must be sent to a BGP neighbor.
- You can specify that only standard communities, only extended communities, or both be sent.
- When you create a neighbor in a VPNv4 address family, that neighbor automatically gets a **neighbor send-community both** command; this command subsequently appears in a **show configuration** display because it is not the default.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override this inheritance for a peer group member.
- Example

  ```
  host1(config-router)#neighbor send-community westcoast extended
  ```

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to send only standard communities to a BGP neighbor. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

set community

- Use to set the community attribute in BGP updates.
- You can specify a community list number in the range 1–4294967295, or in the new community format of *AA*:*NN*, or one of the following well-known communities:
  - **local-as**—Prevents advertisement outside the local AS
  - **no-advertise**—Prevents advertisement to any peer
  - **no-export**—Prevents advertisement beyond the BGP confederation boundary
- Alternatively, you can use the **list** keyword to specify the name of a community list that you previously created with the **ip community-list** command.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set community no-advertise**

- Use the **none** keyword to remove the community attribute from a route.
- Use the **no** version to delete the set clause from a route map.

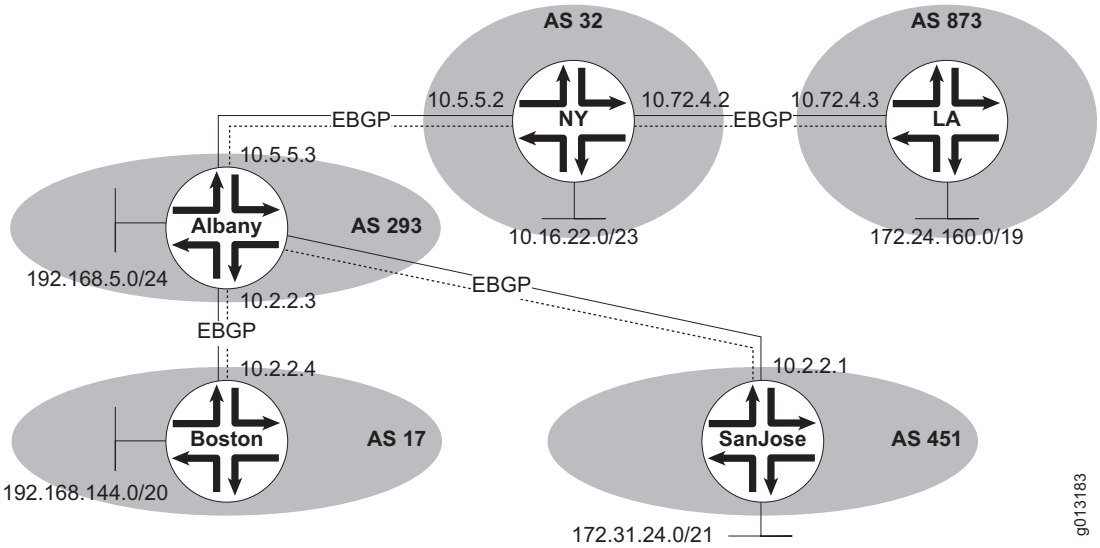
Community Lists

A community list is a sequential collection of permit and deny conditions. Each condition describes the community number to be matched. If you issued the **ip bgp-community new-format** command, the community number is in *AA*:*NN* format; otherwise it is in decimal format.

The router tests the community attribute of a route against the conditions in a community list one by one. The first match determines whether the router accepts (the route is permitted) or rejects (the route is denied) a route having the specified community. Because the router stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the router rejects the route.

Consider the network structure shown in Figure 27.

Figure 27: Community Lists



Suppose you want router Albany to set metrics for routes that it forwards to router Boston based on the communities to which the routes belong. You can create community lists and filter the routes with a route map that matches on the community list. The following commands demonstrate how to configure router Albany:

```
host1(config)#router bgp 293
host1(config-router)#neighbor 10.5.5.2 remote-as 32
host1(config-router)#neighbor 10.2.2.1 remote-as 451
host1(config-router)#neighbor 10.2.2.4 remote-as 17
host1(config-router)#neighbor 10.2.2.4 route-map commtrc out
host1(config-router)#exit
host1(config)#route-map commtrc permit 1
host1(config-route-map)#match community 1
host1(config-route-map)#set metric 20
host1(config-route-map)#exit
host1(config)#route-map commtrc permit 2
host1(config-route-map)#match community 2
host1(config-route-map)#set metric 75
host1(config-route-map)#exit
host1(config)#route-map commtrc permit 3
host1(config-route-map)#match community 3
host1(config-route-map)#set metric 85
host1(config-route-map)#exit
host1(config)#ip community-list 1 permit 25
host1(config)#ip community-list 2 permit 62
host1(config)#ip community-list 3 permit internet
```

Community list 1 comprises routes with a community of 25; their metric is set to 20. Community list 2 comprises routes with a community of 62; their metric is set to 75. Community 3 catches all remaining routes by matching the internet community; their metric is set to 85.

ip community-list

- Use to create a standard or a regular expression community list for BGP and controls access to it.
- A route can belong to any number of communities, so a community list can have many entries comprising many communities.
- You can specify one or more community values when you create a community list. A clause in a route map that includes a list having more than one value only matches a route having all of the values; that is, the multiple values are logical ANDed.
- Example

  host1(config)#**ip community-list 1 permit 100:2 100:3 100:4**
  host1(config)#**route-map marengo permit 10**
  host1(config-route-map)#**match community 1**

  A route matches this community list only if it belongs to at least all three communities in community list 1: Communities 100:2, 100:3, and 100:4.

- Use the **no** version to remove a single community list entry if **permit** or **deny** and a *path-expression* are specified. Otherwise, the entire community list is removed.

The router supports the new BGP extended community attribute. This attribute enables the definition of a new type of IP extended community and extended community list unrelated to the community list that uses regular expressions. BGP speakers can use the new extended community attribute to control routes similarly to the way it uses the community attribute. The extended community attribute is currently defined in the Internet draft, BGP Extended Communities Attribute—draft-ietf-idr-bgp-ext-communities-07.txt (February 2004 expiration).

NOTE: IETF drafts are valid for only 6 months from the date of issuance. They must be considered as works in progress. Please refer to the IETF Web site at http://www.ietf.org for the latest drafts.

ip extcommunity-list

- Use to create an extended community list for BGP and control access to it.
- A route can belong to any number of communities, so an extended community list can have many entries comprising many communities.
- You can specify one or more community values when you create an extended community list. A clause in a route map that includes a list having more than one value only matches a route having all of the values; that is, the multiple values are logical ANDed.

- Example

  host1(config)#**ip extcommunity-list boston1 permit 100:2 100:3 100:4**
  host1(config)#**route-map marengo permit 10**
  host1(config-route-map)#**match extcommunity boston1**

  A route matches this community list only if it belongs to at least all three communities in extended community list boston1: Communities 100:2, 100:3, and 100:4.

- Use the **no** version to remove a single extended community list entry if **permit** or **deny** and a *path-expression* are specified. Otherwise, the entire community list is removed.

Resetting a BGP Connection

When a routing policy changes, use the **clear ip bgp** and **clear bgp ipv6** commands to clear the current BGP session and implement the new policy.

Clearing a BGP session can create a major disruption in the network operation; this is known as a hard clear. For this reason, you can use the **soft in** and **soft out** options of the command (a soft clear) to activate policies without disrupting the BGP session.

The **soft in** option reapplies inbound policy to received routes; the **soft out** option resends routes to a neighbor after reapplying outbound policy. The **soft in prefix-list** option causes BGP to push any prefix list outbound route filters (ORFs) to the peer and then reapply inbound policy to received routes.

NOTE: Resetting the BGP connection is slightly different when you change outbound policies for peer groups for which you have enabled Adj-RIBs-Out. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group.

clear bgp ipv6
clear ip bgp

- Use to clear the current BGP connection or to activate a new policy without clearing the BGP session.
- You can specify the IP address of a BGP neighbor, the name of a BGP peer group, or an address family to be cleared.
- Use the asterisk (*) to clear all BGP connections.
- If you do not use the **soft in** or **soft out** options, the clear is known as a hard clear and clears the current BGP connection.
- Use the **soft in** option to reapply inbound policy to all received routes without clearing the BGP session.
- Use the **soft in prefix-filter** option to push an ORF to the peer and reapply inbound policy to all received routes without clearing the BGP session.

- Use the **soft out** option to reapply outbound policy and resend routes without clearing the BGP session.
- This command takes effect immediately.
- There is no **no** version.

Changing Policies Without Disruption

Changing policies can cause major network disruptions when you bring down sessions to reapply the modified policies. You can use either of the methods in this section to minimize network disruptions.

Soft Reconfiguration

You can use *soft reconfiguration* to enable the nondisruptive reapplication of inbound policies. Issuing the command causes the router to store copies of the routes received from the specified peer or from all members of the specified peer group. The route copies are stored unmodified, before application of inbound policies.

If you then change your inbound policies, you can apply them to the stored routes without clearing your BGP sessions—and causing network disruptions—by issuing the **clear ip bgp soft in** command.

neighbor soft-reconfiguration inbound

- Use to initiate the storage of copies of routes received from the specified IP address or from all members of the specified peer group.
- Use with the **clear ip bgp soft in** command to reapply inbound policies to stored routes without clearing the BGP sessions.
- Example

  host1(config)#**router bgp 37**
  host1(config-router)#**neighbor 192.168.1.1 remote-as 42**
  host1(config-router)#**neighbor 192.168.1.1 soft-reconfiguration inbound**

- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- If the route-refresh capability was negotiated with the peer, BGP immediately sends a route-refresh message to the peer to populate the Adj-RIBs-In table.

  If the route-refresh capability was not negotiated with the peer, BGP automatically bounces the session. The Adj-RIBs-In table is repopulated when routes are received from the peer after the session comes back up.

- Use the **no** version to disable storage of the route copies. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Route-Refresh Capability

The route-refresh capability provides a lower-cost alternative to soft reconfiguration as a means to change policies without major disruptions. The router advertises the route-refresh capability when it establishes a BGP session with a peer to indicate that it is capable of exchanging BGP route-refresh messages. If inbound soft reconfiguration is disabled (the default) and you issue the **clear ip bgp soft in** command, the router sends route-refresh messages to its peers that have advertised this capability. The messages contain a request for the peer to resend its routes to the router. The new inbound policy is then applied to the routes as they are received.

Our implementation conforms to RFC 2918—Route Refresh Capability for BGP-4 (September 2000), but it also supports nonstandard implementations.

Cooperative Route Filtering

If a BGP speaker negotiates the cooperative route filtering capability with a peer, then the speaker can transfer inbound route filters to the peer. The peer then installs the filter as an outbound route filter (ORF) on the remote end. The ORF is applied by the peer after application of its configured outbound policies. This cooperative filtering has the advantage of both reducing the amount of processing required for inbound BGP updates and reducing the amount of BGP control traffic generated by BGP updates.

clear ip bgp soft prefix-filter

- Use to push an ORF to the peer and reapply inbound policy to all received routes without clearing the BGP session.
- You can specify the IP address of a BGP neighbor, the name of a BGP peer group, or an address family to be cleared.
- Use the asterisk (*) to clear all BGP connections.
- If the ORF capability is not configured or received on the peer, then the **prefix-filter** keyword is ignored and the router performs a normal inbound soft reconfiguration.
- This command takes effect immediately.
- There is no **no** version.

neighbor capability

- Use to negotiate the exchange of inbound route filters and their installation as ORFs by specifying the **orf** keyword, an ORF type, and the direction of the capability.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- You cannot configure the **receive** direction for the **orf** capability for a peer that is a member of a peer group or for a peer.
- When issued with the **orf** keyword, this command takes effect immediately and automatically bounces the BGP session.

- Example

  host1(config-router)#**neighbor 192.168.1.158 capability orf prefix-list both**

- Use the **no** version to prevent advertisement of the capability. Use the **default** version to restore the default, advertising the capability.

neighbor maximum-orf-entries

- Use to set the maximum number of ORF entries of any one type that will be accepted from the specified neighbor.
- Example

  host1(config-router)#**neighbor 192.168.1.158 maximum-orf-entries 125000**

- Use the **no** version to restore the default value of no limits.

neighbor prefix-list

- Use to assign an inbound or outbound prefix list.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.
- Example

  host1(config-router)#**neighbor 192.168.1.158 prefix-list seoul19 in**

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to remove the prefix list.

Configuring Route Flap Dampening

Route flap dampening is a mechanism for minimizing instability caused by route flapping. *Route flapping* occurs when a link is having a problem and is constantly going up and down. Every time the link goes down, the upstream peer withdraws the routes from all its neighbors. When the link comes back up again, the peer advertises those routes globally. When the link problem appears again, the peer withdraws the routes again. This process continues until the underlying problem is fixed.

The router stores a penalty value with each route. Each time the route flaps, the router increases the penalty by 1000. If the penalty for a route reaches a configured *suppress* value, the router suppresses the route. That is, the router does not include the route as a forwarding entry and does not advertise the route to BGP peers.

The penalty decrements by 50 percent for each *half-life* interval that passes. The half-life interval resets when the route flaps and the penalty increments. The route remains suppressed until the penalty falls below the configured *reuse* threshold, at which point the router once again advertises the route. You can specify a *max-suppress-time* for route suppression; after this interval passes, the router once again advertises the route.

BGP creates a *dampening parameter block* for each unique set of dampening parameters—such as suppress threshold and reuse threshold—used by BGP. For example, if you have a route map that sets the dampening parameters to one set of values for some routes and to another set of values for the remaining routes, BGP uses and stores two dampening parameter blocks, one for each set.

Global Route Flap Dampening

Use the **bgp dampening** command if you want to enable route flap dampening with the same values on all BGP routes, or on all routes matching the specified route map. If you specify a route map, the router dampens only routes that are permitted by the route map. For example:

```
host1(config-router)#bgp dampening 8 600 2500 30 route-map 1
```

bgp dampening

- Use to enable BGP route flap dampening on all BGP routes or routes matching a specified route map.
- You can specify a complete set of values that determine how routes are dampened. If you choose to do so, you must specify the entire set:
  - *half-life—W*hen this period expires, the penalty assigned to a route is decreased by half
  - *reuse—W*hen the penalty for a flapping route falls below this limit, the route is unsuppressed (added back to the BGP table and used for forwarding)
  - *suppress*—When a route's penalty exceeds this limit, the route is suppressed
  - *max-suppress-time*—When the period a route has been suppressed exceeds this limit, the route becomes unsuppressed
- If you specify the preceding set of dampening values, you can optionally specify a *half-life-unreachable* period to apply to unreachable routes. If you do not specify this value, the same half-life period is used for both reachable and unreachable routes.
- Dampening applies only to routes learned by means of EBGP.

- The new dampening parameters are applied in future flaps. Changing the dampening parameters does not affect the Figure of Merit that has been calculated for routes using the old dampening parameters. To reset the Figure-of-Merit for all routes, you must issue the **clear ip bgp dampening** command.
- Use the **no** version to disable route flap dampening.

clear bgp ipv6 dampening
clear ip bgp dampening

- Use to clear the BGP route flap dampening information and unsuppress the suppressed routes.
- You can use the **flap-statistics** keyword as an alternative to the **dampening** keyword. Both achieve the same results.
- This command takes effect immediately.
- Examples

  To clear IPv6 dampening information for all routes in all routers:

  host1#**clear bgp ipv6 dampening**

  To clear IPv6 dampening information for a specific route:

  host1#**clear bgp ipv6 dampening 6000::/64**

  To clear IPv4 dampening information for all routes in all address families in all VRFs:

  host1#**clear ip bgp dampening**

  To clear IPv4 dampening information for all routes in VRF dogwood:

  host1#**clear ip bgp ipv4 dogwood dampening**

  To clear IPv4 dampening information for all non-VRF routes in the IPv4 unicast address family:

  host1#**clear ip bgp vrf unicast dampening**

  To clear IPv4 dampening information for a specific route:

  host1#**clear ip bgp dampening 192.168.5.0 255.255.255.0**

  To clear IPv4 dampening information for the most specific route matching an address:

  host1#**clear ip bgp dampening 192.168.5.0**

- There is no **no** version.

Policy-Based Route Flap Dampening

You can use policy-based route flap dampening to apply different dampening criteria to different routes. Establish one or more match clauses for an instance of a route map. Then use the **set dampening** command to specify the dampening values that apply to routes that pass all the match clauses for that route map. Consider the following example:

```
host1(config)#route-map 21 permit 5
host1(config-route-map)#match as-path 1
host1(config-route-map)#set dampening 5 1000 1500 45 15
host1(config-route-map)#exit
host1(config)#ip as-path access-list 1 permit ^300_
```

Access list 1 permits routes that originate in AS 300. Instance 5 of route map 21 permits routes that match access list 1 and applies the set of dampening criteria to only those routes; in this case, routes that originate in AS 300.

You can restore the advertisement of routes suppressed as a result of policy-based route flap dampening by issuing the **neighbor unsuppress-map** command. You can unsuppress routes from a specified neighbor or peer group. You must specify a route map; only those routes that match the route map are unsuppressed.

neighbor unsuppress-map

- Use to unsuppress routes that have been suppressed by a **set dampening** clause in a route map.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override this inheritance for a peer group member.
- Routes previously suppressed by a route map that are unsuppressed by this command are not automatically advertised; you must use the **clear ip bgp** command to perform a hard clear or outbound soft clear.
- Example

  ```
  host1(config-router)#neighbor berlin5 unsuppress-map inmap3
  ```

- Use the **no** version to restore the default values.

set dampening

- Use to enable BGP route flap dampening only on routes that pass the match clauses of, and are redistributed by, a particular route map.
- BGP creates a dampening parameter block for each unique set of dampening parameters—such as suppress threshold and reuse threshold—used by BGP. For example, if you have a route map that sets the dampening parameters to one set of values for some routes and to another set of values for the remaining routes, BGP uses and stores two dampening parameter blocks, one for each set.
- Example

  ```
  host1(config)#route-map nyc1 permit 10
  host1(config-route-map)#set dampening 5 1000 1500 45 15
  ```

- Use the **no** version to delete the set clause from a route map.

Policy Testing

You can analyze and check your BGP routing policies on your network before you implement the policies. Use the **test ip bgp neighbor** and **test bgp ipv6 neighbor** commands to test the outcome of a BGP policy. The commands output displays the routes that are advertised or accepted if the specified policy is implemented.

NOTE: You can use the standard redirect operators to redirect the test output to network or local files. See the section *Redirection of show Command Output* in *JUNOSe System Basics Configuration Guide, Chapter 2, Command-Line Interface*.

BGP routes must be present in the forwarding table for these commands to work properly. If you run the policy test on incoming routes, soft reconfiguration (configured with the **neighbor soft reconfiguration in** command) must be in effect.

NOTE: The output of these commands is always speculative. It does not reflect the current state of the router.

test bgp ipv6 neighbor
test ip bgp neighbor

- Use to test the effect of BGP policies on a router without implementing the policy.
- You can apply the test to routes advertised to peers or received from peers.
- You can test the following kinds of policies: distribute lists, filter lists, prefix lists, prefix trees, or route maps. If you do not specify a policy, then the test uses whatever policies are currently in effect on the router.

NOTE: If you test the current policies, the results might vary for routes learned before the current policies were activated if you did not clear the forwarding table when the policies changed.

- The following three items apply to the **test ip bgp neighbor** command only:
  - The *address-family identifier* for the route is the same as is used for identifying the neighbor.
  - If you do not specify a route, the test is performed for all routes associated with the *address-family identifier*.
  - Specifying only an address and mask without a route distinguisher causes all routes sharing the address and mask to be considered. Specifying only an address causes a best match to be performed for the route.
- If you completely specify a route with IP address, mask, and route distinguisher, the command displays detailed route information. Otherwise only summary information is shown. Use the **fields** option to select particular fields of interest.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- You can set a weight value for inbound routes filtered with a filter list.

- Example

  host1#**test ip bgp neighbor 10.12.54.21 advertised-routes distribute-list boston5 fields**

- There is no **no** version.

Selecting the Best Path

BGP selects only one route to a destination as the *best path*. When multiple routes to a given destination exist, BGP must determine which of these routes is the best. BGP puts the best path in its routing table and advertises that path to its BGP neighbors.

If only one route exists to a particular destination, BGP installs that route. If multiple routes exist for a destination, BGP uses tie-breaking rules to decide which one of the routes to install in the BGP routing table.

BGP Path Decision Algorithm

BGP determines the best path to each destination for a BGP speaker by comparing path attributes according to the following selection sequence:

1. Select a path with a reachable *next hop*.
2. Select the path with the highest *weight*.
3. If path weights are the same, select the path with the highest *local preference* value.
4. Prefer locally originated routes (network routes, redistributed routes, or aggregated routes) over received routes.
5. Select the route with the shortest *AS-path* length.
6. If all paths have the same AS-path length, select the path based on *origin*: IGP is preferred over EGP; EGP is preferred over Incomplete.
7. If the origins are the same, select the path with lowest *MED* value.
8. If the paths have the same MED values, select the path learned by means of EBGP over one learned by means of IBGP.
9. Select the path with the lowest IGP cost to the next hop.
10. Select the path with the shortest route reflection cluster list. Routes without a cluster list are treated as having a cluster list of length 0.
11. Select the path received from the peer with the lowest BGP router ID.
12. Select the path that was learned from the neighbor with the lowest peer remote address.

The following sections discuss the attributes evaluated in the path decision process. Examples show how you might configure these attributes to influence routing decisions.

Configuring Next-Hop Processing

Routes sent by BGP speakers include the next-hop attribute. The next hop is the IP address of a node on the network that is closer to the advertised prefix. Routers that have traffic destined for the advertised prefix send the traffic to the next hop. The next hop can be the address of the BGP speaker sending the update or of a third-party node. The third-party node does not have to be a BGP speaker.

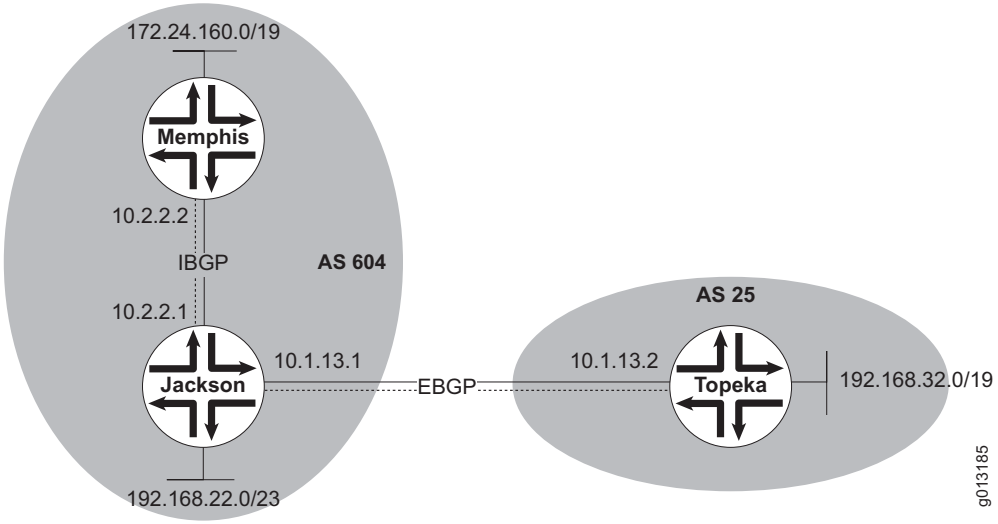
The next-hop attributes conform to the following rules:

- The next hop for EBGP sessions is the IP address of the peer that advertised the route.
- The next hop for IBGP sessions is one of the following:
  - If the route originated inside the AS, the next hop is the IP address of the peer that advertised the route.
  - If the route originated outside the AS—that is, it was injected into the AS by means of an EBGP session—the next hop is the IP address of the external BGP speaker that advertised the route.
- For routes advertised on multiaccess media—such as Frame Relay, ATM, or Ethernet—the next hop is the IP address of the originating router's interface that is connected to the medium.

Next Hops

If you use the **neighbor remote-as** command to configure the BGP neighbors, the next hop is passed according to the rules provided above when networks are advertised. Consider the network configuration shown in Figure 28. Router Jackson advertises 192.168.22.0/23 internally to router Memphis with a next hop of 10.2.2.1. Router Jackson advertises the same network externally to router Topeka with a next hop of 10.1.13.1.

Figure 28: Configuring Next-Hop Processing



Router Memphis advertises 172.24.160/19 with a next hop of 10.2.2.2 to router Jackson. Router Jackson advertises this same network externally to router Topeka with a next hop of 10.1.13.1.

Router Topeka advertises network 192.168.32.0/19 with a next hop of 10.1.13.2 to router Jackson. Because this network originates outside AS 604, router Jackson then internally advertises this network (192.168.32.0/19) to router Memphis with the same next hop, 10.1.13.2 (the IP address of the external BGP speaker that advertised the route).

When router Memphis has traffic destined for 192.168.32.0/19, it must be able to reach the next hop by means of an IGP, because it has no direct connection to 10.1.13.2. Otherwise, router Memphis will drop packets destined for 192.168.32.0/19 because the next-hop address is not accessible. Router Memphis does a lookup in its IP routing table to determine how to reach 10.1.13.2:

Destination	Next Hop
10.1.13.0/24	10.2.2.1

The next hop is reachable through router Jackson, and the traffic can be forwarded.

The following commands configure the routers as shown in Figure 28:

To configure router Jackson:

```
host1(config)#router bgp 604
host1(config-router)#neighbor 10.1.13.2 remote-as 25
host1(config-router)#neighbor 10.2.2.2 remote-as 604
host1(config-router)#network 192.168.22.0 mask 255.255.254.0
```

To configure router Memphis:

```
host2(config)#router bgp 604
host2(config-router)#neighbor 10.2.2.1 remote-as 604
host2(config-router)#network 172.24.160.0 mask 255.255.224.0
```

To configure router Topeka:

```
host3(config)#router bgp 25
host3(config-router)#neighbor 10.1.13.1 remote-as 604
host3(config-router)#network 172.31.64.0 mask 255.255.192.0
```

Additional configuration is required for routers Biloxi, Memphis, and Jackson; the details depend on the IGP running in AS 604.

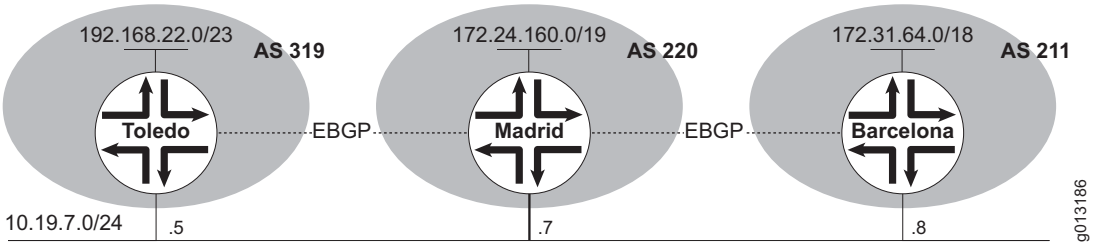
neighbor remote-as

- Use to add an entry to the BGP neighbor table.
- Specifying a neighbor with an AS number that matches the AS number specified in the **router bgp** command identifies the neighbor as internal to the local AS. Otherwise, the neighbor is treated as an external neighbor.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately.
- Use the **no** version to remove an entry from the table.

Next-Hop-Self

In some circumstances, using a third-party next hop causes routing problems. These configurations typically involve nonbroadcast multiaccess (NBMA) media. To better understand this situation, first consider a broadcast multiaccess (BMA) media network, as shown in Figure 29.

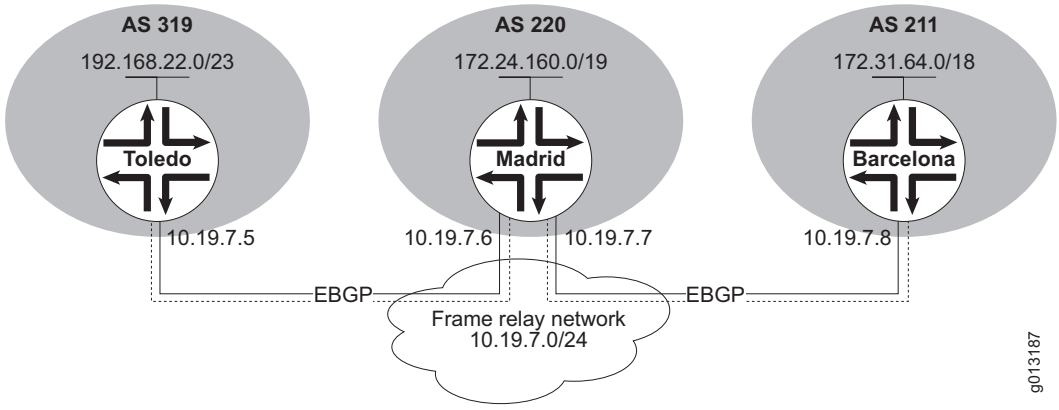
Figure 29: Next-Hop Behavior for Broadcast Multiaccess Media



Routers Toledo, Madrid, and Barcelona are all on the same Ethernet network, which has a prefix of 10.19.7.0/24. When router Toledo advertises prefix 192.168.22.0/23 to router Madrid, it sets the next-hop attribute to 10.19.7.5. Before router Madrid advertises this prefix to router Barcelona, it sees that its own IP address, 10.19.7.7, is on the same subnet as the next hop for the advertised prefix. If router Barcelona can reach router Madrid, then it should be able to reach router Toledo. Router Madrid therefore advertises 192.168.22.0/23 to router Barcelona with a next-hop attribute of 10.19.7.5.

Now consider Figure 30, which shows the same routers on a Frame Relay—NBMA—network.

Figure 30: Next-Hop Behavior for Nonbroadcast Multiaccess Media



Routers Toledo and Madrid are EBGP peers, as are routers Madrid and Barcelona. When router Toledo advertises prefix 192.168.22.0/23 to router Madrid, router Madrid makes the same comparison as in the BMA example, and leaves the next-hop attribute intact when it advertises the prefix to router Barcelona. However, router Barcelona will not be able to forward traffic to 192.168.22.0/23, because it does not have a direct PVC connection to router Toledo and cannot reach the next hop of 10.19.7.5.

You can use the **neighbor next-hop-self** command to correct this routing problem. If you use this command to configure router Madrid, the third-party next hop advertised by router Toledo is not advertised to router Barcelona. Instead, router Madrid advertises 192.168.22.0/23 with the next-hop attribute set to its own IP address, 10.19.7.7. Router Barcelona now forwards traffic destined for 192.168.22.0/23 to the next hop, 10.19.7.7. Router Madrid then passes the traffic along to router Toledo.

To disable third-party next-hop processing, configure router Madrid as follows:

```
host1(config)#router bgp 319
host1(config-router)#neighbor 10.19.7.8 remote-as 211
host1(config-router)#neighbor 10.19.7.8 next-hop-self
```

neighbor next-hop-self

- Use to prevent third-party next hops from being used on NBMA media such as Frame Relay. This command is useful in nonmeshed networks such as Frame Relay or where BGP neighbors may not have direct access to the same IP subnet.
- Forces the BGP speaker to report itself as the next hop for an advertised route it learned from a neighbor.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override the characteristic for a specific member of the peer group.

- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

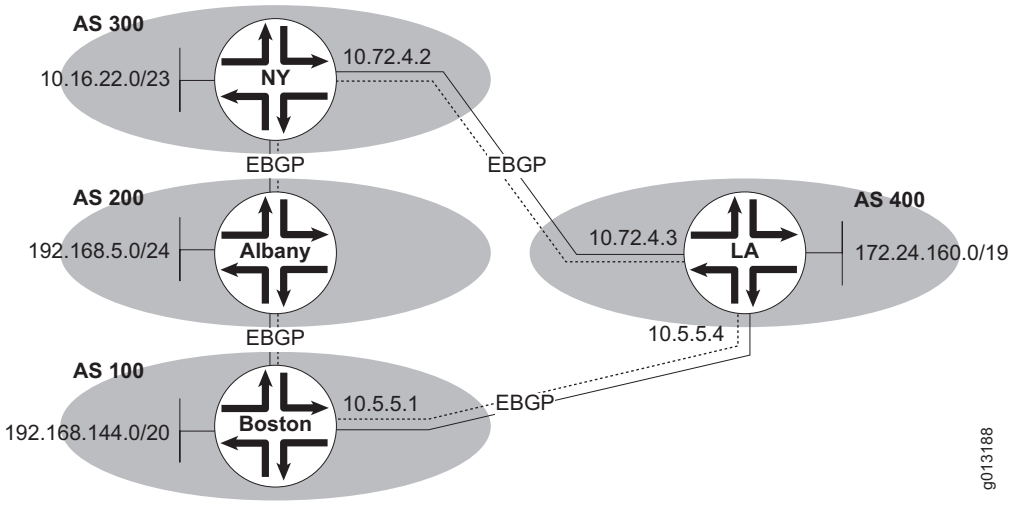
- Issuing this command automatically removes the **neighbor next-hop-unchanged** configuration (enabled or disabled) on the peer or peer group. Issuing the **no** or **default** version of this command has no effect on the **neighbor next-hop-unchanged** configuration.
- Use the **no** version to disable this feature (and therefore enable next-hop processing of BGP updates). Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Assigning a Weight to a Route

You can assign a weight to a route when more than one route exists to the same destination. A weight indicates a preference for that particular route over the other routes to that destination. The higher the assigned weight, the more preferred the route. By default, the route weight is 32768 for paths originated by the router, and 0 for other paths.

In the configuration shown in Figure 31, routers Boston and NY both learn about network 192.68.5.0/24 from AS 200. Routers Boston and NY both propagate the route to router LA. Router LA now has two routes for reaching 192.68.5.0/24 and must decide the appropriate route. If you prefer that router LA direct traffic through router Boston, you can configure router LA so that the weight of routes coming from router Boston are higher—more preferred—than the routes coming from router NY. Router LA subsequently prefers routes received from router Boston and therefore uses router Boston as the next hop to reach network 192.68.5.0/24.

Figure 31: Assigning a Weight to a Neighbor Connection



You can use any of the following three ways to set the weights in routes coming in from router Boston:

- The **neighbor weight** command
- The **set weight** command in route maps
- An AS-path access list

Using the neighbor weight Command

The following commands assign a weight of 1000 to all routes router LA receives from AS 100 and assign a weight of 500 to all routes router LA receives from AS 300:

```
host1(config)#router bgp 400
host1(config-router)#neighbor 10.5.5.1 remote-as 100
host1(config-router)#neighbor 10.5.5.1 weight 1000
host1(config-router)#neighbor 10.72.4.2 remote-as 300
host1(config-router)#neighbor 10.72.4.2 weight 500
```

Router LA sends traffic through router Boston in preference to router NY.

Using a Route Map

A route map instance is a set of conditions with an assigned number. The number after the **permit** keyword designates an instance of a route map. For example, instance 10 of route map 10 begins with the following:

```
host1(config)#route-map 10 permit 10
```

In the following commands to configure router LA, instance 10 of route map 10 assigns a weight of 1000 to any routes from AS 100. Instance 20 assigns a weight of 500 to routes from any other AS.

```
host1(config)#router bgp 400
host1(config-router)#neighbor 10.5.5.1 remote-as 100
host1(config-router)#neighbor 10.5.5.1 route-map 10 in
host1(config-router)#neighbor 10.72.4.2 remote-as 300
host1(config-router)#neighbor 10.72.4.2 route-map 20 in
host1(config-router)#exit
host1(config)#route-map 10
host1(config-route-map)#set weight 1000
host1(config-route-map)#route-map 20
host1(config-route-map)#set weight 500
```

See *JUNOSe IP Services Configuration Guide, Chapter 1, Configuring Routing Policy* for more information about using route maps.

Using an AS-Path Access List

The following commands assign weights to routes filtered by AS-path access lists on router LA:

```
host1(config)#router bgp 400
host1(config-router)#neighbor 10.5.5.1 remote-as 100
host1(config-router)#neighbor 10.5.5.1 filter-list 1 weight 1000
host1(config-router)#neighbor 10.72.4.2 remote-as 300
host1(config-router)#neighbor 10.72.4.2 filter-list 2 weight 500
host1(config-router)#exit
host1(config)#ip as-path access-list 1 permit ^100_
host1(config)#ip as-path access-list 2 permit ^300_
```

Access list 1 permits any route whose AS-path attribute begins with 100 (specified by ^). This permits routes that pass through router Boston, whether they originate in AS 100 (AS path = 100) or AS 200 (AS path = 100 200) or AS 300 (AS path = 100 200 300). Access list 2 permits any route whose AS-path attribute begins with 300. This permits routes that pass through router NY, whether they originate in AS 300 (AS path = 300) or AS 200 (AS path = 300 200) or AS 100 (AS path = 300 200 100).

The **neighbor filter-list** commands assign a weight attribute of 1000 to routes passing through router Boston and a weight attribute of 500 to routes passing through router NY. Regardless of the origin of the route, routes learned through router Boston are preferred.

ip as-path access-list

- Use to define a BGP access list; use the **neighbor filter-list** command to apply a specific access list.
- You can apply access list filters on inbound or outbound BGP routes, or both.
- You can **permit** or **deny** access for a route matching the condition(s) specified by the regular expression.
- If the regular expression matches the representation of the AS path of the route as an ASCII string, then the *permit* or *deny* condition applies.
- The AS path allows substring matching. For example, the regular expression *20* matches AS path = *20* and AS path = *100 200 300*, because *20* is a substring of each path. To disable substring matching and constrain matching to only the specified attribute string, place the underscore (_) metacharacter on both sides of the string, for example *_20_*.
- The AS path does not contain the local AS number.
- Use the **no** version to remove a single access list entry if **permit** or **deny** and a *path-expression* are specified. Otherwise, the entire access list is removed.

neighbor filter-list

- Use to apply an AS-path access list to advertisements inbound from or outbound to the specified neighbor, or to assign a weight to incoming routes that match the AS-path access list.
- You can specify an optional weight value with the **weight** keyword to assign a relative importance to incoming routes matching the AS-path access list.
- The name of the access list is a string of up to 32 characters.
- You can apply the filter to incoming or outgoing advertisements with the **in** or **out** keywords.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer. However, you cannot configure a member of a peer group to override the inherited peer group characteristic for outbound policy.
- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to remove the filter list.

neighbor weight

- Use to assign a weight to a neighbor connection.
- All routes learned from this neighbor will have the assigned weight initially.
- The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network.
- The weights assigned with the **set weight** commands in a route map override the weights assigned with the **neighbor weight** and **neighbor filter-list** commands.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- New policy values are applied to all routes that are sent (outbound policy) or received (inbound policy) after you issue the command.

  To apply the new policy to routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to perform a soft clear or hard clear of the current BGP session.

  Behavior is different for outbound policies configured for peer groups for which you have enabled Adj-RIBs-Out. If you change the outbound policy for such a peer group and want to fill the Adj-RIBs-Out table for that peer group with the results of the new policy, you must use the **clear ip bgp peer-group** command to perform a hard clear or outbound soft clear of the peer group. You cannot merely perform a hard clear or outbound soft clear for individual peer group members because that causes BGP to resend only the contents of the Adj-RIBs-Out table.

- Use the **no** version to remove the weight assignment.

See *Access Lists* on page 80 for more information about using access lists.

Configuring the Local-Pref Attribute

The local-pref attribute specifies the preferred path among multiple paths to the same destination. The preferred path is the one with the higher preference value. Local preference is used only within an AS, to select an exit point.

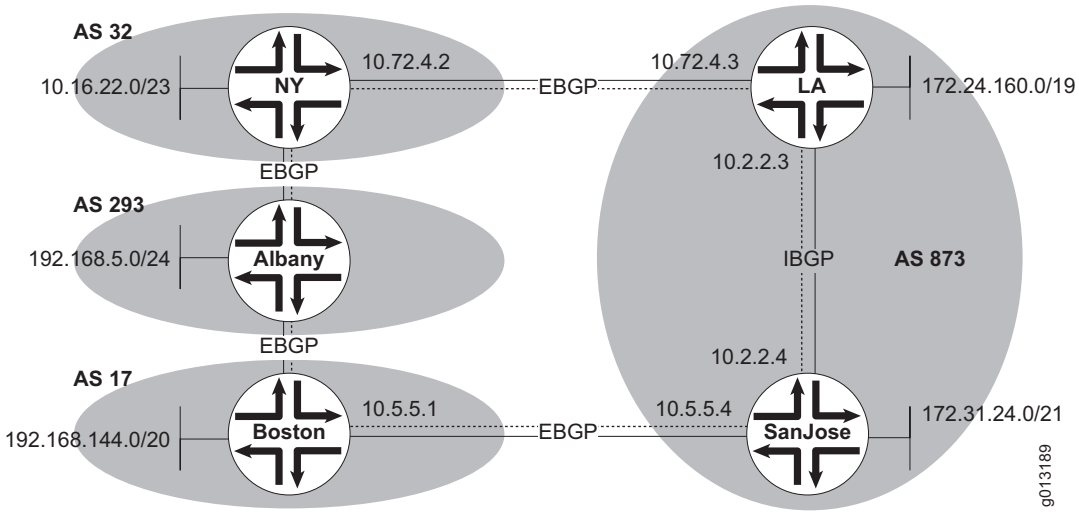
To configure the local preference of a BGP path, you can do one of the following:

- Use the **bgp default local-preference** command to set the local-preference attribute.
- Use a route map to set the local-pref attribute.

Using the bgp default local-preference Command

In Figure 32, AS 873 receives updates for network 192.168.5.0/24 from AS 32 and AS 17.

Figure 32: Configuring the Local-Preference Attribute



The following commands configure router LA:

```
host1(config-router)#router bgp 873
host1(config-router)#neighbor 10.72.4.2 remote-as 32
host1(config-router)#neighbor 10.2.2.4 remote-as 873
host1(config-router)#bgp default local-preference 125
```

The following commands configure router SanJose:

```
host2(config-router)#router bgp 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#bgp default local-preference 200
```

Router LA sets the local preference for all updates from AS 32 to 125. Router SanJose sets the local preference for all updates from AS 17 to 200. Because router LA and router SanJose exchange local preference information within AS 873, they both recognize that routes to network 192.168.5.0/24 in AS 293 have a higher local preference when they come to AS 873 from AS 17 than when they come from AS 32. As a result, both router LA and router SanJose prefer to reach this network through router Boston in AS 17.

bgp default local-preference

- Use to change the default local preference value.
- Changes apply automatically whenever BGP subsequently runs the best-path decision process for a destination prefix; that is, whenever a best route is picked for a given prefix.

To force BGP to run the decision process on routes already received, you must use the **clear ip bgp** command to perform an inbound soft clear or hard clear of the current BGP session.

- Use the **no** version to restore the default value, 100.

Using a Route Map to Set the Local Preference

When you use a route map to set the local preference you have more flexibility in selecting routes for which you can set a local preference based on many criteria, including AS. In the previous section, all updates received by router SanJose were set to a local preference of 200.

Using a route map, you can specifically assign a local preference for routes from AS 17 that pass through AS 293.

The following commands configure router SanJose.

```
host2(config-router)#router bgp 873
host2(config-router)#neighbor 10.2.2.3 remote-as 873
host2(config-router)#neighbor 10.5.5.1 remote-as 17
host2(config-router)#neighbor 10.5.5.1 route-map 10 in
host2(config-router)#exit
host2(config)#ip as-path access-list 1 permit ^17 293$
host2(config)#route-map 10 permit 10
host2(config-route-map)#match as-path 1
host2(config-route-map)#set local-preference 200
host2(config-route-map)#exit
host2(config)#route-map 10 permit 20
```

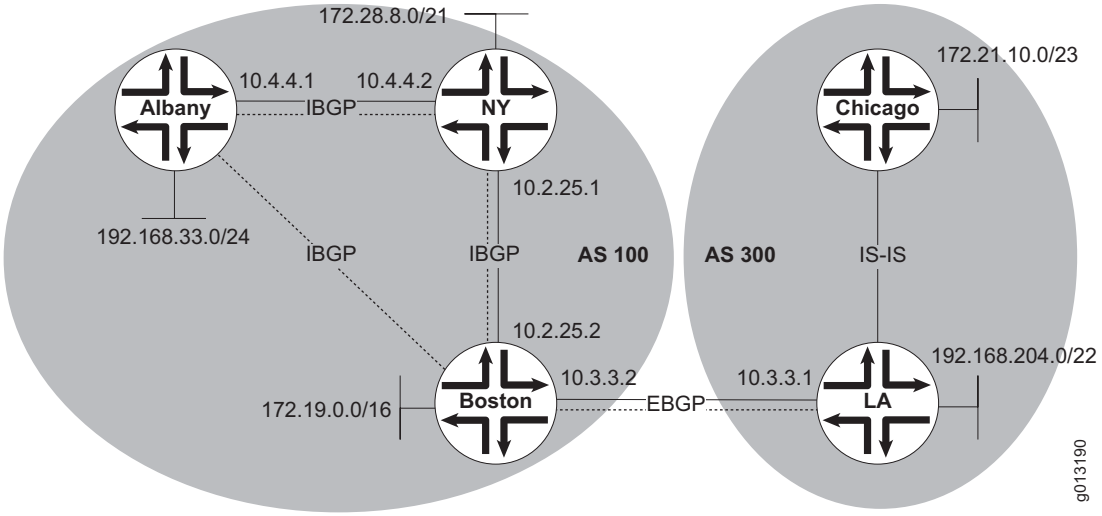
Router SanJose sets the local-pref attributes to 200 for routes originating in AS 293 and passing last through AS 17. All other routes are accepted (as defined in instance 20 of the route map 10), but their local preference remains at the default value of 100, indicating a less-preferred path.

Understanding the Origin Attribute

BGP uses the origin attribute to describe how a route was learned at the origin—the point where the route was injected into BGP. The origin of the route can be one of three values:

- IGP—Indicates that the route was learned by means of an IGP and, therefore, is internal to the originating AS. All routes advertised by the **network** command have an origin of IGP.
- EGP—Indicates that the route was learned by means of an EGP.
- Incomplete—Indicates that the origin of the route is unknown—that is, learned from something other than IGP or EGP. All routes advertised by the **redistribute** command—such as static routes—have an origin of Incomplete. An origin of Incomplete occurs when a route is redistributed into BGP.

Figure 33: The Origin Attribute



Consider the sample topology shown in Figure 33. Because routers Albany and Boston are not directly connected, they learn the path to each other by means of an IGP (not illustrated).

The following commands configure router Boston:

```
host1(config)#ip route 172.31.125.100 255.255.255.252
host1(config)#router bgp 100
host1(config-router)#neighbor 10.2.25.1 remote-as 100
host1(config-router)#neighbor 10.4.4.1 remote-as 100
host1(config-router)#neighbor 10.3.3.1 remote-as 300
host1(config-router)#network 172.19.0.0
host1(config-router)#redistribute static
```

The following commands configure router NY:

```
host2(config)#router bgp 100
host2(config-router)#neighbor 10.4.4.1 remote-as 100
host2(config-router)#neighbor 10.2.25.2 remote-as 100
host2(config-router)#network 172.28.8.0 mask 255.255.248.0
```

The following commands configure router Albany:

```
host3(config)#router bgp 100
host3(config-router)#neighbor 10.4.4.2 remote-as 100
host3(config-router)#neighbor 10.2.25.2 remote-as 100
host3(config-router)#network 192.168.33.0 mask 255.255.255.0
```

The following commands configure router LA:

```
host4(config)#router bgp 300
host4(config-router)#neighbor 10.3.3.2 remote-as 100
host4(config-router)#network 192.168.204.0 mask 255.255.252.0
host4(config-router)#redistribute isis
```

Consider how route 172.21.10.0/23 is passed along to the routers in Figure 33:

1. IS-IS injects route 172.21.10.0/23 from router Chicago into BGP on router LA. BGP sets the origin attribute to Incomplete (because it is a redistributed route) to indicate how BGP originally became aware of the route.

2. Router Boston learns about route 172.21.10.0/23 by means of EBGP from router LA.

3. Router NY learns about route 172.21.10.0/23 by means of IBGP from router Boston.

The value of the origin attribute for a given route remains the same, regardless of where you examine it. Table 21 shows this for all the routes known to routers NY and LA.

Table 21: Origin and AS Path for Routes Viewed on Different Routers

Route	Router	Origin	AS Path
192.168.204.0/22	Albany	IGP	300
192.168.204.0/22	Boston	IGP	300
192.168.204.0/22	NY	IGP	300
192.168.204.0/22	LA	IGP	empty
172.21.10.0/23	Albany	Incomplete	300
172.21.10.0/23	Boston	Incomplete	300
172.21.10.0/23	NY	Incomplete	300
172.21.10.0/23	LA	Incomplete	empty
172.28.8.0/21	Albany	IGP	empty
172.28.8.0/21	Boston	IGP	empty
172.28.8.0/21	NY	IGP	empty
172.28.8.0/21	LA	IGP	100
172.31.125.100	Albany	Incomplete	empty
172.31.125.100	Boston	Incomplete	empty
172.31.125.100	NY	Incomplete	empty
172.31.125.100	LA	Incomplete	100
172.19.0.0/16	Albany	IGP	empty
172.19.0.0/16	Boston	IGP	empty
172.19.0.0/16	NY	IGP	empty
172.19.0.0/16	LA	IGP	100
192.168.330/24	Albany	IGP	empty
192.168.330/24	Boston	IGP	empty
192.168.330/24	NY	IGP	empty
192.168.330/24	LA	IGP	100

As a matter of routing policy, you can specify an origin for a route with a **set origin** clause in a redistribution route map. Changing the origin enables you to influence which of several routes for the same destination prefix is selected as the best route. In practice, changing the origin is rarely done.

Understanding the AS-Path Attribute

The AS-path attribute is a list of the ASs through which a route has passed. Whenever a route enters an AS, BGP prepends the AS number to the AS-path attribute. This feature enables network operators to track routes, but it also enables the detection and prevention of routing loops.

Consider the following sequence of events for the routers shown in Figure 34:

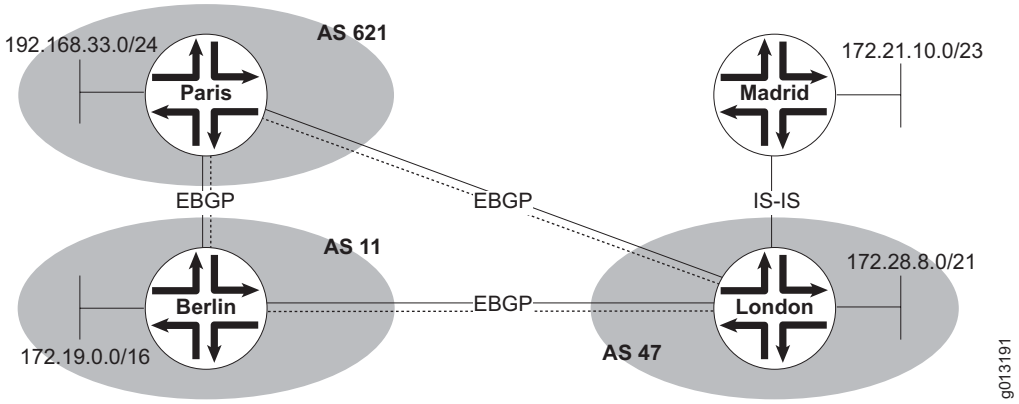
1. Route 172.21.10.0/23 is injected into BGP by means of router London in AS 47.

2. Suppose router London advertises that route to router Paris in AS 621. As received by router Paris, the AS-path attribute for route 172.21.10.0/23 is 47.

3. Router Paris advertises the route to router Berlin in AS 11. As received by router Berlin, the AS-path attribute for route 172.21.10.0/23 is 621 47.

4. Router Berlin advertises the route to router London in AS 47. As received by router London, the AS-path attribute for route 172.21.10.0/23 is 11 621 47.

Figure 34: AS-Path Attributes



A routing loop exists if router London accepts the route from router Berlin. Router London can choose not to accept the route from router Berlin because it recognizes from the AS-path attribute (11 621 47) that the route originated in its own AS 47.

As a matter of routing policy, you can prepend additional AS numbers to the AS-path attribute for a route with a **set as-path prepend** clause in an outbound route map. Changing the AS path enables you to influence which of several routes for the same destination prefix is selected as the best route.

Configuring a Local AS

You can change the local AS of a BGP peer or peer group within the current address family with the **neighbor local-as** command. By using different local AS numbers for different peers, you can avoid or postpone AS renumbering in the event the ASs are merged.

neighbor local-as

- Use to assign a local AS to the given BGP peer or peer group.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately and automatically bounces the BGP session.
- Use the **no** version for an individual peer to restore the value set for the peer group, if present, or set globally for BGP with the **router bgp** command. Use the **no** version for a peer group to restore the value set globally for BGP.

The following example commands change the local AS number for peer 104.4.2 from the global local AS of 100 to 32:

```
host1(config)#router bgp 100
host1(config-router)#address-family ipv4 unicast vrf boston
host1(config-router)#neighbor 10.4.4.2 remote-as 645
host1(config-router)#neighbor 10.4.4.2 local-as 32
```

Configuring the MED Attribute

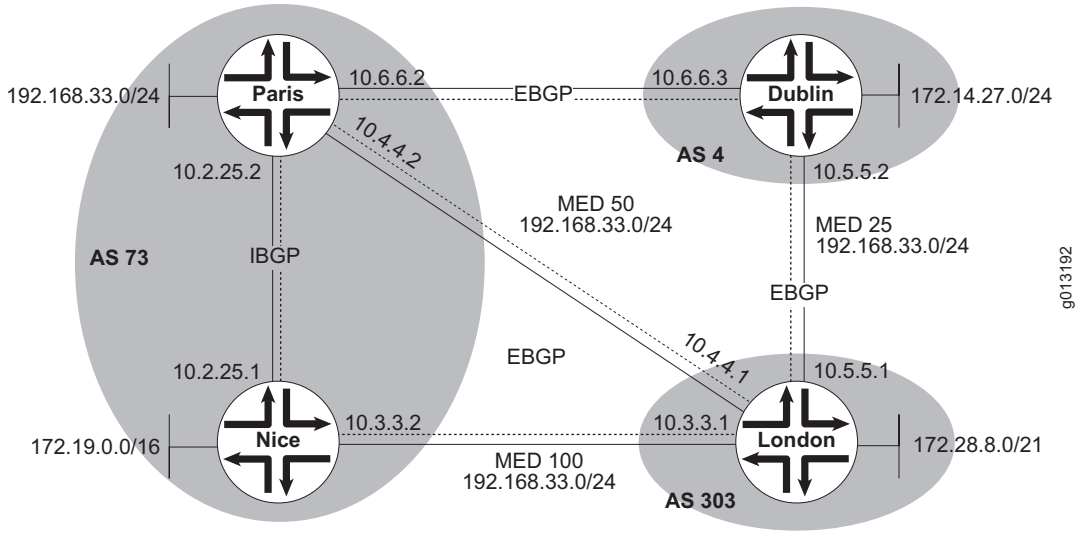
If two ASs connect to each other in more than one place, one link or path might be a better choice to reach a particular prefix within or behind one of the ASs. The MED value is a metric expressing a degree of preference for a particular path. Lower MED values are preferred.

Whereas the Local Preference attribute is used only within an AS (to select an exit point), the MED attribute is exchanged between ASs. A router in one AS sends the MED to inform a router in another AS which path the second router should use to reach particular destinations. If you are the administrator of the second AS, you must therefore trust that the router in the first AS is providing information that is truly beneficial to your AS.

You configure the MED on the sending router by using the **set metric** command in an outbound route map. Unless configured otherwise, a receiving router compares MED attributes only for paths from external neighbors that are members of the same AS. If you want MED attributes from neighbors in different ASs to be compared, you must issue the **bgp always-compare-med** command.

In Figure 35, router London in AS 303 can reach 192.168.33.0/24 in AS 73 through router Paris or through router Nice to router Paris.

Figure 35: Configuring the MED



The following commands configure router London:

```
host1(config)#router bgp 303
host1(config-router)#neighbor 10.4.4.2 remote-as 73
host1(config-router)#neighbor 10.3.3.2 remote-as 73
host1(config-router)#neighbor 10.5.5.2 remote-as 4
host1(config-router)#network 122.28.8.0 mask 255.255.248.0
```

The following commands configure router Paris:

```
host2(config)#router bgp 73
host2(config-router)#neighbor 10.4.4.1 remote-as 303
host2(config-router)#neighbor 10.4.4.1 route-map 10 out
host2(config-router)#neighbor 10.2.25.1 remote-as 73
host2(config-router)#neighbor 10.6.6.1 remote-as 4
host2(config-router)#neighbor 10.6.6.1 route-map 10 out
host2(config-router)#network 192.168.33.0 mask 255.255.255.0
host2(config-router)#exit
host2(config)#route-map 10 permit 10
host2(config-route-map)#set metric 50
```

The following commands configure router Nice:

```
host3(config)#router bgp 73
host3(config-router)#neighbor 10.3.3.1 remote-as 303
host3(config-router)#neighbor 10.3.3.1 route-map 10 out
host3(config-router)#neighbor 10.2.25.2 remote-as 73
host3(config-router)#network 172.19.0.0
host3(config-router)#exit
host3(config)#route-map 10 permit 10
host3(config-route-map)#set metric 100
```

The following commands configure router Dublin:

```
host4(config)#router bgp 4
host4(config-router)#neighbor 10.5.5.1 remote-as 303
host4(config-router)#neighbor 10.5.5.1 route-map 10 out
host4(config-router)#neighbor 10.6.6.2 remote-as 73
host4(config-router)#network 172.14.27.0 mask 255.255.255.0
host4(config-router)#exit
host4(config)#route-map 10 permit 10
host4(config-route-map)#set metric 25
```

Router London receives updates regarding route 192.168.33.0/24 from both router Nice and router Paris. Router London compares the MED values received from the two routers: Router Nice advertises a MED of 100 for the route, whereas router Paris advertises a MED of 50. On this basis, router London prefers the path through router Paris.

Because BGP by default compares only MED attributes of routes coming from the same AS, router London can compare only the MED attributes for route 192.168.33.0/24 that it received from routers Paris and Nice. It cannot compare the MED received from router Dublin, because router Dublin is in a different AS than routers Paris and Nice.

However, you can use the **bgp always-compare-med** command to configure router London to take into account the MED attribute from router Dublin as follows:

```
host1(config)#router bgp 303
host1(config-router)#neighbor 10.4.4.2 remote-as 73
host1(config-router)#neighbor 10.3.3.2 remote-as 73
host1(config-router)#neighbor 10.5.5.2 remote-as 4
host1(config-router)#network 122.28.8.0 mask 255.255.248.0
host1(config-router)#bgp always-compare-med
```

Router Dublin advertises a MED of 25 for route 192.168.33.0/24, which is lower—more preferred—than the MED advertised by router Paris or router Nice. However, the AS path for the route through router Dublin is longer than that through router Paris. The AS path is the same length for router Paris and router Nice, but the MED advertised by router Paris is lower than that advertised by router Nice. Consequently, router London prefers the path through router Paris.

Suppose, however that router Dublin was not configured to set the MED for route 192.168.33.0/24 in its outbound route map 10. Would router London receive a MED of 50 passed along by router Paris through router Dublin? No, because the MED attribute is nontransitive. Router Dublin does not transmit any MED that it receives. A MED is only of value to a direct peer.

bgp always-compare-med

- Use to enable the comparison of the MED for paths from neighbors in different ASs.
- Unless you specify the **bgp always-compare-med** command, the router compares MED attributes only for paths from external neighbors that are in the same AS.
- The BGP path decision algorithm selects a lower MED value over a higher one.
- Unlike local preferences, the MED attribute is exchanged between ASs, but does not leave the AS.
- The value is used for decision making within the AS only.
- When BGP propagates a route received from outside the AS to another AS, it removes the MED.
- Example

  host1(config-router)#**bgp always-compare-med**

- Changes apply automatically whenever BGP subsequently runs the best-path decision process for a destination prefix; that is, whenever a best route is picked for a given prefix.

  To force BGP to run the decision process on routes already received, you must use the **clear ip bgp** command to perform an inbound soft clear or hard clear of the current BGP session.

- Use the **no** version to disable the feature.

set metric

- Use to set the metric value—for BGP, the MED—for a route.
- Sets an absolute metric. You cannot use both an absolute metric and a relative metric within the same route map sequence. Setting either metric overrides any previously configured value.
- Example

  host1(config)#**route-map nyc1 permit 10**
  host1(config-route-map)#**set metric 10**

- Use the **no** version to delete the set clause from a route map.

Missing MED Values

By default, a route that arrives with no MED value is treated as if it had a MED of 0, the most preferred value. You can use the **bgp bestpath missing-as-worst** command to specify that a route with any MED value is always preferred to a route that is missing the MED value.

bgp bestpath missing-as-worst

- Use to set a missing MED value to infinity, the least preferred value.
- After issuing this command, a route missing the MED is always preferred less than any route that has a MED configured.
- Example

  host1(config-router)#**bgp bestpath missing-as-worst**

- Changes apply automatically whenever BGP subsequently runs the best-path decision process for a destination prefix; that is, whenever a best route is picked for a given prefix.

  To force BGP to run the decision process on routes already received, you must use the **clear ip bgp** command to perform an inbound soft clear or hard clear of the current BGP session.

- Use the **no** version to restore the default condition, where a missing MED value is set to 0, the most preferred value.

Comparing MED Values Within a Confederation

A BGP speaker within a confederation of sub-ASs might need to compare routes to determine the best path to a destination. By default, BGP does not use the MED value when comparing routes originated in different sub-ASs within the confederation to which the BGP speaker belongs. (Within the confederation, routes learned from different sub-ASs are treated as having originated in different places.) You can use the **bgp bestpath med confed** command to force MED values to be taken into account within a confederation.

bgp bestpath med confed

- Use to specify that BGP take into account the MED when comparing routes originated in different sub-ASs within the confederation to which the BGP speaker belongs.
- This command does not affect the comparison of routes that are originated in other ASs and does not affect the comparison of routes that are originated in other confederations.
- Example

  host1(config-router)#**bgp bestpath med confed**

- Changes apply automatically whenever BGP subsequently runs the best-path decision process for a destination prefix; that is, whenever a best route is picked for a given prefix.

  To force BGP to run the decision process on routes already received, you must use the **clear ip bgp** command to perform an inbound soft clear or hard clear of the current BGP session.

- Use the **no** version to restore the default, where the MED is not taken into account.

Suppose a BGP speaker has three routes to prefix 10.10.0.0/16:

- Route 1 is originated by sub-AS 1 inside the confederation.
- Route 2 is originated by sub-AS 2 inside the confederation.
- Route 3 is originated by AS 3 outside the confederation.

BGP compares these routes to each other to determine the best path to the prefix. If you have issued the **bgp bestpath med confed** command, BGP takes into account the MED when comparing Route 1 with Route 2. However, BGP does not take into account the MED when comparing Route 3 with either Route 1 or Route 2, because Route 3 originates outside the confederation.

Capability Negotiation

The router accepts connections from peers that perform capability negotiation. Capabilities are negotiated by means of the open messages that are exchanged when the session is established. The router supports the following capabilities:

- Cisco-proprietary route refresh—Capability code 128
- Cooperative route filtering—Capability code 3
- Deprecated dynamic capability negotiation—Capability code 66
- Dynamic capability negotiation—Capability code 67
- Four-octet AS numbers—Capability code 65
- Graceful restart—Capability code 64
- Multiprotocol extensions—Capability code 1
  - address family IPv4 unicast—AFI 1 SAFI 1
  - address family IPv4 multicast—AFI 1 SAFI 2
  - address family IPv4 unicast and multicast—AFI 1 SAFI 3
  - address family VPN-IPv4 unicast—AFI 1 SAFI 128
- Route refresh—Capability code 2

The router advertises these capabilities—except for the cooperative route filtering capability—by default. You can prevent the advertisement of specific capabilities with the **no neighbor capability** command. You can also use this command to prevent all capability negotiation with the specified peer.

NOTE: The graceful restart capability is controlled with the **bgp graceful-restar**t and **neighbor graceful-restart** commands rather than the **neighbor capability** command. However, the **no neighbor capability** command will prevent negotiation of the graceful restart capability.

Cooperative Route Filtering

The cooperative route filtering capability—also referred to as outbound route filtering (ORF)—enables a BGP speaker to send an inbound route filter to a peer and have the peer install it as an outbound filter on the remote end of the session.

You must specify both the type of inbound filter (ORF type) and the direction of ORF capability. The router currently supports prefix-lists as the inbound filter sent by the BGP speaker. The inbound filter sent by the BGP speaker can be a prefix list or a Cisco proprietary prefix list. The BGP speaker must indicate whether it will send inbound filters to peers, accept inbound filters from peers, or both. The router supports both standard and Cisco-proprietary orf messages.

Dynamic Capability Negotiation

If both peers acknowledge support of dynamic capability negotiation, then at any subsequent point after the session is established, either peer can send a capabilities message to the other indicating a desire to negotiate another capability or to remove a previously negotiated capability.

The data field of the capability message contains a list of all the capabilities that can be dynamically negotiated. In earlier versions, now deprecated, the data field did not carry this information. Use the **dynamic-capability-negotiation** keyword to include the list. Use the **deprecated-dynamic-capability-negotiation** keyword to exclude sending the list.

Nondynamic capability negotiation is supported for the cooperative route filtering, four-octet AS numbers, deprecated dynamic capability negotiation, and dynamic capability negotiation capabilities. Dynamic capability negotiation of these capabilities is not supported.

If both sides of the connection advertise support for the new dynamic capability negotiation capability, then the peers negotiate which capabilities are dynamic and which are not.

If both sides of the connection advertise support only for the deprecated dynamic capability negotiation, then the BGP speaker uses dynamic capability negotiation for all capabilities that allow it without attempting to negotiate this with the peer.

Four-Octet AS Numbers

BGP speakers that support four-octet AS and sub-AS numbers are sometimes referred to as “new” speakers. The four-octet AS numbers are employed by the AS-path and aggregator attributes. “Old” speakers are those that do not support the four-octet numbers.

Two new transitional optional attributes, new-as-path and new-aggregator, are used to carry the four-octet numbers across the old speakers. A new speaker communicating with an old speaker will send the new attributes with the four-octet numbers for locally-originated and propagated routes. The old speaker propagates the new attributes for received routes. The new speaker also sends the AS-path and aggregator attributes with two-octet numbers; any AS number greater than 65535 is replaced with a reserved AS number, 23456.

Graceful Restarts

When BGP restarts on a router, all of the router's BGP peers detect that the BGP session transitioned from up to down. The transition causes a routing flap throughout the network as the peers recalculate their best routes in light of the loss of routes from that peering session.

The BGP graceful restart capability reduces the network disruption that normally results from a peer session going down. If the session is with a peer that had previously advertised the graceful restart capability, the receiving BGP speaker marks all routes from that peer in the BGP routing table as stale. BGP keeps these stale routes for a limited time and continues to use these routes to forward traffic. Any existing stale routes from that peer are deleted to account for consecutive restarts.

When the restarting peer reestablishes the session, the receiving BGP speaker replaces the stale routes with the fresh routes it receives from the peer. The restarted peer sends an End-of-RIB marker to signal when it has finished sending all its routes to the BGP speaker. Until this point, BGP has still been using the stale routes to forward traffic. Upon receipt of the End-of-RIB marker, the BGP speaker flushes any remaining stale routes from the restarted peer.

The End-of-RIB marker is an update message that contains no advertised or withdrawn prefixes; it is sent only to BGP speakers that have previously advertised the graceful restart capability.

The receiving speaker also sends its own routes to the restarted speaker, and sends an End-of-RIB marker when it completes the update. The restarted peer defers reinitiating the BGP best-path selection process until it has received this marker from all peers with which it had a session in the established state and from which it had received an End-of-RIB marker before it restarted.

After running the selection process to pick the best route to all prefixes using the fresh routes, BGP then installs the best routes in the IP routing table on the restarted peer. Any of these that are best overall routes to a prefix are then pushed by the router to the forwarding tables on the line modules.

By waiting for all restarted peers to send the End-of-RIB marker, BGP risks delaying the initiation of the best path decision process indefinitely due to a single very slow peer. For a specific peer, you can avoid this delay by hard clearing the peer or issuing the **clear ip bgp wait-end-of-rib** command. Either method removes that peer from the set of peers for which BGP is awaiting an End-of-RIB marker. Alternatively, you can minimize this effect by using the **bgp graceful-restart path-selection-defer-time-limit** command to specify a maximum period that the restarted peer waits for the marker from its peers.

Note that the receiving peer does not defer its best-path selection process while waiting for a restarted peer to reestablish a session. The receiving peer continues to use the stale routes from the restarted peer in the decision process. When it flushes stale routes, the receiving peer then uses the freshly updated routes.

A restarting peer must bring the session back up and refresh its routes within a limited period, or BGP on the receiving peer will flush all the stale routes. When a BGP speaker advertises the graceful restart capability, it also advertises how long it expects to take to reestablish a session if it restarts. If the session is not reestablished within this restart period, the speaker's peers flush the stale routes from the speaker. You can use the **bgp graceful-restart restart-time** command to modify the restart period advertised to all peers; the **neighbor graceful-restart restart-time** command modifies the restart period advertised to specific peers or peer groups. A receiving peer starts the timer as soon as it recognizes that the session with the restarting peer has transitioned to down.

The receiving peer also has a configurable timer that starts when it recognizes that the session with the restarting peer has gone down. The **bgp graceful-restart stalepaths-time** command determines how long a receiving peer is willing to use stale paths from any restarted peer; the **neighbor graceful-restart stalepaths-time** command does the same for a specified restarted peer or peer group. If the receiving peer does not receive an End-of-RIB marker from the restarted peer before the stalepaths timer expires, the receiving peer flushes all stale routes from the peer.

bgp graceful-restart

- Use to enable the BGP graceful restart capability.
- Advertisement of the graceful restart capability is enabled by default.
- The **no neighbor capability negotiation** command prevents the advertisement of all BGP capabilities, including graceful restart, to the specified peers.
- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**bgp graceful-restart**

- Use the **no** version to disable advertisement of the graceful restart capability. Use the **default** version to restore the default condition, advertising this capability.

bgp graceful-restart path-selection-defer-time-limit

- Use to set the maximum time a restarted BGP speaker defers reinitiating the best-path selection process.
- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**bgp graceful-restart path-selection-defer-time-limit 180**

- Use the **no** version to restore the default value, 120 seconds.

bgp graceful-restart restart-time

- Use to set the time BGP advertises to all peers within which it expects to reestablish a session after restarting. Peers flush stale routes from the speaker if the session is not restarted within this period.
- Specify an interval shorter than the stalepaths time.
- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**bgp graceful-restart restart-time 240**

- Use the **no** version to restore the default value, 120 seconds.

bgp graceful-restart stalepaths-time

- Use to set the maximum time BGP waits to receive an End-of-RIB marker from any restarted peer before flushing all remaining stale routes from that peer. The timer begins when BGP recognizes that the peer session has gone down.
- This command prevents an excessive delay in BGP reconvergence due to a peer that brings a session back up but is slow to send fresh routes.
- Specify an interval longer than the restart time.
- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**bgp graceful-restart stalepaths-time 480**

- Use the **no** version to restore the default value, 360 seconds.

clear ip bgp wait-end-of-rib

- Use to clear a peer or peer group from the set of peers for which BGP is waiting to receive an End-of-RIB marker after a peer restart.
- Alternatively, performing a hard clear of a peer without this keyword has the same effect.
- This command takes effect immediately.
- Example

  host1#**clear ip bgp 192.168.1.158 wait-end-of-rib**

- There is no **no** version.

neighbor graceful-restart

- Use to control the advertisement of the BGP graceful restart capability for specified peers or peer groups.
- Advertisement of the graceful restart capability is enabled by default.
- The **no neighbor capability negotiation** command prevents the advertisement of all BGP capabilities, including graceful restart, to the specified peers, but does not affect global advertisement of the graceful restart capability.

- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**no neighbor 10.21.3.5 graceful-restart**

- Use the **no** version to disable advertisement of the graceful restart capability. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the capability configuration.

neighbor graceful-restart restart-time

- Use to set the time BGP advertises to specified peers or peer groups within which it expects to reestablish a session after restarting. Peers flush stale routes from the speaker if the session is not restarted within this period.
- Specify an interval shorter than the stalepaths time.
- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**neighbor graceful-restart restart-time 240**

- Use the **no** version to restore the default value, 120 seconds.

neighbor graceful-restart stalepaths-time

- Use to set the maximum time BGP waits to receive an End-of-RIB marker from the specified restarted peer or peer group before flushing all remaining stale routes from that peer. The timer begins when BGP recognizes that the peer session has gone down.
- This command prevents an excessive delay in BGP reconvergence due to a peer that brings a session back up but is slow to send fresh routes.
- Specify an interval longer than the restart time.
- This command takes effect immediately and automatically bounces the session.
- Example

  host1(config-router)#**neighbor graceful-restart stalepaths-time 480**

- Use the **no** version to restore the default value, 360 seconds.

Route Refresh

If the router detects that a peer supports both Cisco-proprietary and standard route refresh messages, it will prefer to use the standard route refresh messages.

neighbor capability

- Use to control the advertisement of BGP capabilities to peers. Capability negotiation and advertisement of all capabilities are enabled by default.
- You can specify the **deprecated-dynamic-capability-negotiation**, **dynamic-capability-negotiation**, **four-octet-as-numbers**, **orf**, **route-refresh**, and **route-refresh-cisco** capabilities. The graceful restart capability is controlled by specific **graceful-restart** commands.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- You cannot configure the **receive** direction for the **orf** capability for a peer that is a member of a peer group or for a peer.
- If you issue the **route-refresh** or **route-refresh-cisco** keywords, the command takes effect immediately. If dynamic capability negotiation was negotiated for the session, a capability message is sent to inform the peer of the new capability configuration. If dynamic capability negotiation was not negotiated for the session, the session is bounced automatically.
- If you issue the **deprecated-dynamic-capability-negotiation**, **dynamic-capability-negotiation**, **four-octet-as-numbers**, **negotiation**, or **orf** keywords, the command takes effect immediately and bounces the session.
- If the BGP speaker receives a capability message for a capability that BGP did not previously advertise in the dynamic capability negotiation capability, BGP sends a notification to the peer with the error code "capability message error" and error subcode "unsupported capability code."
- IPv6 ORF prefix lists are not supported. Therefore you can specify an IPv6 address with the **orf** keyword only within the IPv4 address family and when you want to advertise IPv4 routes to IPv6 peers.
- Example

  host1(config-router)#**neighbor 10.6.2.5 capability orf prefix-list both**

- Use the **no** version to prevent advertisement of the specified capability or use the **negotiation** keyword with the **no** version to prevent all capability negotiation with the specified peer. Use the **default** version to restore the default, advertising the capability.

Interactions Between BGP and IGPs

Interactions between BGP and an interior gateway protocol are more likely to occur in an enterprise topology than in a service provider topology. You can also encounter interactions when configuring small test topologies. The main interaction factors are the following:

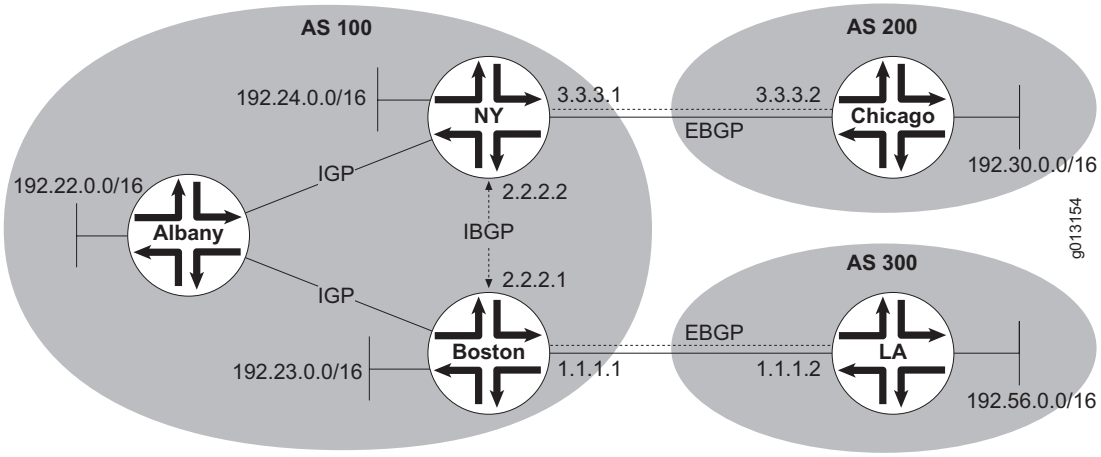
- Synchronization between BGP and IGPs
- Administrative distances for routes learned from various sources

Synchronizing BGP with IGPs

In Figure 36, AS 100 provides transit service but does not run BGP on all of the routers in the AS. In this situation, you must redistribute BGP into the IGP so that the non-BGP routers—for example, router Albany—learn how to forward traffic to customer prefixes. If BGP converges faster than the IGP, a prefix might be advertised to other ASs before that prefix can be forwarded.

For example, suppose router LA advertises a route to router Boston using EBGP, and router Boston propagates that route to router NY using IBGP. If router NY propagates the route to router Chicago before the IGP within AS 100 has converged—that is, before router Albany learns the route—then router Chicago might start sending traffic for that route before router Albany can forward that traffic.

Figure 36: Synchronization



Synchronization solves this problem by preventing a BGP speaker from advertising a route over an EBGP session until all routers within the speaker's AS have learned about the route. If the AS contains routers connected by means of an IGP, the BGP speaker cannot propagate a BGP route that it learned from a peer until an IGP route to the prefix has been installed in the BGP speaker's IP routing table. The BGP speaker advertises the BGP route externally even if the IGP route is better than the BGP route. By contrast, if synchronization is disabled, a BGP speaker propagates a BGP route learned from a peer only if it is the best route to the prefix in the IP routing table.

Synchronization is enabled by default. However, you must configure redistribution of external routes into the IGP, or the routing tables will not receive the IGP routes.

NOTE: When you create an address family for a VRF, synchronization is automatically disabled for that address family.

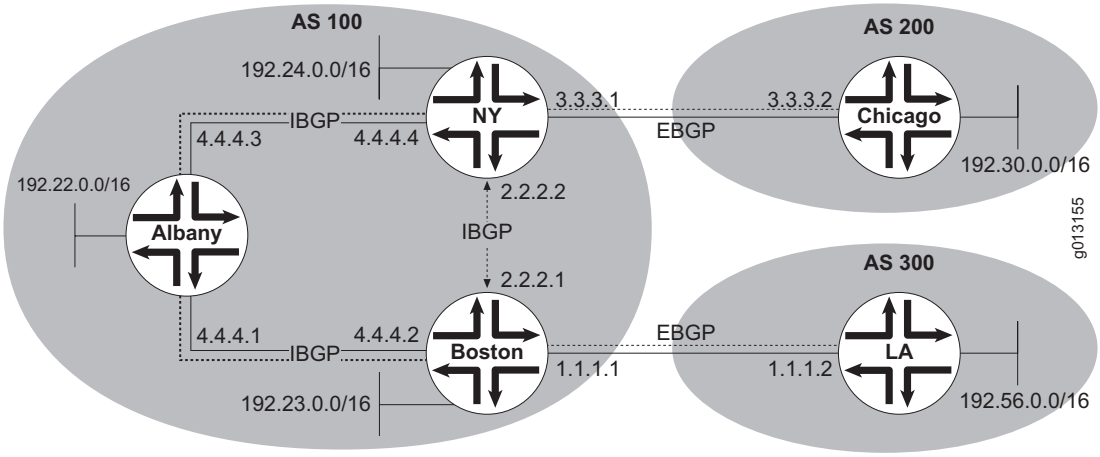
If synchronization is enabled and if redistribution is configured for the networks in Figure 36, router NY checks its IGP routing table for a route to 192.56.0.0/16 when it learns about the prefix from the IBGP session with router Boston. If the route is not present, the prefix is not reachable through router Albany, so router NY does not advertise it as available. Router NY keeps checking its IGP routing table; if the route appears, router NY knows that it can pass traffic to the prefix and advertises the route by means of EBGP to router Chicago.

In practice, service providers rarely redistribute BGP into an IGP because existing IGPs cannot handle the full Internet routing table (about 100,00 routes). Instead, all routers in an AS typically run BGP; in these cases it is advisable to turn synchronization off everywhere.

Disabling Synchronization

Because the routes learned by means of EBGP are extensive, redistributing those routes into your IGP consumes processor and memory resources. You can disable synchronization if your AS does not pass traffic from one AS to another or if all the transit routers in your AS run BGP. Figure 37 shows the same configuration as in the previous example, except that all the routers in AS 100 now run IBGP. As a result, all the routers receive updates learned by the area border routers from external BGP speakers.

Figure 37: Disabling Synchronization



If synchronization is disabled, a BGP speaker propagates a BGP route learned from a peer only if it is the best route to the prefix in the IP routing table. However, the speaker does advertise the routes that it originates.

The following commands show how to configure routers Boston, NY, and Chicago (shown in Figure 37) with synchronization disabled for routers NY and Boston. The **no synchronization** command enables router NY to put the route to 192.56.0.0/16 in its IP routing table and advertise it to router Chicago without learning about 192.56.00/16 from router Albany. The command also enables router Boston to put the route to 192.30.0.0/16 in its IP routing table and advertise it to router LA without learning about 192.30.00/16 from router Albany.

To configure router Boston:

```
host1((config)#router bgp 100
host1(config-router)#neighbor 2.2.2.2 remote-as 100
host1(config-router)#neighbor 4.4.4.1 remote-as 100
host1(config-router)#neighbor 1.1.1.2 remote-as 300
host1(config-router)#no synchronization
```

To configure router NY:

```
host2(config)#router bgp 100
host2(config-router)#neighbor 3.3.3.2 remote-as 200
host2(config-router)#neighbor 2.2.2.1 remote-as 100
host2(config-router)#neighbor 4.4.4.3 remote-as 100
host2(config-router)#no synchronization
```

To configure router Albany:

```
host3(config)#router bgp 100
host3(config-router)#neighbor 4.4.4.4 remote-as 100
host3(config-router)#neighbor 4.4.4.2 remote-as 100
host3(config-router)#no synchronization
```

To configure router Chicago:

```
host4(config)#router bgp 200
host4(config-router)#neighbor 3.3.3.1 remote-as 100
```

To configure router LA:

```
host5(config)#router bgp 300
host5(config-router)#neighbor 1.1.1.1 remote-as 100
host5(config-router)#network 192.56.0.0
```

synchronization

- Use to enable and disable synchronization between BGP and an IGP.
- Synchronization is enabled by default. However, creating an address family for a VRF automatically disables synchronization for that address family.
- This command takes effect immediately.
- Use the **no** version to advertise a route without waiting for the IGP to learn a route to the prefix.

Setting the Administrative Distance for a Route

The administrative distance is an integer in the range 0–255 that is associated with each route known to a router. The distance represents how reliable the source of the route is considered to be. A lower value is preferred over a higher value. An administrative distance of 255 indicates no confidence in the source; routes with this distance are not installed in the routing table. As shown in Table 22, default distances are provided for each type of source from which a route can be learned.

Table 22: Default Administrative Distances for Route Sources

Route Source	Default Distance
Connected interface	0
Static route	1
External BGP	20
OSPF	110
IS-IS	115
RIP	120
Internal BGP	200
Unknown	255

If the IP routing table contains several routes to the same prefix—for example, an OSPF route and an IBGP route—the route with the lowest administrative distance is used for forwarding.

By default, BGP propagates received BGP routes to EBGP routes only if the BGP route is used for forwarding traffic—that is, if it is the route with the lowest administrative distance in the IP forwarding table. However, you can modify this behavior by using the **bgp advertise-inactive** command. See *Advertising Inactive Routes* on page 60 for more information.

You can use the **distance bgp** command to configure the administrative distance associated with routes. If you choose to set an administrative distance, you must specify a value for all three of the following types of routes:

- external—Administrative distance for BGP external routes. External routes are routes for which the best path is learned from a BGP peer external to the AS. Acceptable values are from 1 to 255. The default value is 20.

- internal—Administrative distance for BGP internal routes. Internal routes are those routes that are learned from a BGP peer within the same AS. Acceptable values are from 1 to 255. The default value is 200.

- local—Administrative distance for BGP local routes. Local routes are those routes locally originated by BGP. BGP can locally originate routes if you issue the **network** command, if you configure redistribution into BGP, or by means of a non-AS-set aggregate route. Acceptable values are from 1 to 255. The default value is 200.

CAUTION: Changing the administrative distance of BGP internal routes is considered dangerous and is not recommended. One problem that can arise is the accumulation of routing table inconsistencies, which can break routing.

You can use the **distance bgp** command to configure these preferences. The following commands leave the internal distance at 200, set the external distance to 150, and set the local distance to 80:

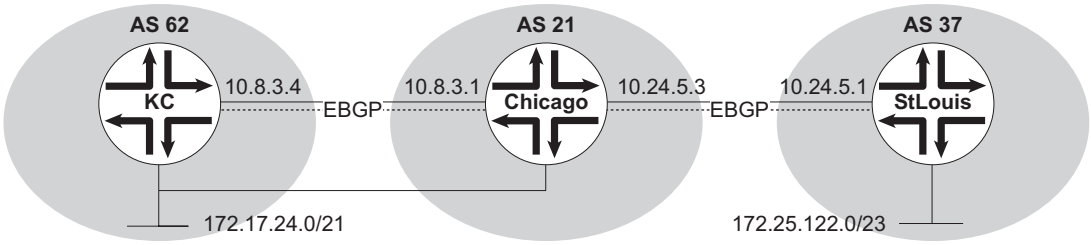
```
host1(config)#router bgp 100
host1(config-router)#network 172.28.0.0
host1(config-router)#neighbor 156.128.5.5 remote-as 310
host1(config-router)#neighbor 142.132.1.1 remote-as 50
host1(config-router)#distance bgp 150 200 80
```

distance bgp

- Use to set the administrative distance for all BGP routes.
- You must specify the following:
  - *external-distance*—Administrative distance for routes external to the AS in the range 1–255. The default is 20.
  - *internal-distance*—Administrative distance for routes internal to the AS in the range 1–255. The default is 200.
  - *local-distance*—Administrative distance for local routes in the range 1–255. The default is 200.
- The default value is 20 for external routes, 200 for internal route, and 200 for local routes.
- The new distance is applied to all routes that are subsequently placed in the IP routing table. To apply the new distance to routes that are already present in the IP routing table, you must use the **clear ip routes *** command to reinstall BGP routes in the IP routing table.
- Use the **no** version to return the distances to their default values, 20, 200, and 200.

Example 1 Routes learned from other sources can be preferred to routes learned by means of BGP. Consider the network structure shown in Figure 38.

Figure 38: Administrative Distances



Suppose router KC originates 172.17.24.0/21 and advertises the route to router Chicago by means of EBGP. Both router KC and router Chicago are directly connected to the network represented by 172.17.24.0/21. If you issue the **show ip route** command on router Chicago, the BGP route does not appear. Instead, only the connected route is displayed.

Both routes are in the IP routing table, but the **show ip route** command displays only the *best* route. (Use the **show ip route all** command to display all best routes; in this case the BGP route and the connected route.) Connected routes have a default distance of 0. Routes learned by means of EBGP have a default value of 20. The connected route is a better route than the EBGP route and appears in the command display.

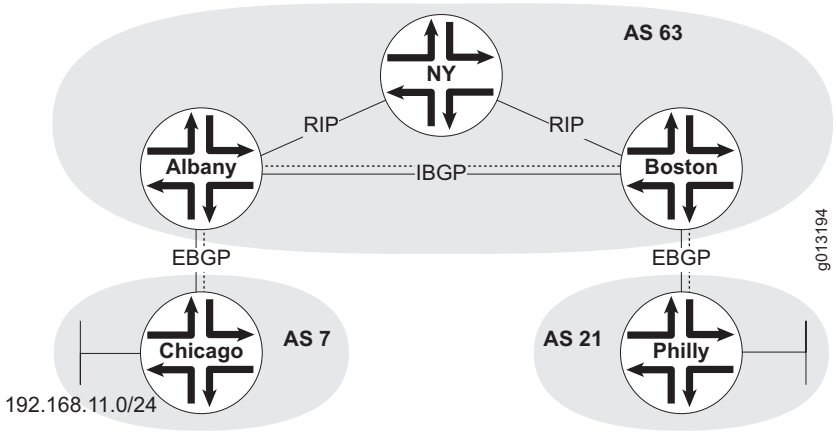
In practice, if two BGP peers are connected to the same network, both peers should originate the route.

Example 2 Consider the network structure shown in Figure 39. Router Chicago originates prefix 192.168.11.0/24 and advertises it by means of EBGP to router Albany. Router Albany advertises the route to router Boston by means of IBGP.

Router Albany also redistributes the route into the interior gateway protocol RIP, which informs router NY of the route. Router NY propagates the route to router Boston by means of RIP, from which it is injected into BGP.

In this example, both router Albany and router Boston have synchronization turned on. When synchronization is on, BGP propagates a received route to EBGP peers, even if the IP forwarding table contains a non-BGP route with a better administrative distance than the BGP route. This example demonstrates why synchronization is needed.

Figure 39: Administrative Distance and Synchronization



Router Boston does *not* advertise the route externally to router Philly. At first, this is because router Boston has not yet heard about the prefix from router NY, and therefore the IGP route does not appear in router Boston's IP routing table.

BGP routes are not propagated until a route to the prefix by means of any IGP appears in the IP routing table. In other words, routers connected by means of an IGP must have a route to the prefix before a BGP speaker can advertise the route it learned from a peer.

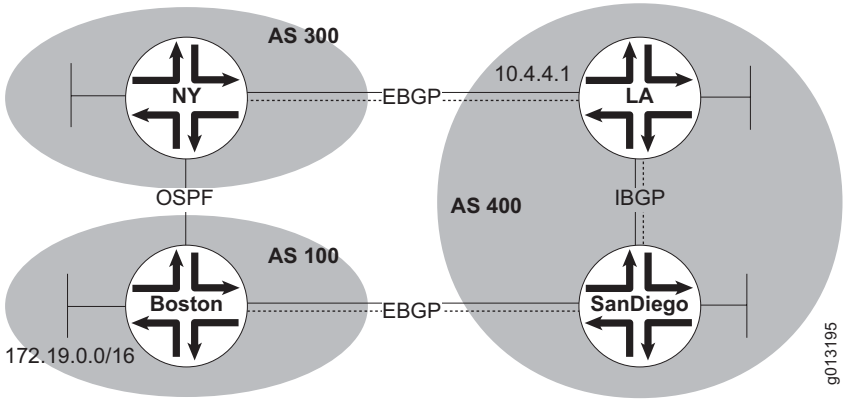
When the RIP route appears on router Boston, the router has both an IBGP route and a RIP route to the same prefix. Even though the RIP route has a better administrative distance, the IBGP route is propagated to router Philly because synchronization is turned on.

Configuring Backdoor Routes

In certain network topologies, a BGP speaker might learn routes to the same prefix from an external BGP peer and by means of an IGP protocol. Consider the network structure shown in Figure 40.

A company has established an OSPF link between routers NY and Boston. This private link between the two routers is known as a *backdoor* link. Router NY learns two routes to prefix 172.19.0.0/16; one by means of OSPF from router Boston, and one by means of EBGP from router LA through router SanDiego. As was shown in Table 22, EBGP routes have an administrative distance of 20 and are preferred over IGP routes, which have much higher administrative distances. In this example, the longer path by means of EBGP is preferred over the OSPF backdoor path with its distance of 110.

Figure 40: Backdoor Route



You can modify this behavior by issuing the **network backdoor** command on router NY:

```
host1(config)#router bgp 300
host1(config-router)#neighbor 10.4.4.1 remote-as 400
host1(config-router)#network 172.19.0.0 backdoor
```

Unlike the typical **network** command, **network backdoor** does not cause the BGP speaker to advertise the specified prefix. Instead, it sets the administrative distance for the EBGP path to that prefix to the same value as a route learned by means of IBGP. That is, the EBGP administrative distance is changed from the highly preferred value of 20 to the highly unpreferred value of 200. In Figure 40, this change in value results in the backdoor OSPF being more preferred as a way to reach prefix 172.19.0.0/16.

network backdoor

- Use to cause a backdoor IGP route to be preferred over an EBGP route to the same prefix by setting the administrative distance of the EBGP route to that of an IBGP route, 200.
- Issuing this command does not cause the BGP speaker to advertise the specified route.
- This command takes effect immediately.
- Example

  host1(config-router)#**network 10.53.42.0 backdoor**

- Use the **no** version to restore the default distance to the EBGP route, 20.

Setting the Maximum Number of Equal-Cost Multipaths

You can use the **maximum-paths** command to specify the number of equal-cost paths to the same destination that BGP can submit to the IP routing table.

If you set the value to 1, the router installs the single best route in the IP routing table. If you set the value greater than 1, the router installs that number of parallel routes.

maximum-paths

- Use to set the maximum number of equal-cost multipaths.
- Specify a value in the range 1–16; the default value is 1.
- If you do not specify a keyword, the maximum number applies only to routes learned from external peers. If you specify the **ibgp** keyword, the maximum number applies only to routes received from internal peers. If you specify the **eibgp** keyword (valid only for VRF IPv4 unicast or IPv6 unicast address families), the maximum number applies to routes received from internal peers and external peers.
- This command takes effect immediately; it does not bounce the session.
- You can specify the maximum number of equal-cost multipaths in the context of the virtual router, an IPv4 unicast or IPv6 unicast address family, or a VRF specified in the context of an IPv4 unicast or IPv6 unicast address family.
- This command is not supported for VPNv4 or VPNv6 address families.
- Example 1

  host1(config-router)#**maximum-paths 3**

- Example 2

  host1:vr1(config-router-af)#**maximum-paths ibgp 5**

- Use the **no** version to restore the default value, 1.

Detecting Peer Reachability with BFD

You can configure a Bidirectional Forwarding Detection (BFD) session with a BGP neighbor or peer group to determine relatively quickly whether the neighbor or peer group is reachable. For more information on BFD, see *JUNOSe IP Services Configuration Guide, Chapter 5, Configuring BFD*. BFD is supported only for single-hop IBGP and EBGP sessions with either IPv4 or IPv6 neighbors in the core or within a VRF. BFD is not supported for multi-hop BGP sessions (IBGP multi-hop or EBGP multi-hop). BFD behavior is identical for IBGP and EBGP single-hop sessions, and for IPv4 and IPv6 neighbors.

When you configure BFD for a BGP session, the normal BGP keepalive mechanism is not disabled. Unless you configure BGP not to do so, BGP still sends keepalive messages and brings the BGP session down if the holdtimer expires.

When you configure this feature, BGP requests BFD to start a BFD protocol session as soon as the BGP session enters the established state. BGP allows the BFD protocol session to come up only when the source address of received BFD packets matches the destination address of the BGP neighbor. When the BFD protocol session comes up, BGP logs this event and reports the session in subsequent **show** commands. If the BFD protocol session goes down, BGP immediately brings down the BGP session and takes all associated actions.

Whenever a BGP session leaves the established state, BGP requests BFD to stop the BFD protocol session. BGP also requests BFD to bring the BFD protocol session down and inform BGP if the local interface goes down.

To enable a BGP session to come up even if the remote peer does not support BFD or has not been configured to use BFD, the following behavior applies:

- The BGP session can come up when the BFD protocol session is not yet up.
- The BGP session can stay up even when the BFD protocol session never comes up.

You can specify a desired rate for receiving BFD packets from the peer, transmitting them to the peer, or both, by setting a desired time interval between the packets. The actual timer values can be different as a result of other applications requesting BFD protocol sessions on the same interface with different timer values, as a result of timer value negotiation between the local and remote BFD speakers, or both.

In the following example, the router is configured to send BFD packets to peer 10.25.43.1 with a minimum interval of 450 milliseconds between the packets, and to accept BFD packets from the peer only with the same minimum interval:

```
host1(config)#router bgp 100
host1(config-router)#neighbor 10.25.43.1 bfd-liveness-detection
minimum-interval 450
```

neighbor bfd-liveness-detection

- Use to enable BGP to detect whether a neighbor is unreachable by means of a BFD protocol session to the neighbor.
- The peers in a BGP adjacency use the configured values to negotiate the actual transmit intervals for BFD packets.
  - You can use the **minimum-transmit-interval** keyword to specify the interval at which the local peer proposes to transmit BFD control packets to the remote peer. The default value is 300 milliseconds.
  - You can use the **minimum-receive-interval** keyword to specify the minimum interval at which the local peer must receive BFD control packets from the remote peer. The default value is 300 milliseconds.
  - You can use the **minimum-interval** keyword to specify the same value for both of those intervals. Configuring a minimum interval has the same effect as configuring the minimum receive interval and the minimum transmit interval to the same value. The default value is 300 milliseconds.
- You can use the **multiplier** keyword to specify the detection multiplier value. The calculated BFD liveness detection interval can be different on each peer. The multiplier value is roughly equivalent to the number of packets that can be missed before the BFD session is declared to be down. The default value is 3.
- For details on liveness detection negotiation, see *Negotiation of the BFD Liveness Detection Interval* section in *JUNOSe IP Services Configuration Guide, Chapter 5, Configuring BFD*.
- You can change the BFD liveness detection parameters at any time without stopping or restarting the existing session; BFD automatically adjusts to the new parameter value. However, no changes to BFD parameters take place until the values resynchronize with each peer.

- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- This command takes effect immediately.
- The BGP session does not flap when you enable BFD for a session that is already up or change the BFD timer values for an established session.
- If you remove the BFD configuration while the BGP sessions and the BFD protocol session are up, then the BGP session may flap because the remote BGP speaker cannot detect why the BFD session went down.
- Use the **no** version to disable BFD liveness detection for the neighbor. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

BFD and BGP Graceful Restart

So that BFD can maintain its BFD protocol sessions across a BGP graceful restart, BGP requests that BFD set the C bit to 1 in transmitted BFD packets. When the C bit is set to 1, BFD can maintain its session in the forwarding plane in spite of disruptions in the control plane. Setting the bit to 1 gives BGP neighbors acting as a graceful restart helper the most accurate information about whether the forwarding plane is up.

When BGP is acting as a graceful restart helper and the BFD session to the BGP peer is lost, one of the following actions takes place:

- If the C bit received in the BFD packets was 1, BGP immediately flushes all routes, determining that the forwarding plane on the BGP peer has gone down.
- If the C bit received in the BFD packets was 0, BGP marks all routes as stale but does not flush them because the forwarding plane on the BGP peer might be working and only the control plane has gone down.

Managing a Large-Scale AS

BGP requires that IBGP peers be fully meshed, creating significant routing overhead as the number of peers increases. The number of IBGP sessions increases rapidly with the number of routers:

$$\text{IBGP sessions} = \frac{(\text{number of BGP peers in the AS})^2 - (\text{number of BGP peers in the AS})}{2}$$

For example, in an AS with 9 BGP peers, the peers can conduct 36 sessions:

$$\text{IBGP sessions} = \frac{9^2 - 9}{2} = 36$$

BGP provides the following two alternative configuration strategies to reduce the number of fully meshed peers:

- Configure confederations.
- Configure route reflectors.

Both of these strategies are complex and can create their own problems. Neither strategy is typically used unless the mesh of IBGP peers approaches 100 sessions per peer.

Configuring a Confederation

IBGP requires that BGP speakers within an AS be fully meshed. You can reduce the IBGP mesh inside an AS by subdividing the AS into a confederation of sub-ASs. Each sub-AS must be fully meshed internally, but the sub-ASs do not have to be fully meshed with each other. Confederations are most useful when the number of IBGP speakers within an AS increases to the point that each router has about 100 peering sessions.

Figure 41 shows a simpler topology. AS 29 consists of 10 fully meshed IBGP peers (for clarity, only the BGP sessions are shown). Border router Salem has an EBGP session with a neighbor in AS 325. Border router Boston has an EBGP session with a neighbor in AS 413.

Figure 41: A Fully Meshed Autonomous System

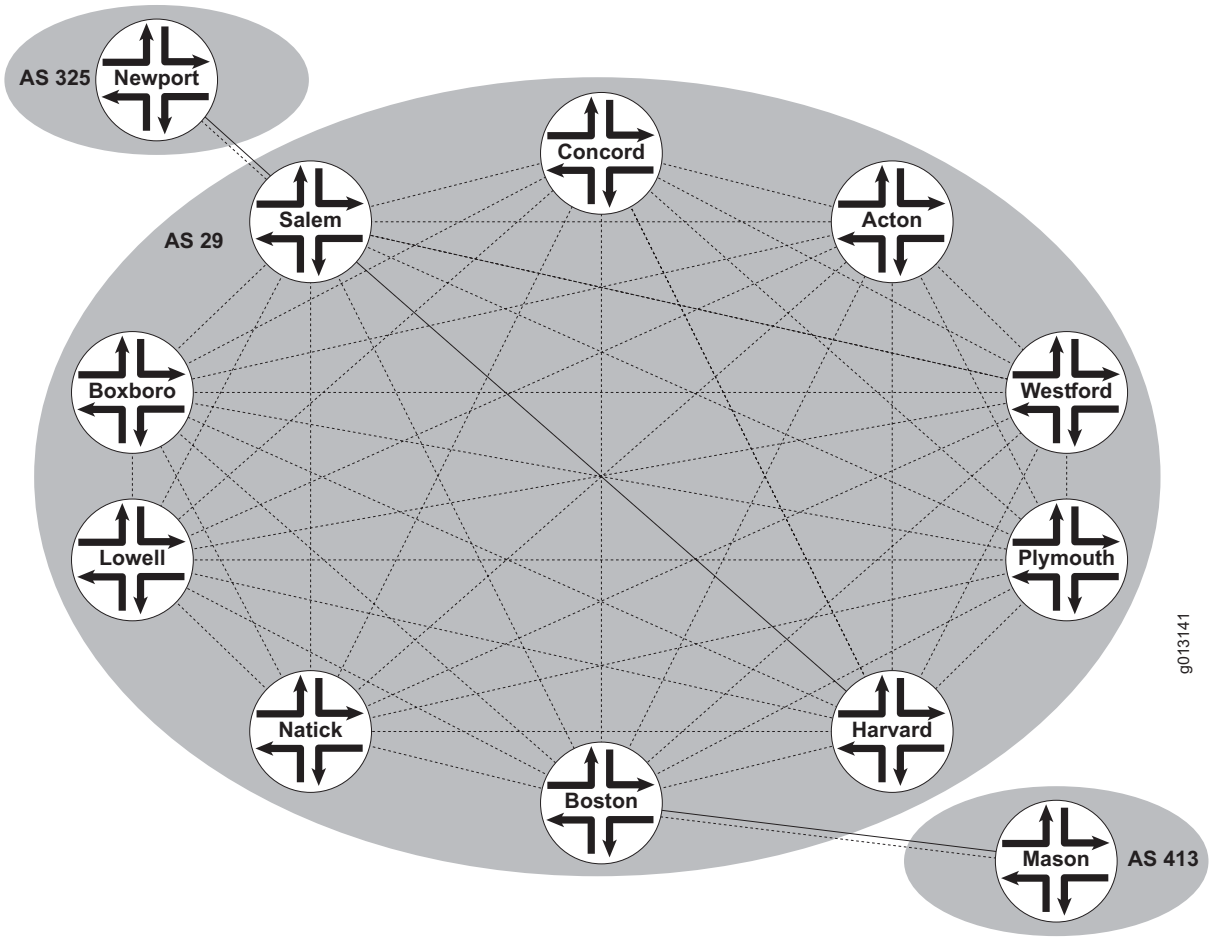
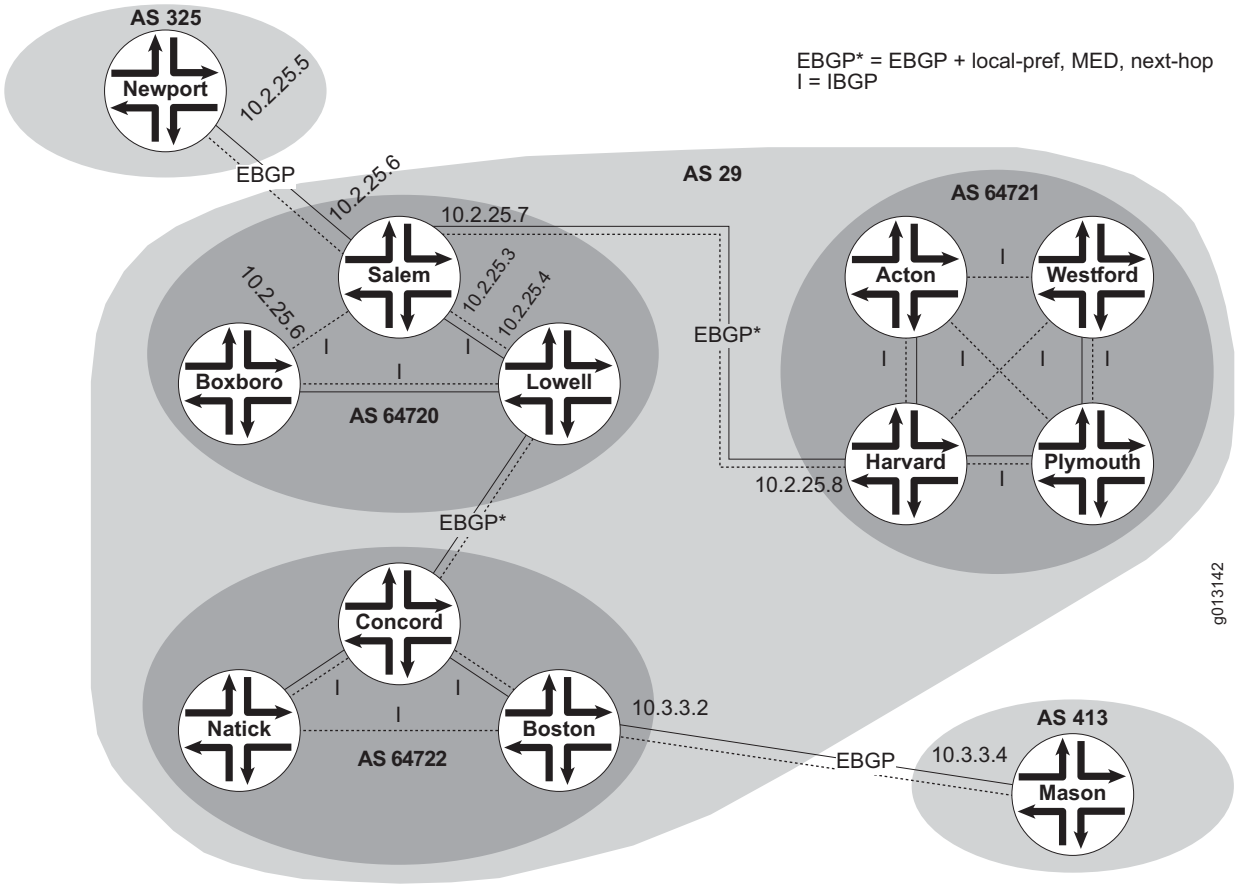


Figure 42 illustrates how you can create three sub-ASs within AS 29 to greatly reduce the number of peering sessions. According to common practice, use a number from the private range of AS numbers—from 64512 to 65535—to identify each sub-AS. AS 29 is now a confederation of three sub-ASs: AS 64720, AS 64721, and AS 64722. Each sub-AS consists of fully meshed IBGP peers. A slightly modified version of EBGP runs between the sub-ASs: It acts like IBGP within an AS because the local-pref, MED, and next-hop attributes are preserved across the sub-AS boundaries. To the external neighbors, AS 29 appears the same as it ever was.

Figure 42: A Confederation of Subautonomous Systems



The following commands partially configure router Salem:

```
host1(config)#router bgp 64720
host1(config-router)#bgp confederation identifier 29
host1(config-router)#bgp confederation peers 64721 64722
host1(config-router)#neighbor 10.2.25.4 remote-as 64720
host1(config-router)#neighbor 10.2.25.8 remote-as 64721
host1(config-router)#neighbor 10.2.25.2 remote-as 325
```

The **bgp confederation identifier** command establishes router Salem as a member of Confederation 29. The **bgp confederation peers** command specifies that sub-AS 64721 and sub-AS 64722 are members of the same confederation as the sub-AS that includes router Salem. The **neighbor remote-as** commands specify the IBGP connection with a neighbor in sub-AS 64720 and the EBGP connections with neighbors in sub-AS 64721 and outside the confederation in AS 325.

Similarly, the following commands partially configure router Harvard:

```
host2(config)#router bgp 64721
host2(config-router)#bgp confederation identifier 29
host2(config-router)#bgp confederation peers 64720 64722
host2(config-router)#neighbor 10.2.25.7 remote-as 64720
```

From router Newport's perspective, router Salem is simply a member of AS 29:

```
host3(config)#router bgp 325
host3(config-router)#neighbor 10.2.25.6 remote-as 29
```

From router Mason's perspective, router Boston is simply a member of AS 29:

```
host4(config)#router bgp 413
host4(config-router)#neighbor 10.3.3.2 remote-as 29
```

bgp confederation identifier

- Use to establish a router as a member of the specified BGP confederation.
- To routers outside the confederation, the confederation appears as an autonomous system with an AS number the same as the confederation identifier.
- The new confederation identifier is used in open messages and in the AS path in update messages that are sent after you issue the command.

  To force sessions that are already up to use the new confederation identifier, you must use the **clear ip bgp** command to perform a hard clear.

- Use the **no** version to remove the sub-AS from the confederation.

bgp confederation peers

- Use to enable EBGP sessions with routers in the peer sub-ASs; the EBGP sessions preserve local-pref, MED, and next-hop attributes.
- You can specify one or more individual sub-AS numbers, or you can issue the **filter-list** keyword and an AS-path access list (which is based on regular expressions) to specify a list of sub-AS numbers.
- If the remote AS of a peer appears in the specified list of sub-ASs or is identified by the filter list, then the peer is treated as being in the same confederation.
- This command takes effect immediately and bounces only those sessions whose peer type changed as a result of issuing the command.
- Use the **no** version to remove individually specified sub-ASs, all sub-ASs specified by the filter list, or all sub-ASs from the confederation.

ip bgp-confed-as-set new-format

- Use to specify that AS-confed-sets are displayed enclosed within square brackets rather than parentheses, and that the AS paths in the set are delimited by commas rather than spaces.
- Example

  host1(config)#**ip bgp-confed-as-set new-format**

- Use the **no** version to restore the default display within parentheses and with space-delimited ASs.

Configuring Route Reflectors

Router reflection is an alternative to confederations as a strategy to reduce IBGP meshing. BGP specifies that a BGP speaker cannot advertise routes to an IBGP neighbor if the speaker learned the route from a different IBGP neighbor. A *route reflector* is a BGP speaker that advertises routes learned from each of its IBGP neighbors to its other IBGP neighbors; routes are reflected among IBGP routers that are not meshed. The route reflector's neighbors are called *route reflector clients*. The clients are neighbors only to the route reflector, not to each other. Each route reflector client depends on the route reflector to advertise its routes within the AS; each client also depends on the route reflector to pass routes to the client.

A route reflector and its clients are collectively referred to as a *cluster*. Clients peer only with a route reflector and do not peer outside their cluster. Route reflectors peer with clients and other route reflectors within the cluster; outside the cluster they peer with other reflectors and other routers that are neither clients nor reflectors. Route reflectors and nonclient routers must be fully meshed.

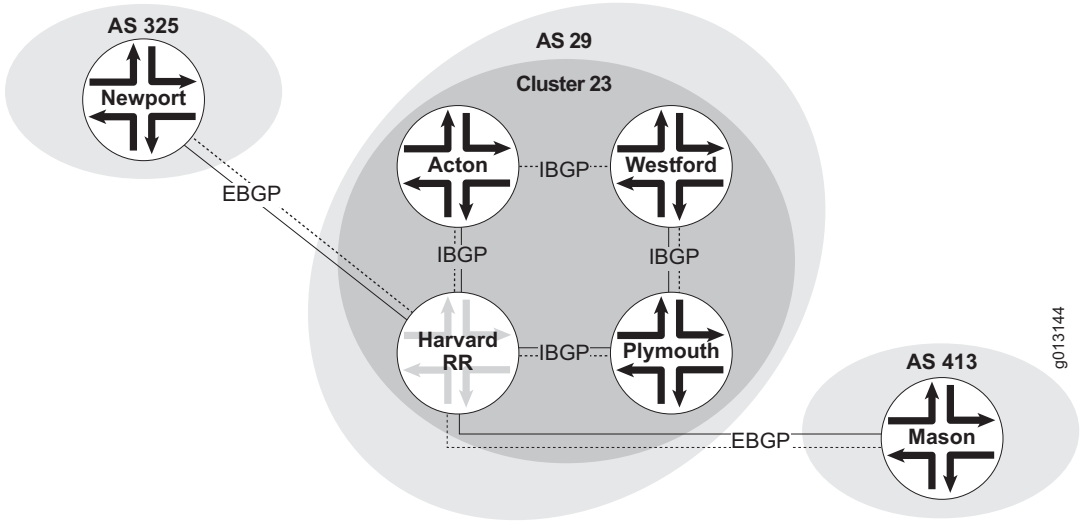
Clients and nonclients have no knowledge of route reflection; they operate as standard BGP peers and require no configuration. You simply configure the route reflectors.

Route reflectors advertise routes learned from:

- A nonclient peer only to clients
- A client peer to all nonclient peers and to all client peers except for the originator of the route
- An EBGP peer to all nonclient peers and all client peers

Figure 43 illustrates a simple route reflection setup. Configured as a route reflector, Router Harvard reflects routes among its clients within Cluster 23: Routers Plymouth, Westford, and Acton. These route reflector clients see router Harvard and each other simply as IBGP neighbors. Router Newport in AS 325 and router Mason in AS 413 see router Harvard simply as an EBGP neighbor in AS 29.

Figure 43: Simple Route Reflection

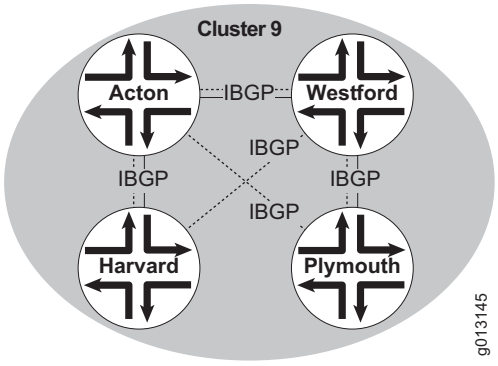


Route Reflection and Redundancy

Reliability and redundancy are important issues when using route reflection because the members of a cluster are not fully meshed. For example, if router Harvard in Figure 43 goes down, all of its clients are isolated from networks outside the cluster. Having one or more redundant route reflectors in a cluster protects against such an occurrence.

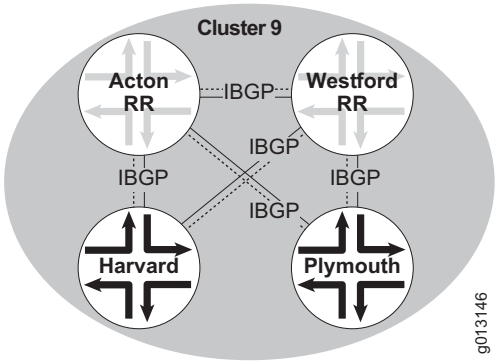
However, you cannot rely on logical redundancy alone. Consider the cluster shown in Figure 44. The operator has attempted to provide redundancy in Cluster 9 by configuring two route reflectors, router Acton and router Westford. Unfortunately, router Harvard is physically isolated if its link to router Acton goes down, or if router Acton itself goes down. Similarly, router Plymouth is isolated if any problems develop with router Westford.

Figure 44: Route Reflection: Logical Redundancy



In Figure 45, the operator has added physical redundancy to the cluster configuration. Now, loss of either one of the route reflectors does not isolate the reflector clients.

Figure 45: Route Reflection: Physical and Logical Redundancy



Route Reflection and Looping

BGP prevents looping *between* ASs by evaluating the AS-path attribute to determine a route's origin. Border routers reject routes they receive from external neighbors if the AS path indicates that the route originated within the border router's AS.

Route reflection creates the possibility of looping *within* an AS. Routes that originate within a cluster might be forwarded back to the cluster. Because this happens within a given AS, the AS-path attribute is of no use in detecting a loop.

Route reflectors add an *originator ID* to each route that identifies the originator of the route within the local AS by its router ID. If a router receives a route having the originator ID set to its own router ID, it rejects the route.

You can also use a *cluster list* to prevent looping. Each cluster has an identifying number, the cluster ID. For clusters with a single route reflector, the cluster ID is the router ID of the route reflector; otherwise you configure the cluster ID. The cluster list records the cluster ID of each cluster traversed by a route. When a route reflector passes a route from a client to a nonclient router outside the cluster, the reflector appends the cluster ID to the list. When a route reflector receives a route from a nonclient, it rejects the route if the list contains the local cluster ID.

What about routes that a client forwards out of the cluster? No cluster ID is needed, because clients can forward routes only to EBGP peers, that is, to peers outside the AS. Looping between ASs is prevented by the AS-path list.

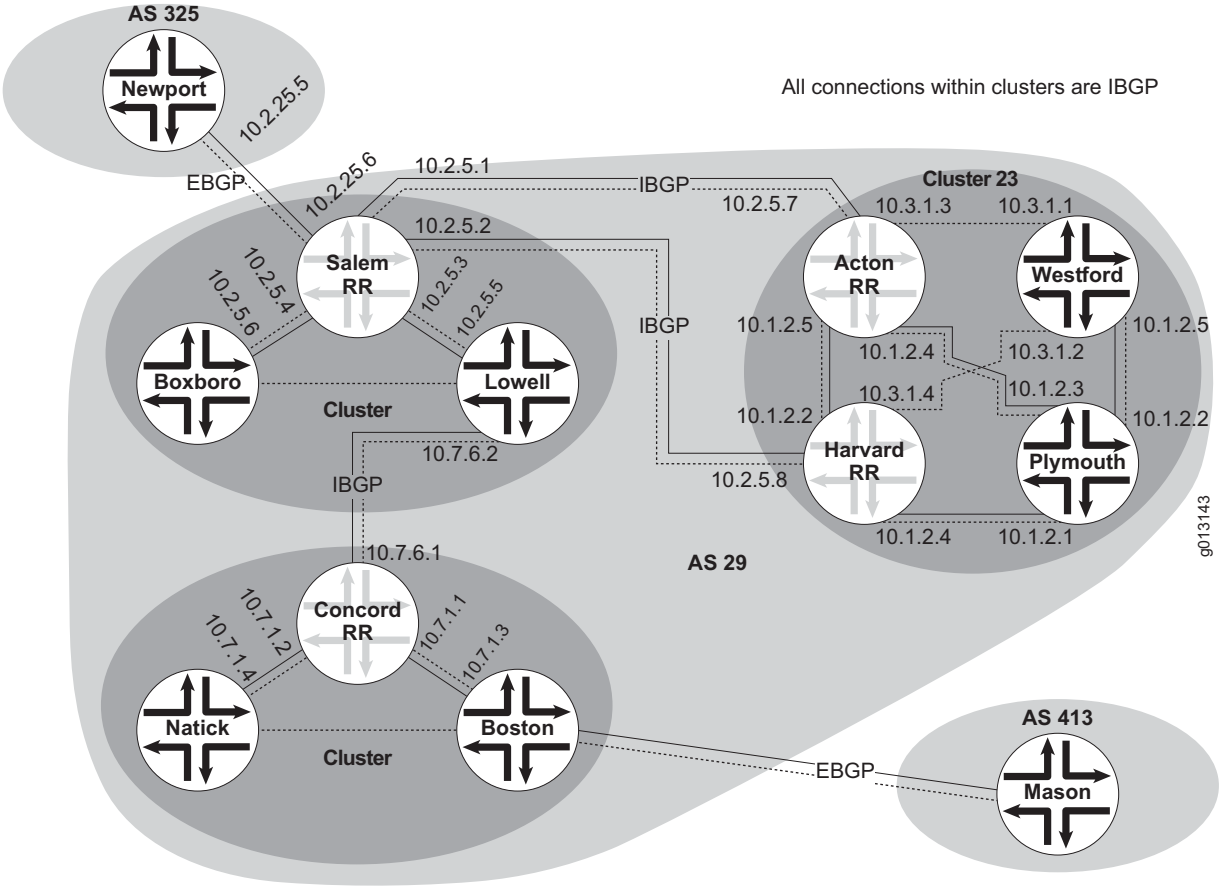
The following commands configure the route reflectors for the network topology shown in Figure 46. You configure the other routers, whether nonclients or route reflector clients, as usual for IBGP and EBGP peers.

To configure router Salem as a route reflector:

```
host1(config)#router bgp 29
host1(config-router)#neighbor 10.2.5.5 remote-as 29
host1(config-router)#neighbor 10.2.5.5 route-reflector-client
host1(config-router)#neighbor 10.2.5.6 remote-as 29
host1(config-router)#neighbor 10.2.5.6 route-reflector-client
host1(config-router)#neighbor 10.2.5.7 remote-as 29
host1(config-router)#neighbor 10.2.5.8 remote-as 29
host1(config-router)#neighbor 10.2.25.5 remote-as 325
```

You do not configure a cluster ID, because router Salem is the only route reflector in this cluster.

Figure 46: BGP Route Reflection



To configure router Concord as a route reflector:

```
host2(config)#router bgp 29
host2(config-router)#neighbor 10.7.1.3 remote-as 29
host2(config-router)#neighbor 10.7.1.3 route-reflector-client
host2(config-router)#neighbor 10.7.1.4 remote-as 29
host2(config-router)#neighbor 10.7.1.4 route-reflector-client
host2(config-router)#neighbor 10.7.6.2 remote-as 29
```

You do not configure a cluster ID, because router Concord is the only route reflector in this cluster.

To configure router Acton as a route reflector:

```
host3(config)#router bgp 29
host3(config)#bgp cluster-id 23
host3(config-router)#neighbor 10.3.1.1 remote-as 29
host3(config-router)#neighbor 10.3.1.1 route-reflector-client
host3(config-router)#neighbor 10.1.2.3 remote-as 29
host3(config-router)#neighbor 10.1.2.3 route-reflector-client
host3(config-router)#neighbor 10.3.3.4 remote-as 29
host3(config-router)#neighbor 10.2.5.1 remote-as 29
```

You must configure a cluster ID, because router Acton and router Harvard are both route reflectors in this cluster.

To configure router Harvard as a route reflector:

```
host4(config)#router bgp 29
host4(config)#bgp cluster-id 23
host4(config-router)#neighbor 10.3.1.2 remote-as 29
host4(config-router)#neighbor 10.3.1.2 route-reflector-client
host4(config-router)#neighbor 10.1.2.1 remote-as 29
host4(config-router)#neighbor 10.1.2.1 route-reflector-client
host4(config-router)#neighbor 10.3.3.2 remote-as 29
host4(config-router)#neighbor 10.2.5.2 remote-as 29
```

You must configure a cluster ID, because router Harvard and router Acton are both route reflectors in this cluster.

bgp client-to-client reflection

- Use to reenable the reflector to reflect routes among all clients.
- Client-to-client reflection is enabled by default. If the route reflector's clients are fully meshed, you can disable reflection because it is not necessary.
- If client-to-client reflection is enabled (the default), clients of a route reflector cannot be members of a peer group.
- Example

  ```
  host1(config-router)#no bgp client-to-client reflection
  ```

- Changes apply automatically to any routes received after you issue the command. To advertise or withdraw routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to issue a hard clear or an outbound soft clear.
- Use the **no** version to disable route reflection; use only if the route reflector's clients are fully meshed.

bgp cluster-id

- Use to configure a cluster ID on the route reflectors if the BGP cluster has more than one route reflector. For clusters with a single reflector, the cluster ID is the reflector's router ID and does not have to be configured.
- You specify a cluster ID number or an IP address of a router acting as a route reflector.
- The new cluster ID is used in update messages sent after you issue the command. To force BGP to resend all routes with the new cluster ID, you must use the **clear ip bgp** command to perform a hard clear or a soft clear.
- Use the **no** version to cause BGP to use the router ID as the cluster ID.

neighbor route-reflector-client

- Use to configure the local router as the route reflector and the specified neighbor as one of its clients. The reflector and its clients constitute a cluster. BGP neighbors that are not specified as clients are nonclients.
- Route reflectors pass routes among the client routers.
- Route reflection eliminates the need for all IBPG peers to be fully meshed. The members of a cluster do not have to be fully meshed, but BGP speakers outside the cluster must be fully meshed.
- If client-to-client reflection is enabled (the default), clients of a route reflector cannot be members of a peer group.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command. You cannot override this inheritance for a peer group member.
- Changes apply automatically to any routes received after you issue the command. To advertise or withdraw routes that are already present in the BGP routing table, you must use the **clear ip bgp** command to issue a hard clear or an outbound soft clear.
- Use the **no** version to indicate that the neighbor is no longer a client. Use the **default** version to remove the explicit configuration from the peer or peer group and reestablish inheritance of the feature configuration.

Configuring BGP Multicasting

The BGP multiprotocol extensions (MP-BGP) enable BGP to carry IP multicast routes used by the Protocol Independent Multicast (PIM) to build data distribution trees. (See *JUNOSe Multicast Routing Configuration Guide, Chapter 5, Configuring IPv4 Multicast* for information about PIM.) You can configure a multicast routing topology different from your unicast topology to achieve greater control over network resources. This application of MP-BGP is often referred to as multicast BGP (MBGP).

The BGP multiprotocol extensions specify that BGP can exchange information within different types of *address families*:

- Unicast IPv4—If you do not explicitly specify the address family, the router is configured to exchange unicast IPv4 addresses by default.
- Multicast IPv4—If you specify the multicast IPv4 address family, you can use BGP to exchange routing information about how to reach a multicast source instead of a unicast destination. For information about BGP multicasting commands, see *Chapter 1, Configuring BGP Routing*. For a general description of multicasting, see *JUNOSe Multicast Routing Configuration Guide, Chapter 5, Configuring IPv4 Multicast*.
- VPN IPv4—If you specify the VPN-IPv4 (also known as VPNv4) address family, you can configure the router to provide IPv4 VPN services over an MPLS backbone. These VPNs are often referred to as BGP/MPLS VPNs.
- Unicast IPv6—If you specify the IPv6 unicast address family, you can configure the router to exchange unicast IPv6 routes. For a description of IPv6, see *JUNOSe IP, IPv6, and IGP Configuration Guide, Chapter 2, Configuring IPv6*.
- Multicast IPv6—If you specify the multicast IPv6 address family, you can use BGP to exchange routing information about how to reach an IPv6 multicast source instead of an IPv6 unicast destination. For a general description of multicasting, see *JUNOSe Multicast Routing Configuration Guide, Chapter 5, Configuring IPv4 Multicast*.
- VPN IPv6—If you specify the VPN-IPv6 address family, you can configure the router to provide IPv6 VPN services over an MPLS backbone. These VPNs are often referred to as BGP/MPLS VPNs.
- L2VPN—If you specify the L2VPN address family, you can configure the router to exchange layer 2 network layer reachability information (NLRI) for all Virtual Private LAN Service (VPLS) instances. Optionally, you can use the **signaling** keyword for the L2VPN address family to specify BGP signaling of L2VPN reachability information. Currently, you can omit the **signaling** keyword with no adverse effects. For a description of VPLS, see *Chapter 8, Configuring VPLS*.
- VPLS—If you specify the VPLS address family, you can configure the router to exchange layer 2 NLRI for the VPLS address family for a specified VPLS instance. For a description of VPLS, see *Chapter 8, Configuring VPLS*.
- VPWS—If you specify the VPWS address family, you can configure the PE router to exchange layer 2 NLRI for a specified L2VPN (VPWS) instance. For a description of L2VPNs (VPWS), see *Chapter 11, Configuring L2VPNs*.

As discussed in *Understanding BGP Command Scope* on page 17, BGP configuration commands fall into five categories. If you specify the multicast address family, from within the Address Family Configuration mode you can issue the commands listed in Table 8 on page 18 to configure parameters that affect the multicast address family globally. You can issue the commands listed in Table 10 on page 19 to configure a peer or peer group that you have activated in the multicast address family without affecting those configuration parameters for any other address family within which the peer or peer group is activated.

If you issue any of the commands listed in Table 9 on page 19 from within the default IPv4 unicast address family to configure a peer or peer group, you can apply those configuration values to the same entity in the multicast address family by activating the peer or peer group in the multicast address family.

Example To add a peer to the multicast routing table, first add the peer to the unicast routing table, and then copy it to the multicast routing table.

```
host1(config)#router bgp 22
host1(config-router)#neighbor 192.168.55.122 remote-as 33
host1(config-router)#address-family ipv4 multicast
host1(config-router-af)neighbor 192.168.55.122 activate
```

address-family

- Use to configure the router to exchange IPv4 or IPv6 addresses by creating the specified address family.
- IPv4 addresses can be exchanged in unicast, multicast, or VPN mode. IPv6 addresses can be exchanged in unicast mode.
- The default setting is to exchange IPv4 addresses in unicast mode from the default router.
- This command takes effect immediately.
- Examples

  ```
  host1:vr1(config-router)#address-family ipv4 multicast
  host1:vr1(config-router)#address-family vpnv4
  host1:vr1(config-router)#address-family ipv4 unicast vrf vr2
  ```

- Use the **no** version to disable the exchange of a type of prefix.

exit-address-family

- Use to exit Address Family Configuration mode and access Router Configuration mode.
- Example

  ```
  host1:vr1(config-router-af)#exit-address-family
  ```

- There is no **no** version.

neighbor activate

- Use to specify a peer with which routes of the current address family are exchanged.
- A peer can be activated in more than one address family. By default, a peer is activated only for the IPv4 unicast address family.
- The peer must be created in unicast IPv4 or VPN IPv4 before you can activate it in another address family.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.

- The address families that are actively exchanged over a BGP session are negotiated when the session is established.
- This command takes effect immediately. If dynamic capability negotiation was not negotiated with the peer, the session is automatically bounced so that the exchanged address families can be renegotiated in the open messages when the session comes back up.

  If dynamic capability negotiation was negotiated with the peer, BGP sends a capability message to the peer to advertise or withdraw the multiprotocol capability for the address family in which this command is issued. If a neighbor is activated, BGP also sends the full contents of the BGP routing table of the newly activated address family.

- Example

  host1:vr1(config-router-af)#**neighbor 192.168.1.158 activate**

- Use the **no** version to indicate that routes of the current address family must not be exchanged with the peer.

Monitoring BGP Multicast Services

To display values from the BGP multicast routing table, use the show BGP commands with the **ipv4 multicast** keyword. For more information about displaying BGP parameters, see *Monitoring BGP* on page 155.

Using BGP Routes for Other Protocols

You can use the **ip route-type** or **ipv6 route-type** command to specify whether BGP IPv4 or IPv6 unicast routes are available only for unicast routing protocols or for both unicast and multicast routing protocols to perform RPF checks. Routes available for unicast routing protocols appear in the unicast view of the routing table, whereas routes available for multicast routing protocols appear in the multicast view of the routing table.

Typically you use MP-BGP to learn the RPF routes for multicast protocols, especially if the topology for multicast networks differs from that for unicast networks. However, you might use this command if you do not want to run multicast MP-BGP, or if you are running BGP between CE routers in a given BGP/MPLS VPN (the current specification does not provide a way to transmit multicast MP-BGP routes across a BGP/MPLS VPN core).

ip route-type
ipv6 route-type

- Use to specify whether BGP routes are available for other unicast protocols or both unicast and multicast protocols.
- You cannot specify that BGP routes are available only for multicast protocols.
- Use the **show ip route** or **show ipv6 route** command to view the routes available for unicast protocols.
- Use the **show ip rpf-route** or **show ipv6 rpf-route** command to view the routes available for multicast protocols. It does not display routes available only to unicast protocols.

- By default, BGP IPv4 and IPv6 unicast routes are available only for other unicast routing protocols.
- Example 1

  host1(config)#**router bgp 100**
  host1(config-router)#**ipv6 route-type both**

- Example 2

  host1(config)#**router bgp 100**
  host1(config-router)#**address-family ipv4 unicast vrf v1**
  host1(config-router-af)#**ip route-type both**

- Use the **no** version to restore the default value, unicast.

Configuring BGP/MPLS VPNs

The BGP multiprotocol extensions enable the exchange of BGP information within different types of address families. The VPN IPv4 address family enables you to configure the router to provide IPv4 VPN services over an MPLS backbone. These VPNs are often referred to as BGP/MPLS VPNs. For detailed information, see *Chapter 3, Configuring BGP-MPLS Applications*.

Testing BGP Policies

You can analyze and check your BGP routing policies on your network before you implement the policies. Use the **test ip bgp neighbor** and **test bgp ipv6 neighbor** commands to test the outcome of a BGP policy. The commands output display the routes that are advertised or accepted if the specified policy is implemented.

BGP routes must be present in the forwarding table for this command to work properly. If you run the policy test on incoming routes, soft reconfiguration (configured with the **neighbor soft reconfiguration in** command) must be in effect.

NOTE: You can use the standard redirect operators to redirect the test output to network or local files. See the section *Redirection of show Command Output* in *JUNOSe System Basics Configuration Guide, Chapter 2, Command-Line Interface*.

NOTE: The output of these commands is always speculative. It does not reflect the current state of the router.

test bgp ipv6 neighbor
test ip bgp neighbor

- Use to test the effect of BGP policies on a router without implementing the policy.
- You can apply the test to routes advertised to peers or received from peers.
- You can test the following kinds of policies: distribute lists, filter lists, prefix lists, prefix trees, or route maps. If you do not specify a policy, then the test uses whatever policies are currently in effect on the router.

NOTE: If you test the current policies, the results might vary for routes learned before the current policies were activated if you did not clear the forwarding table when the policies changed.

- The following three items apply to the **test ip bgp neighbor** command only:
  - The *address-family identifier* for the route is the same as is used for identifying the neighbor.
  - If you do not specify a route, the test is performed for all routes associated with the *address-family identifier*.
  - Specifying only an address and mask without a route distinguisher causes all routes sharing the address and mask to be taken into account. Specifying only an address causes a best match to be performed for the route.
- If you completely specify a route with IP address, mask, and route distinguisher, the command displays detailed route information. Otherwise only summary information is shown. Use the **fields** option to select particular fields of interest.
- If you specify a BGP peer group by using the *peerGroupName* argument, all the members of the peer group inherit the characteristic configured with this command unless it is overridden for a specific peer.
- You can set a weight value for inbound routes filtered with a filter list.
- Example

  host1#**test ip bgp neighbor 10.12.54.21 advertised-routes distribute-list boston5 fields all**

- There is no **no** version.

Monitoring BGP

Use the **show** commands in this section to monitor BGP activity.

NOTE: The E120 router and E320 router output for **monitor** and **show** commands is identical to output from other E-series routers, except that the E120 and E320 router output also includes information about the adapter identifier in the interface specifier (*slot/adapter/port*).

Use the **baseline ip bgp** command to set the baseline on all BGP statistics.

You can use the output filtering feature of the **show** command to include or exclude lines of output based on a text string you specify. See *JUNOSe System Basics Configuration Guide, Chapter 2, Command-Line Interface*, for details.

Use the **debug ip bgp** command to get information about problems with BGP or the network.

baseline ip bgp

- Use to set the baseline on all BGP statistics as the current values.
- For example, if you issue the **baseline ip bgp** command, all the current values of BGP statistics become the baseline values. If the current value of the *Total message sent* parameter is 105, and the value goes up to 120 messages, the new value is displayed as 15.
- Example

  host1#**baseline ip bgp**

- There is no **no** version.

debug ip bgp

- Use to display information about BGP logs for inbound or outbound events, or both.
- Example

  host1#**debug ip bgp**

- There is no **no** version, but you can use the **undebug ip bgp** command to disable display of information previously enabled with the **debug ip bgp** command.

default-fields peer

- Use to specify fields that are displayed by default by a subsequently issued **show ip bgp summary** command.
- Use the **intro** keyword to enable the display of introductory information about BGP attributes.
- The order in which you specify the fields has no effect on the order in which they are displayed.
- Example

```
host1:pe2(config-router)#default-fields peer remote-as state
messages-received messages-sent up-down-time
host1:pe2#show ip bgp summary
                                          Messages    Messages
Neighbor AS   State         Up/down time   Sent        Received
1.1.1.1  100  Established   00:07:55       94          92
```

- Use the **no** version to remove fields from the output of subsequently issued **show ip bgp summary** commands.

default-fields route

- Use to specify fields that are displayed by default by any subsequently issued **show ip bgp** command that displays BGP routes.
- Use the **intro** keyword to enable the display of introductory information about BGP attributes.
- This command does not affect the output of the **show ip bgp summary** command.
- The order in which you specify the fields has no effect on the order in which they are displayed.
- Example

```
host1:pe2(config-router)#default-fields route intro next-hop med loc-pref
weight as-path
host1:pe2#show ip bgp vpnv4 all
Local BGP identifier 2.2.2.2, local AS 100
 6 routes (388 bytes)
 7 destinations (560 bytes) of which 0 have a route
 0 routes selected for route table installation
 6 path attribute entries (936 bytes)
 Local-RIB version 74. FIB version 74.

Prefix           Next-hop   MED  LocPrf  Weight  AS-path
99.99.99.11/32   1.1.1.1    1    100     0       65011
99.99.99.12/32   1.1.1.1    0    100     0       empty
99.99.99.13/32   1.1.1.1    2    100     0       empty
99.99.99.21/32   21.21.21.2 1    0               65021
99.99.99.22/32   22.22.22.2 0    32768           empty
99.99.99.23/32   23.23.23.2 2    32768           empty
```

- Use the **no** version to remove fields from the output of subsequently issued **show ip bgp** commands that displays BGP routes.

show ip as-path-access-list

- Use to display access lists.
- Example

```
host1#show ip as-path-access-list
AS Path Access List 10:
  permit [200-220]
  permit ^114
  permit ^117.*225$
AS Path Access List 11:
  deny .*
AS Path Access List 20:
  deny [1100-1250]
  permit .*
```

show ip bgp
show bgp ipv6

- Use to display the BGP routing table.
- If you specify an IP address, displays the route that best matches the specified IP address.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.

- Field descriptions
  - Learned from peer—Peer from which route was learned
  - Next hop IP address—IP address of the next router that is used when a packet is forwarded to the destination network
  - AS path—AS path through which this route has been advertised
  - Aggregator AS number—AS number of the AS that aggregated this route
  - Aggregate IP address—IP address of the router that aggregated this route
  - Origin—Origin of the route
  - MED—Multiexit discriminator for the route
  - LocPrf—Local preference for the route
  - Weight—Weight of the route
  - Communities—Community number associated with the route
  - Originator ID—Router ID of the router in the local AS that originated the route
  - Cluster ID list—List of cluster IDs through which the route has been advertised
  - Stale—Route has gone stale due to peer restart
- Example 1—Displays information about routes in the IPv6 multicast address family

```
host1#show bgp ipv6 multicast
Local BGP identifier 10.13.13.13, local AS 400
  4 routes (160 bytes)
  4 destinations (288 bytes) of which 4 have a route
  4 routes selected for route table installation
  3 path attribute entries (456 bytes)
  Local-RIB version 31. FIB version 31.

Status codes: > best, * invalid, s suppressed, d dampened, r rejected,
              a auto-summarized

  Prefix                  Peer            Next-hop               MED  LocPrf   Weight  Origin
  ::103.103.103.0/120     103.103.103.3   ::103.103.103.3        0            0       inc.
> 3ffe:0:0:1::/64         11.11.11.11     ::101.101.101.1        0    100     0       inc.
> 3ffe:0:0:3::/64         103.103.103.3   ::103.103.103.3        0            0       inc.
> 3ffe:0:1:1::/64         12.12.12.12     ::102.102.102.2        0    100     0       inc.
```

- Example 2—Displays route information for prefix 10.88.88.1/32

```
host1:pe1#show ip bgp 10.88.88.1
BGP route information for prefix 10.88.88.1/32
  Network route (best route
    Advertised to both internal and external peers
    Address Family Identifier (AFI) is ip-v4
    Subsequent Address Family Identifier (SAFI) is unicast
    Next hop IP address is 0.0.0.0 (metric 2)
    Multi-exit discriminator is 1
    Local preference is not present
    Weight is 32768
    Origin is IGP
    AS path is empty
    Extended communities empty
```

- Example 3—Displays information about IPv6 prefix 2001:0430::1/128

```
host1#show bgp ipv6 2001:0430::1/128
BGP route information for prefix 2001:1::1/128
  Received route learned from internal peer 2.2.2.2 (best route)
    Route placed in IP forwarding table
    Best to advertise to external peers
    Address Family Identifier (AFI) is ip-v6
    Subsequent Address Family Identifier (SAFI) is unicast
    MPLS in-label is none
    MPLS out-label is 17
    Next hop IP address is ::ffff:2.2.2.2 (metric 3)
    Multi-exit discriminator is 0
    Local preference is 100
    Weight is 0
    Origin is IGP
    AS path is 65021
```

- Example 4—Displays information about next hop routers for VRF PE 11 in the IPv4 VPN address family

```
host1:pe1#show ip bgp vpnv4 vrf pe11 next-hops
Indirect next-hop 11.11.11.2
    Resolution in IP route table of VR pe11
Reachable (metric 0)
IP indirect next-hop index 35
Direct next-hop ATM2/0.11 (11.11.11.2)
    Resolution in IP tunnel-route table of VR pe11
        Not reachable
      Reference count is 1

Indirect next-hop 2.2.2.2
    Resolution in IP route table of VR pe1
IP indirect next-hop index 123
Reachable (metric 100)
    Direct next-hop POS4/0 (10.10.10.1)
                POS4/1 (12.12.12.1)

Resolution in IP tunnel-route table of VR pe1
MPLS indirect next-hop index 578
Reachable (metric 100)
    Direct next-hop Push 23, POS4/0 (10.10.10.1)
                Push 43, POS4/1 (12.12.12.1)

Reference count is 1
```

- Example 5—Displays information about routes in the route-target address family

```
host1#show ip bgp route-target signaling
Local BGP identifier 13.13.13.13, local AS 100
  4 routes (240 bytes)
  3 destinations (228 bytes) of which 3 have a route
  3 routes selected for route tables installation
  0 unicast/multicast routes selected for route table installation
  0 unicast/multicast tunnel-usable routes selected for route table installation
  0 tunnel-only routes selected for tunnel-route table installation
  10 path attribute entries (1520 bytes)
  Local-RIB version 19. FIB version 19.

Status codes: > best, * invalid, s suppressed, d dampened, r rejected,
              a auto-summarized
```

```
  Prefix            Peer            Next-hop         MED LocPrf Weight Origin
> 0:0:0/0           12.12.12.12     12.12.12.12              100      0 IGP
> 100:100:1/96      11.11.11.11     11.11.11.11              100      0 IGP
  100:100:1/96      14.14.14.14     14.14.14.14              100      0 IGP
> 100:100:2/96      11.11.11.11     11.11.11.11              100      0 IGP
```

- Example 6—Displays information for routes in the route-target address family corresponding to the specified RT-MEM-NLRI

```
host1#show ip bgp route-target signaling 100:100:1/96
BGP route information for prefix 100:100:1/96
  Received route learned from internal peer 11.11.11.11 (best route)
    Route not placed in IP forwarding table
    Best to advertise to both internal and external peers
    Address Family Identifier (AFI) is ip-v4
    Subsequent Address Family Identifier (SAFI) is route-target-signaling
    Next hop IP address is 11.11.11.11 (metric 0)
    Multi-exit discriminator is not present
    Local preference is 100
    Weight is 0
    Origin is IGP
    AS path is empty
  Received route learned from internal peer 14.14.14.14
    Route not placed in IP forwarding table
    Do not advertise to any peers
    Address Family Identifier (AFI) is ip-v4
    Subsequent Address Family Identifier (SAFI) is route-target-signaling
    Next hop IP address is 14.14.14.14 (metric 0)
    Multi-exit discriminator is not present
    Local preference is 100
    Weight is 0
    Origin is IGP
    AS path is empty
```

- Example 7—Displays for network routes in the route-target address family

```
host1:pe1#show ip bgp route-target signaling network
Prefix            Weight Route-map                        Backdoor
102:111:34/96                                             No
1111111111:23:1/96                                        No

host1:pe1#show ip bgp route-target signaling network 102:111:34
Prefix            Weight Route-map                        Backdoor
102:111:34/96                                             No
```

- Example 8—Error message generated when a prefix less than 32 or greater than 96 is specified for the RT-MEM-NLRI

```
host1#show ip bgp route-target signaling 100:100:1/31
                                        ^
% Invalid route-target membership NLRI
```

- You can use the field options to display filtered information about a specified network or all networks in the BGP routing table. Only the fields that you specify are displayed, except that the Prefix field is always displayed.
- The **stale** field option shows which routes are stale due to peer restart.

- Examples

```
host1:5#show ip bgp fields peer next-hop next-hop-cost

Prefix             Peer           Next-hop      Next-hop-cost
11.11.11.11/32     3.3.3.3        3.3.3.3       Unreachable
11.11.11.11/32     4.4.4.4        4.4.4.4       Unreachable
22.22.22.22/32     3.3.3.3        3.3.3.3       Unreachable
22.22.22.22/32     4.4.4.4        4.4.4.4       Unreachable
33.33.33.33/32     3.3.3.3        3.3.3.3       Unreachable
44.44.44.44/32     4.4.4.4        4.4.4.4       Unreachable
55.55.55.55/32     0.0.0.0        0.0.0.0       0
66.66.66.66/32     6.6.6.6        6.6.6.6       Unreachable
77.77.77.77/32     57.57.57.7     57.57.57.7    1
88.88.88.88/32     57.57.57.7     57.57.57.7    1

host1:pe1#show ip bgp fields best peer next-hop stale
  Prefix             Stale Peer           Next-hop
> 10.22.22.1/32      stale 10.12.12.2     10.12.12.2
> 10.22.22.2/32      stale 10.12.12.2     10.12.12.2
> 10.22.22.3/32      stale 10.12.12.2     10.12.12.2
> 10.33.33.1/32            10.13.13.3     10.13.13.3
> 10.33.33.2/32            10.13.13.3     10.13.13.3
> 10.33.33.3/32            10.13.13.3     10.13.13.3
```

- You can use the **default-fields route** command to specify default fields to be displayed by subsequently issued **show ip bgp** commands.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option.
- Field descriptions for introductory fields displayed only when the **intro** keyword has been issued
  - Local BGP identifier—BGP router ID of the local router
  - routes—Total number of routes stored in the BGP routing table and amount of memory consumed by routes. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - destinations—Number of routes to unique prefixes stored in the BGP routing table and amount of memory consumed by routes. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - routes selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table, plus prefixes for which there are currently no routes but which have had to be withdrawn from peers to which these prefixes may been previously advertised
  - unicast/multicast routes selected for route table installation—Number of unicast routes in the BGP routing table that have been inserted into the IP routing table that are also available for use in the multicast view of the IP routing table
  - unicast/multicast tunnel-usable routes selected for route table installation—Number of unicast and multicast routes in the BGP routing table that have been inserted into the IP routing table that are also available for use in the IP tunnel routing table

  - tunnel-only routes selected for tunnel-route table installation—Number of routes in the BGP routing table that have been inserted into the IP tunnel routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Local-RIB version—Number that is increased by one each time a route in that RIB is added, removed or modified.
  - FIB version—Number that is increased by one each time BGP updates the routes in the IP routing table based on changes in the local RIB. The FIB version matches the local-RIB version when BGP has finished updating the routes in the IP route table. The FIB version is less than the local-RIB version when BGP is still in the process of updating the IP routing table.
  - Statistics baseline set—Timestamp indicating when the statistics baseline was last set
- Example

```
host1#show ip bgp 0.0.0.0 /0 fields intro
Local BGP identifier 192.168.254.79, local AS 6730
  201058 routes (12063492 bytes)
  201540 destinations (15317040 bytes) of which 201058 have a route
  193909 routes selected for route tables installation
  0 unicast/multicast routes selected for route table installation
  0 unicast/multicast tunnel-usable routes selected for route table
    installation
  0 tunnel-only routes selected for tunnel-route table installation
  35097 path attribute entries (5334744 bytes)
  Local-RIB version 20969483. FIB version 20969483.
  Statistics baseline set WED JUL 12 2006 10:31:53 METDST
  ...
```

show ip bgp advertised-routes

show bgp ipv6 advertised-routes

- Use to display the routes in the specified neighbor's or peer group's Adj-RIBs-Out table.
- For peers, the attributes displayed are those associated with the route before the application of any outbound policy.
- For peer groups, the attributes displayed are those associated with the route after the application of any outbound policy; that is, the actual advertised attributes.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- You must first enable storage of routes to the Adj-RIBs-Out tables with the **no rib-out disable** command or the **no neighbor rib-out disable** command. Otherwise, this command returns an error message.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.

- Field descriptions
  - Local BGP identifier—BGP router ID of the local router
  - routes—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - distinct destinations—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - routes selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Prefix—Prefix for the routing table entry
  - Peer—IP address of BGP peer
  - Next-hop—IP address of the next hop
  - MED—Multiexit discriminator for the route
  - LocPrf—Local preference for the route
  - Weight—Assigned path weight
  - Origin—Origin of the route
- Example

```
host1#show ip bgp neighbors 5.72.116.1 advertised-routes
Local BGP identifier 2.2.2.2, local AS 2222
  0 routes (0 bytes used), 0 distinct destinations (0 bytes used)
  0 routes selected for route table installation
  0 path attribute entries (0 bytes used)

Status codes: > best, * invalid, s suppressed, d dampened, r rejected

  Prefix             Peer            Next-hop     MED LocPrf Weight Origin
> 0.0.0.0/0         5.72.116.1     5.72.1.1               0       IGP
> 10.10.0.87/32     5.72.116.1     5.72.1.1               0       inc.
> 13.13.13.13/32    5.72.116.1     5.72.1.1               0       IGP
> 33.0.0.0/16       0.0.0.0        5.72.1.1         1     32768   inc.
> 33.0.0.0/24       0.0.0.0        5.72.1.1         1     32768   inc.
> 44.44.0.0/16      5.72.116.1     5.72.1.1               0       inc.
```

show ip bgp aggregate-address

show bgp ipv6 aggregate-address

- Use to display information about aggregate addresses.
- Field descriptions
  - Prefix—Prefix of the aggregate address
  - AS set—ASs in the AS-set path
  - Summary only—Displays a summary of aggregate address information

- Attribute map—Displays the attribute maps for aggregate addresses
- Advertise map—Displays the advertise maps for aggregate addresses

- Example

```
host1#show bgp ipv6 aggregate-address
Prefix             AS set  Summ only  Attribute map   Advertise map     Suppress map
3ffe::/48          No      No         None            None              None
```

show ip bgp cidr-only

- Use to display information about routes that have nonnatural network masks.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local BGP identifier—BGP router ID of the local router
  - routes—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - distinct destinations—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - routes selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Prefix—Prefix for the routing table entry
  - Peer—IP address of BGP peer
  - Next-hop—IP address of the next hop
  - MED—Multiexit discriminator for the route
  - LocPrf—Local preference for the route
  - Weight—Assigned path weight
  - Origin—Origin of the route
- Example

```
host1#show ip bgp cidr-only
Local BGP identifier 111.111.111.111, local AS 444
  0 routes (0 bytes used), 0 distinct destinations (0 bytes used)
  0 routes selected for route table installation
  0 path attribute entries (0 bytes used)
```

```
Status codes: > best, * invalid, s suppressed, d dampened, r rejected

  Prefix         Peer       Next-hop    MED LocPrf  Weight  Origin
  33.0.0.0/24   5.72.1.1   5.72.1.1     1           0       inc.
> 44.44.0.0/24  0.0.0.0    192.168.1.1  1           32768   inc.
```

show ip bgp community
show bgp ipv6 community

- Use to display all routes that are members of the specified BGP community. Does not accept regular expressions.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- Specify the community number in *AA:NN* format:
  - *AA*—Number that identifies the autonomous system
  - *NN*—Number that identifies the community within the autonomous system
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local router ID—BGP router ID of the local router
  - local AS—Local autonomous system number
  - paths—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - distinct prefixes—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - paths selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Prefix—Prefix for the route table entry
  - Peer—IP address of BGP peer
  - Next-hop—IP address of the next hop
  - MED—Multiexit discriminator
  - CalPrf—Calculated preference
  - Weight—Assigned path weight
  - Origin—Origin of the route

- Example

```
host1#show ip bgp community 999:999
Local router ID 192.168.1.153, local AS 100
  40845 paths, 40845 distinct prefixes (2940840 bytes used)
  40845 paths selected for route table installation
  13651 path attribute entries (1864908 bytes used)

  Prefix        Peer       Next-hop        MED CalPrf Weight Origin
> 24.0.0.0/12   10.5.0.48  10.5.0.48            100    100    IGP
> 24.4.252.0/22 10.5.0.48  10.5.0.48            100    100    IGP
> 24.6.0.0/23   10.5.0.48  10.5.0.48            100    100    IGP
> 24.6.11.0/24  10.5.0.48  10.5.0.48            100    100    IGP
```

show ip bgp community-list

show bgp ipv6 community-list

- Use to display all routes that are members of communities on the specified BGP community list.
- Accepts regular expressions.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local router ID—BGP router ID of the local router
  - local AS—Local autonomous system number
  - paths—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - distinct prefixes—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - paths selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Prefix—Prefix for the routing table entry
  - Peer—IP address of BGP peer
  - Communities—Community number in *AA*:*NN* format:
    - *AA*—Number that identifies the autonomous system
    - *NN*—Number that identifies the community within the autonomous system

- Example

```
host1#show ip bgp community-list 1 fields peer communities
Local router ID 192.168.1.153, local AS 100
  72077 paths, 72077 distinct prefixes (5189544 bytes used)
  72077 paths selected for route table installation
  21627 path attribute entries (2957324 bytes used)

Prefix             Peer           Communities
3.0.0.0/8          10.5.0.48      777:777 888:888
4.0.0.0/8          10.5.0.48      777:777 888:888
4.17.106.0/24      10.5.0.48      777:777 888:888
4.17.115.0/24      10.5.0.48      777:777 888:888
6.0.0.0/8          10.5.0.48      777:777 888:888
9.2.0.0/16         10.5.0.48      777:777 888:888
9.20.0.0/17        10.5.0.48      777:777 888:888
12.0.0.0/8         10.5.0.48      777:777 888:888
```

show ip bgp dampened-paths

show bgp ipv6 dampened-paths

- Use to display information about dampened routes.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local router ID—IP address of the local router
  - local AS—Number of the local AS
  - Route flap dampening—Status of route flap dampening (enabled or disabled)
  - Decay half-life—Time (in minutes) after which a penalty is decreased. After the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default).
  - Cutoff threshold—Value of the penalty for a flapping route below which the route is unsuppressed
  - Reuse threshold—Time (in hours:minutes:seconds) after which the path will be made available
  - Maximum hold-down time—Interval, in seconds, after not receiving a keepalive message that the software declares a peer dead
  - route flap history—Status of route flap history for route paths
  - Prefix—The prefix for the IP address
  - Peer—IP address of the BGP peer
  - Status—Status of route dampening of the route path
  - Figure of Merit—A measure of the route's stability. Higher values indicate more recent route flap activity or less stability.
  - Time until Reuse/Remove—Time until the route is either reused (if currently suppressed) or its history entry is removed (if currently available)

- Example

```
host1#show ip bgp dampened-paths
Local router ID 192.168.1.218, local AS 100
  Route flap dampening is enabled
  Decay half-life is 10 minutes while reachable, 20 minutes while
unreachable
  Cuttoff threshold is 2000, reuse threshold is 750
  Maximum hold-down time is 20 minutes
  60 paths have active route flap histories (4560 bytes used)
 11 paths are suppressed
                                                Figure   Time until
Prefix          Peer      Status                of Merit Reuse/Remove
24.31.128.0/19   10.2.1.48 Suppressed/Reachable   2681    00:17:00
24.93.128.0/19   10.2.1.48 Suppressed/Reachable   2681    00:17:00
24.95.0.0/19     10.2.1.48 Suppressed/Reachable   2681    00:17:00
128.192.0.0/16   10.2.1.48 Available              1997    00:15:08
148.161.0.0/16   10.2.1.48 Available              1997    00:15:10
164.81.0.0/16    10.2.1.48 Available              1997    00:15:11
192.29.60.0/24   10.2.1.48 Available              1997    00:15:12
192.58.228.0/24  10.2.1.48 Available              1997    00:15:15
192.88.8.0/24    10.2.1.48 Available              1997    00:15:17
192.107.253.0/24 10.2.1.48 Suppressed/Unreachable 4331    00:19:42
192.195.44.0/24  10.2.1.48 Suppressed/Reachable   2923    00:19:15
192.195.49.0/24  10.2.1.48 Suppressed/Reachable   2923    00:19:15
192.195.50.0/24  10.2.1.48 Suppressed/Reachable   2923    00:19:15
192.197.150.0/24 10.2.1.48 Available              1997    00:15:25
192.222.89.0/24  10.2.1.48 Suppressed/Unreachable 2788    00:19:42
204.17.195.0/24  10.2.1.48 Suppressed/Reachable   2923    00:17:20
204.52.186.0/24  10.2.1.48 Available              1997    00:15:26
204.68.178.0/24  10.2.1.48 Available              1000    00:19:38
204.101.0.0/16   10.2.1.48 Available              1997    00:15:29
204.128.227.0/24 10.2.1.48 Suppressed/Reachable   2923    00:17:16
204.146.24.0/22  10.2.1.48 Available              1997    00:15:30
204.146.24.0/24  10.2.1.48 Available              1997    00:15:30
```

show ip bgp filter-list
show bgp ipv6 filter-list

- Use to display all routes whose AS-path matches the specified AS-path access list.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local router ID—BGP router ID of the local router
  - local AS—Local autonomous system number
  - paths—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - distinct prefixes—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.

- paths selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
- path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
- Prefix—Prefix for the routing table entry
- Next-hop—IP address of the next hop
- MED—Multiexit discriminator
- CalPrf—Calculated preference
- Weight—Assigned path weight
- AS path—Autonomous system path

- Example

```
host1#show ip bgp filter-list 1
Local router ID 192.168.1.153, local AS 100
  72080 paths, 72080 distinct prefixes (5189760 bytes used)
  72080 paths selected for route table installation
  21667 path attribute entries (2962828 bytes used)

  Prefix        Next-hop  MED CalPrf Weight AS-path
> 6.0.0.0/8     10.5.0.48   100        100  11488 701 7018 7170
> 12.0.0.0/8    10.5.0.48   100        100  11488 701 1740 7018
> 12.1.248.0/24 10.5.0.48   100        100  11488 701 7018 13391
> 12.2.6.0/24   10.5.0.48   100        100  11488 701 7018 11101
> 12.2.7.0/24   10.5.0.48   100        100  11488 701 7018 11101
> 12.2.76.0/24  10.5.0.48   100        100  11488 701 7018 11812
> 12.2.99.0/24  10.5.0.48   100        100  11488 701 7018 10656
> 12.2.109.0/24 10.5.0.48   100        100  11488 701 7018 10656
> 12.2.169.0/24 10.5.0.48   100        100  11488 701 7018 11806
> 12.4.114.0/24 10.5.0.48   100        100  11488 701 7018 14065
> 12.4.119.0/24 10.5.0.48   100        100  11488 701 7018 14065
> 12.4.175.0/24 10.5.0.48   100        100  11488 701 7018 11895
> 12.4.196.0/22 10.5.0.48   100        100  11488 701 7018 12163
> 12.5.48.0/21  10.5.0.48   100        100  11488 701 7018 12163
> 12.5.164.0/24 10.5.0.48   100        100  11488 701 7018 11134
> 12.6.42.0/23  10.5.0.48   100        100  11488 701 7018 11090
```

show ip bgp flap-statistics

show bgp ipv6 flap-statistics

- Use to display information about flap statistics.
- Field descriptions
  - Local BGP identifier—BGP router ID of the local router where route flap dampening is enabled
  - local AS—Local autonomous system number
  - Route flap dampening—Status of route flap dampening (enabled or disabled)
  - Default decay half-life—Time (in minutes) after which a penalty is decreased. After the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default).

  - Default cutoff threshold—Value of the penalty for a flapping route below which the route is unsuppressed
  - Default reuse threshold—Time in minutes after which the path will be made available
  - Default maximum hold-down time—Interval, in seconds, after not receiving a keepalive message that the software declares a peer dead
  - route flap history—Status of route flap history for route paths
  - Prefix—Prefix for the routing table entry
  - Peer—IP address of BGP peer
  - Status—Status of route dampening of the route path
  - Figure of Merit—Measure of the route's stability. Higher values indicate more recent route flap activity or less stability.
  - Time until Reuse/Remove—Time in hours:minutes:seconds until the route is either reused (if currently suppressed) or its history entry is removed (if currently available)
- Example

```
host1#show ip bgp flap-statistics
Local BGP identifier 192.168.1.232, local AS 100
  Route flap dampening is enabled
  Default decay half-life is 15 minutes
  Default cutoff threshold is 2000, default reuse threshold is 750
  Default maximum hold-down time is 60 minutes
  307 paths have active route flap histories (27016 bytes used)
  5 paths are suppressed
                                                    Figure      Time until
Prefix             Peer            Status         of Merit     Reuse/Remove
24.201.0.0/18      192.168.1.158   Available       925         00:58:23
24.201.64.0/18     192.168.1.158   Available       925         00:58:23
52.128.224.0/19    192.168.1.158   Available       750         00:54:12
61.8.0.0/19        192.168.1.158   Available       993         00:59:53
61.8.30.0/24       192.168.1.158   Available       993         00:59:53
62.229.73.0/24     192.168.1.158   Unreachable     925         00:58:23
63.69.150.0/24     192.168.1.158   Available       750         00:54:12
```

show ip bgp inconsistent-as

show bgp ipv6 inconsistent-as

- Use to display information about routes that have inconsistent AS-paths.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local BGP identifier—BGP router ID of the local router
  - local AS—Local autonomous system number
  - routes—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.

- distinct destinations—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
- routes selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
- path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
- Prefix—Prefix for the routing table entry
- Next-hop—IP address of the next hop
- MED—Multiexit discriminator for the route
- LocPrf—Local preference for the route
- Weight—Assigned path weight
- Origin—Origin of the route
- AS-path—AS-path through which this route bas been advertised

- Example

```
host1#show ip bgp inconsistent-as
Local BGP identifier 192.168.1.10, local AS 123
  0 routes (0 bytes used), 0 distinct destinations (0 bytes used)
  0 routes selected for route table installation
  0 path attribute entries (0 bytes used)

Status codes: > best, * invalid, s suppressed, d dampened, r rejected

  Prefix             Next-hop         MED LocPrf Weight AS-path
> 4.0.0.0/8          0.0.0.0            1         32768 empty
  4.0.0.0/8          192.168.1.1        0 11488     701 1
```

show ip bgp longer-prefixes

show bgp ipv6 longer-prefixes

- Use to display all routes with a prefix that is equal to or more specific than the specified prefix.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local router ID—BGP router ID of the local router
  - local AS—Local autonomous system number
  - paths—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.

- distinct prefixes—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
- paths selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
- path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
- Prefix—Prefix for the routing table entry
- Peer—IP address of BGP peer
- Next-hop—IP address of the next hop
- MED—Multiexit discriminator
- CalPrf—Calculated preference
- Weight—Assigned path weight
- Origin—Origin of the route

- Example

```
host1#show ip bgp 12.2.0.0 255.255.0.0 longer-prefixes
Local router ID 192.168.1.153, local AS 100
  72074 paths, 72074 distinct prefixes (5189328 bytes used)
  72074 paths selected for route table installation
  21685 path attribute entries (2965327 bytes used)

  Prefix         Peer       Next-hop   MED CalPrf  Weight  Origin
> 12.2.6.0/24    10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.7.0/24    10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.76.0/24   10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.88.0/22   10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.97.0/24   10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.99.0/24   10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.109.0/24  10.5.0.48  10.5.0.48    100         100     IGP
> 12.2.169.0/24  10.5.0.48  10.5.0.48    100         100     IGP
```

show ip bgp neighbors
show bgp ipv6 neighbors

- Use to display information about BGP neighbors.
- Field descriptions
  - BGP neighbor ID—BGP identifier of the BGP neighbor
  - remote AS—Remote AS of the BGP neighbor
  - Description—Textual description of the BGP neighbor
  - Member of peer group—Name of the peer group of which this BGP neighbor is a member
  - Remote router ID—Router ID of the remote router
  - negotiated BGP version—BGP version being used to communicate with the neighbor

- Administrative status—Desired state of the peer connection
- Connection state—Current state of the BGP connection
- Connection has been established—Time that TCP connection was established
- Reason for last reset—Reason for last reset of the BGP session
- TCP error code—TCP connection error type
- Default originate—Status of default originate (enabled or disabled)
- EBGP multi-hop—Status of EBGP multihop (enabled or disabled)
- IBGP single-hop—Status of IBGP single hop (enabled or disabled)
- Next hop self—Status of next-hop self (enabled or disabled)
- Route reflector status—Identifies the neighbor as a route-reflector client
- Neighbor weight—Weight of routes from the BGP neighbor
- Incoming update distribute list—Distribute list for incoming routes, if configured
- Outgoing update distribute list—Distribute list for outgoing routes, if configured
- Incoming update filter list—Update filter list for incoming routes, if configured
- Outgoing update filter list—Update filter list for outgoing route, if configured
- Weight filter list—Weight filter list for routes, if configured
- Incoming route map—Incoming route map, if configured
- Outgoing route map—Outgoing route map, if configured
- Connect retry interval—Time between a BGP peer's attempts to reestablish a connection to the neighbor
- Minimum route advertisement interval—Minimum time between route advertisements
- Minimum AS origination interval—Minimum time between advertisement of changes within the speaker's AS
- Configured keep-alive interval—Frequency of keep-alive messages generated
- Negotiated keepalive interval—Negotiated frequency of keep-alive messages generated
- Configured hold time—Configured maximum time allowed between received messages
- Negotiated hold time—Negotiated maximum time allowed between received messages
- Configured update source IP address—IP address used when sending update messages
- Local IP address—Local IP address used for TCP communication to this peer

- Local port—Local TCP port number used for TCP communication to this peer
- Remote IP address—Remote IP address used for TCP communication to this peer
- Remote port—Remote IP address used for TCP communication to this peer
- Total messages sent—Total BGP messages sent to this neighbor
- Total messages received—Total BGP messages received from this neighbor
- Total update messages sent—Total BGP update messages sent to this neighbor
- Total update messages received—Total BGP update messages received from this neighbor
- Time since last update message was received—Time since last BGP update message was received from this neighbor
- Address Family dependent capabilities—Lists type of ORF send and receive capability per address family and whether it is advertised (configured) or received
- Maximum number of ORF entries—Limit of ORF entries that will be accepted from the neighbor
- Capability advertisement—Lists whether the specific capability (capabilities option, deprecated dynamic capability negotiation, dynamic capability negotiation, multiprotocol extensions, route refresh, route refresh (Cisco proprietary), four octet AS numbers, and graceful restart) has been sent, received, or both
- Multi-protocol extensions negotiation—Lists the relevant address family and whether it has been sent, received, or used
- BFD—Status of BFD configuration, enabled, enabled but not supported because the peer is an IBGP neighbor a multihop EBGP neighbor, or disabled
- BFD session—Type and address of peer to which BFD session is established
- Minimum transmit interval—Desired interval between BFD packets transmitted to members of peer group
- Minimum receive interval—Desired interval between BFD packets received from members of peer group
- Multiplier—Number of BFD packets that can be missed before declaring BFD session down
- Negotiated detection time—Interval between BFD packets negotiated by peers
- Advertise-map—Name of route map that specifies routes to be advertised when routes in conditional route maps are matched
- Condition-map—Name of route map that specifies routes to be matched by routes in the BGP routing table

  - Sequence—Position of the specified advertise route map in a list of advertise route maps configured for a particular peer within the same address-family. A lower sequence number has a higher priority; that route map is processed before one with a higher sequence number.
  - Status—Status of the routes specified by the route map, advertise (route map condition has been met) or withdraw (route map condition has not been met; regardless of this status, the specified routes might be governed by another route map with a lower sequence number and actually advertised or not according to that map

- Example

```
host1#show ip bgp neighbors
BGP neighbor ID 10.2.1.48, remote AS 11488 (external peer)
  Remote router ID is 172.31.1.48, negotiated BGP version is 4
  Administrative status is Start, connection state is Established
  Reason for last reset was tcp connection error
  TCP error code 60 (Connection timed out)
  Connection has been established 1 time, up for 0 17:42:31
  Options:
    Default originate is disabled
    EBGP multi-hop is enabled
    IBGP single-hop is disabled
    Next hop self is disabled
    seconds
Policy:
    Neighbor weight is 100
 Timers:
    Connect retry interval is 120 seconds
    Minimum route advertisement interval is 30 seconds
 Minimum AS origination interval is 10 seconds
    Configured keep-alive interval is 30 seconds, negotiated 30
       seconds
    Configured hold time is 90 seconds, negotiated 90
 TCP connection:
    Local IP address is 192.168.1.218, local port is 1024
    Remote IP address is 10.2.1.48, remote port is 179
 Statistics:
    Total of 4100 messages sent, 44913 messages received
    2053 update messages sent, 42785 update messages received
    0 00:00:17 since last update message was received
```

- Fields relevant to multiprotocol extensions:

```
Multi-protocol extensions negotiation:
    ip-v4 unicast: sent, received, used
    ip-v6 unicast-labeled: sent, received, used
```

- For the graceful restart capability, additional information is presented.
  - Fields concerning graceful restart attributes that apply to peers as a whole (for all address families):

```
Graceful restart negotiation:
    Sent restart time is 120 seconds
    Sent restart state bit is zero (we are not restarting)
    Received restart time is 120 seconds
    Received restart state bit is zero (peer is not restarting)
    Maximum time for keeping stale paths is 360 seconds
```

- Fields concerning attributes that apply to peers a particular address family:

```
Peer is capable of preserving forwarding stat(3)
Peer preserved forwarding state during last restart
We have received an end-of-rib marker from the peer
We have sent an end-of-rib marker to the peer
```

- Fields relevant if the peer is currently restarting:

```
Graceful restart waiting for the session to come back up
  Restart-time advertised by the peer is 120 seconds
  Remaining time for the peer to come back up is 117 seconds
  Remaining time for keeping stale routes from the peer is 357 seconds
```

- Fields relevant during reconvergence after the peer has restarted:

```
Graceful restart negotiation:
  Sent restart time is 120 seconds
  Sent restart state bit is zero (we are not restarting)
  Received restart time is 120 seconds
  Received restart state bit is zero (peer is not restarting)
  Maximum time for keeping stale paths is 300 seconds
  Remaining time for keeping stale routes from the peer is 297 seconds
```

- For BFD, additional information is presented.
  - Fields relevant to BFD when BFD is not configured:

```
BFD is disabled
```

  - Fields relevant to BFD when BFD is configured for an IBGP peer:

```
BFD is enabled but not supported (IBGP neighbor)
```

  - Fields relevant to BFD when BFD is configured for a multihop EBGP peer:

```
BFD is enabled but not supported (multi-hop EBGP neighbor)
```

  - Fields relevant to BFD when BFD is configured but the BGP session is not established:

```
BFD is enabled:
     Single-hop IPv4 BFD session to 1.2.3.4
     Minimum transmit interval is 300 ms
     Minimum receive interval is 300 ms
     Multiplier is 3
     Waiting for BGP to become established before initiating BFD session
```

  - Fields relevant to BFD when BFD is configured, the BGP session is established, but the BFD protocol session is not up:

```
BFD is enabled:
     Single-hop IPv4 BFD session to 1.2.3.4
     Minimum transmit interval is 300 ms
     Minimum receive interval is 300 ms
     Multiplier is 3
     BFD session is down
```

- Fields relevant to BFD when BFD is configured, the BGP session is established, and the BFD protocol session is up:

```
BFD is enabled:
     Single-hop IPv4 BFD session to 1.2.3.4
     Minimum transmit interval is 300 ms
     Minimum receive interval is 300 ms
     Multiplier is 3
     BFD session is up for 00:00:50
     Negotiated detection time is 900 ms
```

- Fields relevant to conditional advertisement:

```
Advertise-map is advertisetoR1
    Condition-map: trigger1
    Sequence: 5
    Status: Withdraw
Advertise-map is alternatetoR1
    Condition-map: trigger2
    Sequence: 10
    Status: Advertise
```

show ip bgp neighbors dampened-routes

show bgp ipv6 neighbors dampened-routes

- Use to display information about routes with a dampening history for the specified BGP neighbor.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local BGP identifier—BGP router ID of the local router
  - routes—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.
  - distinct destinations—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - routes selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Prefix—Prefix for the routing table entry
  - Peer—IP address of BGP peer
  - Next-hop—IP address of the next hop
  - MED—Multiexit discriminator for the route

- LocPrf—Local preference for the route
- Weight—Assigned path weight
- Origin—Origin of the route

- Example

```
host1#show ip bgp neighbors 192.168.1.158 dampened-routes
Local BGP identifier 192.168.1.232, local AS 100
  120 routes (5760 bytes used), 94 distinct destinations (9024 bytes used)
  67 routes selected for route table installation
  23 path attribute entries (3450 bytes used)

Status codes: > best, * invalid, s suppressed, d dampened, r rejected

  Prefix            Peer            Next-hop MED LocPrf Weight Origin
 d12.8.12.0/24      192.168.1.158   192.168.1.1              0 IGP
 d24.48.12.0/24     192.168.1.158   192.168.1.1              0 IGP
 d24.72.12.0/24     192.168.1.158   192.168.1.1              0 inc.
 d24.116.12.0/23    192.168.1.158   192.168.1.1              0 IGP
 d24.143.12.0/24    192.168.1.158   192.168.1.1              0 IGP
 d24.154.12.0/24    192.168.1.158   192.168.1.1              0 inc.
 d24.216.12.0/24    192.168.1.158   192.168.1.1              0 IGP
 d24.240.12.0/24    192.168.1.158   192.168.1.1              0 IGP
 d24.244.12.0/22    192.168.1.158   192.168.1.1              0 IGP
 d24.246.12.0/22    192.168.1.158   192.168.1.1              0 IGP
 d61.0.12.0/24      192.168.1.158   192.168.1.1              0 IGP
 d61.11.12.0/24     192.168.1.158   192.168.1.1              0 IGP
 d62.74.12.0/22     192.168.1.158   192.168.1.1              0 IGP
 d62.76.12.0/22     192.168.1.158   192.168.1.1              0 IGP
 d63.65.12.0/24     192.168.1.158   192.168.1.1              0 inc.
 d63.73.12.0/24     192.168.1.158   192.168.1.1              0 IGP
```

show ip bgp neighbors paths

show bgp ipv6 neighbors paths

- Use to display path information for the specified BGP neighbor.
- This command displays only the most common path attributes. BGP internally maintains additional attributes that are not displayed—for example, the MED, local preference, and communities attributes.
- Field descriptions
  - Address—Hexadecimal number that uniquely identifies the path attributes
  - Refcount—Number of routes that share the path attributes
  - Origin—Value of the origin path attribute
  - Next-hop—Value of the next-hop path attribute
  - AS-path—Value of the AS-path attribute
- Example

```
host1#show ip bgp neighbors 1.02.3.4 paths
Address     Refcount  Origin  Next-hop         AS-path
0xC384BD0   1         IGP     192.168.1.1      11488 701 2853 5515 764
0xC384C40   1         IGP     192.168.1.1      11488 701 4183
0xC384CB0   1         IGP     192.168.1.1      11488 701 1239 1833 1833 1833 1299 8308
0xC384D20   1         IGP     192.168.1.1      11488 701 6453 786
0xC384D90   1         IGP     192.168.1.1      11488 701 6453 1103 1103
0xC384E00   1         IGP     192.168.1.1      11488 701 6762 9116 9116 9116 6888 6888
0xC384E70   1         IGP     192.168.1.1      11488 701 6453 8297 6758
```

```
0xC384EE0   1         IGP     192.168.1.1      11488 701 5511 3215
0xC384F50   1         IGP     192.168.1.1      11488 701 3561 5683 5551
0xC384FC0   1         IGP     192.168.1.1      11488 701 1239 1755 1273 8793 8793 8793
0xC385030   1         IGP     192.168.1.1      11488 701 5705 5693
```

show ip bgp neighbors received prefix-filter

- Use to display prefix-list outbound route filters received from the neighbor.
- Field descriptions
  - seq—Sequence number of the entry in the prefix list
  - permit, deny—Condition statement for addresses matching the listed address
- Example

```
host1#show ip bgp neighbors 192.168.1.158 received prefix-filter
ip prefix-list filter 192.168.1.158 for address family ipv4:unicast
   seq 5 permit 10.1.1.1/32
   seq 10 permit 10.1.1.2/32
   seq 15 permit 10.1.1.3/32
```

show ip bgp neighbors received-routes

show bgp ipv6 neighbors received-routes

- Use to display routes originating from the specified BGP neighbor before inbound policy is applied.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Prefix—Prefix for the routing table entry
  - Peer—IP address of BGP peer
  - Next-hop—IP address of the next hop
  - MED—Multiexit discriminator for the route
  - LocPrf—Local preference for the route
  - Weight—Assigned path weight
  - Origin—Origin of the route
- Example

```
host1#show ip bgp neighbors 192.168.1.158 received-routes
Local BGP identifier 111.111.111.111, local AS 444
  0 routes (0 bytes used), 0 distinct destinations (0 bytes used)
  0 routes selected for route table installation
  0 path attribute entries (0 bytes used)

Status codes: > best, * invalid, s suppressed, d dampened, r rejected

Prefix          Peer           Next-hop      MED LocPrf Weight Origin
>0.0.0.0/0      192.168.1.158  192.168.1.158              0       IGP
>13.13.13.13/32 192.168.1.158  192.168.1.158       0      0       IGP
```

show ip bgp neighbors routes
show bgp ipv6 neighbors routes

- Use to display, after inbound policy is applied, all routes that originate from the specified BGP neighbor.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Field descriptions
  - Local router ID—BGP router ID of the local router
  - local AS—Local autonomous system number
  - paths—Total number of routes stored in the BGP routing table. If several peers have advertised a route to the same prefix, all routes are included in this count.‘
  - distinct prefixes—Number of routes to unique prefixes stored in the BGP routing table. If several peers have advertised a route to the same prefix, only the best route is included in this count.
  - paths selected for route table installation—Number of routes in the BGP routing table that have been inserted into the IP routing table
  - path attribute entries—Number of distinct path attributes stored in BGP's internal path attributes table. If BGP receives two routes for different prefixes but with identical path attributes, BGP will create only one entry in its internal path attribute table and share it between the two routes to conserve memory.
  - Local-RIB version—Number that is increased by one each time a route in that RIB is added, removed or modified.
  - FIB version—Number that is increased by one each time BGP updates the routes in the IP routing table based on changes in the local RIB. The FIB version matches the local-RIB version when BGP has finished updating the routes in the IP route table. The FIB version is less than the local-RIB version when BGP is still in the process of updating the IP routing table.
  - Prefix—Prefix for the routing table entry
  - Peer— IP address of BGP peer
  - Next-hop—IP address of the next hop
  - MED—Multiexit discriminator
  - CalPrf—Calculated preference
  - Weight—Assigned path weight
  - Origin—Origin of the route

- Example

```
host1#show bgp ipv6 neighbors 12.12.12.12 routes
Local BGP identifier 11.11.11.11, local AS 400
  5 routes (200 bytes)
  5 destinations (360 bytes) of which 5 have a route
  5 routes selected for route table installation
  4 path attribute entries (608 bytes)
  Local-RIB version 33. FIB version 33.

Status codes: > best, * invalid, s suppressed, d dampened, r rejected,
              a auto-summarized

  Prefix               Peer           Next-hop            MED LocPrf Weight
Origin
> 3ffe:0:1:1::/64      12.12.12.12    ::102.102.102.2       0    100      0
inc.
```

show ip bgp network
show bgp ipv6 network

- Use to display information about networks in an AS.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- Example

```
host1#show bgp ipv6 network
Prefix                      Weight   Route-map    Backdoor
3ffe:0:0:2::/64                                  No
```

show ip bgp next-hops
show bgp ipv6 next-hops

- Use to display information about BGP next hops.
- Specify all indirect next hops or a particular indirect next hop.
- Example

```
host1:3#show ip bgp next-hops
Indirect next-hop 4.4.4.4
  Reachable (metric 2)
  Direct next-hop atm2/0.34 (34.34.34.4)
  Reference count is 3

Indirect next-hop ::ffff:2.2.2.2
  MPLS stacked label 17
  Reachable (metric 3)
  Direct next-hop  tun mpls:vpnInL17-23
  Reference count is 1

Indirect next-hop 5.5.5.5
  Reachable (metric 2)
  Direct next-hop atm2/0.35 (35.35.35.5)
  Reference count is 3
```

```
Indirect next-hop 6.6.6.6
  Reachable (metric 3)
  Direct next-hop atm2/0.34 (34.34.34.4)
                  atm2/0.35 (35.35.35.5)
  Reference count is 3

Indirect next-hop 13.13.13.1
  Not reachable
  Reference count is 2
```

show ip bgp paths
show bgp ipv6 paths

- Use to display information about BGP paths.
- This command displays only the most common path attributes. BGP internally maintains additional attributes that are not displayed—for example, the MED, local preference, and communities attributes.
- Field descriptions
  - Address—Hexadecimal number that uniquely identifies the path attributes
  - Refcount—Number of routes that share the path attributes
  - Origin—Value of the origin path attribute
  - Next-hop—Value of the next-hop path attribute
  - AS-path—Value of the AS-path attribute
- Example

```
host1#show bgp ipv6 paths
Address     Refcount  Metric  AS-path
0x4B311118  1                 100
0x4C548224  1         0       100
0x4C548530  1         0       200
0x4C548704  2         0       300
```

show ip bgp peer-group
show bgp ipv6 peer-group

- Use to display information about BGP peer groups.
- Field descriptions
  - BGP peer group—Name of a BGP peer group
  - remote AS—Number of the remote AS
  - Description—Textual description of the BGP peer group
  - Members—IP addresses of the members of the BGP peer group
  - Default originate—Status of default origination of the BGP peer group
  - EBGP multi-hop—Status of EBGP multihop for the peer group
  - IBGP single-hop—Status of IBGP single hop for the peer group
  - BFD—Status of BFD configuration for the peer group
  - BFD session—Type and address of peer to which BFD session is established

- Minimum transmit interval—Desired time interval between BFD packets transmitted to members of peer group
- Minimum receive interval—Desired time interval between BFD packets received from members of peer group
- Multiplier—Number of BFD packets that can be missed before declaring BFD session down
- Next hop self—Status of next-hop self information for the peer group
- Peers are route reflector clients—BGP peer group is configured as a route reflector. This field does not appear when route reflectors are not configured.
- weight—Neighbor weights assigned to BGP peer groups
- Incoming update distribute list—Distribute lists for incoming routes, if configured
- Outgoing update distribute list—Distribute list for outgoing routes, if configured
- Incoming update filter list—Filter list for incoming routes, if configured
- Outgoing update filter list—Filter list for outgoing routes, if configured
- Weight filter list—Weight filter list for routes, if configured
- Incoming route map—Incoming route map, if configured
- Outgoing route map—Outgoing route map, if configured
- Minimum route advertisement interval—Minimum time between route advertisements
- Configured update source IP address—IP address used when sending update messages
- Advertise-map—Name of route map that specifies routes to be advertised when routes in conditional route maps are matched
- Condition-map—Name of route map that specifies routes to be matched by routes in the BGP routing table
- Sequence—Position of the specified advertise route map in a list of advertise route maps configured for a particular peer group within the same address-family. A lower sequence number has a higher priority; that route map is processed before one with a higher sequence number.
- Status—Status of the routes specified by the route map, advertise (route map condition has been met) or withdraw (route map condition has not been met; regardless of this status, the specified routes might be governed by another route map with a lower sequence number and actually advertised or not according to that map

- Example

```
host1#show ip bgp peer-group
BGP peer-group leftcoast, remote AS 200
  Peer-group members are external peers
  Local AS 100
  Administrative status is Start
  EBGP multi-hop is disabled
  IBGP single-hop is disabled
```

```
BFD is enabled:
   Single-hop IPv4 BFD session
   Minimum transmit interval is 300 ms
   Minimum receive interval is 300 ms
   Multiplier is 3
Maximum update message size is 1024 octets
Neighbor weight is 0
Connect retry interval is 10 seconds initially
Configured keep-alive interval is 30 seconds
Configured hold time is 90 seconds
Minimum route advertisement interval is 30 seconds
Minimum AS origination interval is 10 seconds
Graceful restart negotiation:
  Restart time is 120 seconds
  Stale paths time is 360 seconds

Configuration for address family ipv4:unicast
  RIB-out is disabled
  Default originate is disabled
  Next hop self is disabled
  Next hop unchanged is disabled
  Don't send communities
  Inbound soft reconfiguration is disabled
  Private AS number stripping is disabled
  Override site AS with provider AS is disabled
  No loops in the received AS-path are allowed
  Members: 10.2.2.2 10.3.3.3
```

- Fields relevant to conditional advertisement:

```
Advertise-map is advertisetoR1
    Condition-map: trigger1
    Sequence: 5
    Status: Withdraw
Advertise-map is alternatetoR1
    Condition-map: trigger2
    Sequence: 10
    Status: Advertise
```

show ip bgp quote-regexp
show bgp ipv6 quote-regexp

- Use to display information about BGP routes whose AS-path matches the specified regular expression.
- Use with only a single regular expression element.
- You can use output filtering.
- You must enclose any elements containing a space within quotation marks ("*element*").
- Regular expressions match numbers for which the specified path is a substring—for example, if you specify *20*, *200* matches because *20* is a substring of *200*. You can disallow substring matching by using the underscore (_) metacharacter to constrain matching to the specified pattern, for example, *_20_*.

- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.

show ip bgp regexp
show bgp ipv6 regexp

- Use to display information about BGP routes whose AS-path matches the specified regular expression.
- Use with one or more regular expression elements.
- You cannot use output filtering.
- You do not have to enclose elements containing a space within quotation marks.
- Regular expressions match numbers for which the specified path is a substring—for example, if you specify *20*, *200* matches because *20* is a substring of *200*. You can disallow substring matching by using the underscore (_) metacharacter to constrain matching to the specified pattern, for example, *_20_*.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.

Examples for regexp and quote-regexp

In many cases, you can use either **show ip bgp regexp** or **show ip bgp quote-regexp** with the same results. For example, to show all routes whose AS-path starts with 200 you can use either command as follows:

```
host1#show ip bgp regexp ^200
Local router ID 192.168.1.232, local AS 100
  6 paths, 3 distinct prefixes (324 bytes used)
  3 paths selected for route table installation
  7 path attribute entries (872 bytes used)

Prefix        Next-hop  MED  CalPrf  Weight  AS-path
10.99.1.2/32  10.1.1.2       100     100     200
10.99.1.3/32  10.1.1.2       100     100     200 10
10.99.1.4/32  10.1.1.2       100     100     200 10 20

host1#show ip bgp quote-regexp ^200
Local router ID 192.168.1.232, local AS 100
  6 paths, 3 distinct prefixes (324 bytes used)
  3 paths selected for route table installation
  7 path attribute entries (872 bytes used)

Prefix        Next-hop  MED  CalPrf  Weight  AS-path
10.99.1.2/32  10.1.1.2       100     100     200
10.99.1.3/32  10.1.1.2       100     100     200 10
10.99.1.4/32  10.1.1.2       100     100     200 10 20
```

If the regular expression contains one or more spaces, you must place quotation marks around the expression in the **show ip bgp quote-regexp** command but not in the **show ip bgp regexp** command. For example, to show all routes whose AS-path contains AS number 10 followed immediately by AS number 20:

```
host1#show ip bgp regexp 10 20
Local router ID 192.168.1.232, local AS 100
  6 paths, 3 distinct prefixes (324 bytes used)
  3 paths selected for route table installation
  7 path attribute entries (872 bytes used)

Prefix         Next-hop   MED   CalPrf  Weight  AS-path
10.99.1.4/32   10.1.1.2         100     100     200 10 20

host1#show ip bgp quote-regexp 10 20
                                    ^
% Invalid input detected at '^' marker.

host1#show ip bgp quote-regexp “10 20"
Local router ID 192.168.1.232, local AS 100
  6 paths, 3 distinct prefixes (324 bytes used)
  3 paths selected for route table installation
  7 path attribute entries (872 bytes used)

Prefix         Next-hop   MED  CalPrf  Weight  AS-path
10.99.1.4/32   10.1.1.2         100     100     200 10 20
```

The **show ip bgp regexp** command accepts multiple strings as arguments. If you try to apply output filtering, the command interprets the filter information as a regular expression and fails:

```
host1#show ip bgp regexp ^200 | begin Prefix
% invalid regular expression
```

Because the **show ip bgp quote-regexp** command accepts only one string as an argument to the regular expression, output filtering is possible:

```
host1#show ip bgp quote-regexp ^200 | begin Prefix
Prefix         Next-hop   MED  CalPrf  Weight  AS-path
10.99.1.2/32   10.1.1.2         100     100     200
10.99.1.3/32   10.1.1.2         100     100     200 10
10.99.1.4/32   10.1.1.2         100     100     200 10 20
```

show ip bgp summary

show bgp ipv6 summary

- Use to summarize the status of all BGP neighbors.
- You can use the field options to display filtered information about BGP neighbors.
- If you filter the display with field options, the usual introductory information about BGP attributes is displayed only if you issue the **intro** fields option. See **show ip bgp** for descriptions of the fields displayed by this keyword.
- You can use the **default-fields peer** command to specify default fields to be displayed by subsequently issued **show ip bgp summary** commands.

- Field descriptions
  - Local router ID—Router ID of the local router
  - Local AS—AS number of local router
  - Administrative state—BGP administrative state, start or stop
  - BGP Operational state—Operational state, up, down, or overload
  - Shutdown in overload state—Status, enabled or disabled
  - Default local preference—Default value for local preference
  - IGP synchronization—Synchronization status, enabled or disabled
  - Default originate—Whether network 0.0.0.0 is redistributed into BGP (enabled) or not (disabled)
  - Auto-summary—Status of auto summarization of routes redistributed into BGP
  - Always compare MED—Status, enabled or disabled
  - Compare MED within confederation—Status, enabled or disabled
  - Advertise inactive routes—Status, enabled or disabled
  - Advertise best external route to internal peer—Status, enabled or disabled
  - Enforce first AS—Status, enabled or disabled
  - Missing MED as worst—Status, enabled or disabled
  - Route flap dampening—Status, enabled or disabled
  - Maximum number of equal-cost EBGP paths—Number of paths
  - Maximum number of equal-cost IBGP paths—Number of paths
  - Log neighbor changes—Status, enabled or disabled
  - Fast External Fallover—Status, enabled or disabled
  - No maximum received AS-path length—Indicates whether limit is set for AS path length and, if set, the limit
  - BGP administrative distances—Distances for external, internal, and local BGP routes
  - Router is a route reflector—Indicates whether the router has been configured as a route reflector
  - Client-to-client reflection—Whether client-to-client reflection is configured (enabled) or not (disabled)
  - Cluster ID—Identifying number for cluster ID
  - Route-target filter—Status, enabled or disabled
  - Default IPv4-unicast—Status, enabled or disabled
  - Redistribution of iBGP routes—Status, enabled or disabled
  - Check reachability of next-hops for VPN routes—Status, enabled or disabled
  - Graceful restart—Status, enabled or disabled
  - Global graceful-restart restart time—Time in seconds

- Global graceful-restart stale paths time—Time in seconds
- Graceful-restart path selection defer time—Time in seconds
- Route Distinguisher—RD assigned to the VRF
- Confederation ID—Confederation ID
- Confederation peers—Confederation peers
- Import route map—Route map associated with the VRF that filters and modifies routes imported to the VRF from the global BGP VPN RIB. The map applies to both IPv4 and IPv6 routes, unless the field name is preceded by IPv4 (applies the map to only IPv4 routes) or IPv6 (applies the map to only IPv6 routes).
- Export route map—Route map associated with the VRF that modifies and filters routes exported by the VRF to the global BGP VPN RIB. The map applies to both IPv4 and IPv6 routes, unless the field name is preceded by IPv4 (applies the map to only IPv4 routes) or IPv6 (applies the map to only IPv6 routes). The can filter routes text appears only if the **filter** keyword was issued for export map.
- Global import route map—Route map associated with the VRF that modifies routes imported to the VRF from the global BGP non-VPN RIB. The map applies to both IPv4 and IPv6 routes, unless the field name is preceded by IPv4 (applies the map to only IPv4 routes) or IPv6 (applies the map to only IPv6 routes).
- routes imported from global table—Number of routes imported from the global BGP non-VPN RIB; also lists the maximum number of routes that can be imported
- Global export route map—Route map associated with the VRF that modifies routes exported by the VRF to the global BGP non-VPN RIB. The map applies to both IPv4 and IPv6 routes, unless the field name is preceded by IPv4 (applies the map to only IPv4 routes) or IPv6 (applies the map to only IPv6 routes).
- Local-RIB version—Number that is increased by one each time a route in that RIB is added, removed or modified.
- FIB version—Number that is increased by one each time BGP updates the routes in the IP routing table based on changes in the local RIB. The FIB version matches the local-RIB version when BGP has finished updating the routes in the IP route table. The FIB version is less than the local-RIB version when BGP is still in the process of updating the IP routing table.
- Neighbor—BGP neighbors
- AS—AS number of the peer
- Ver—Negotiated BGP version number
- State—State of the connection
- Up/down time—Time the connection has been up or down
- Messages sent—Number of messages sent to peer
- Messages received—Number of messages received from peer
- Prefixes received—Number of prefixes received from peer
- Rib Ver—Last RIB version queued to be sent to peer

- Send Q—Number of messages queued to be sent to peer
- More InQ—Status indicating whether any messages are waiting to be sent to peer

- Example 1

```
host1#show bgp ipv6 summary
Local router ID 10.13.13.13, local AS 400
  Administrative state is Start
  BGP Operational state is Up
  Shutdown in overload state is disabled
  Default local preference is 100
  IGP synchronization is disabled
  Default originate is disabled
  Always compare MED is disabled
  Compare MED within confederation is disabled
  Advertise inactive routes is disabled
  Advertise best external route to internal peers is disabled
  Enforce first AS is disabled
  Missing MED as worst is disabled
  Route flap dampening is disabled
  Maximum number of equal-cost EBGP paths is 2
  Maximum number of equal-cost IBGP paths is 2
  Log neighbor changes is disabled
  Fast External Fallover is disabled
  No maximum received AS-path length
  BGP administrative distances are 20 (ext), 200 (int), and 200 (local)
  Client-to-client reflection is enabled
  Cluster ID is 10.13.13.13
  Route-target filter is enabled
  Default IPv4-unicast is enabled
  Redistribution of iBGP routes is disabled
  Graceful restart is globally disabled
  Global graceful-restart restart time is 120 seconds
  Global graceful-restart stale paths time is 360 seconds
  Graceful-restart path selection defer time is 360 seconds
  This platform supports only the receiver role of graceful restart
  Route Distinguisher: 100:11
  Import route map: test2-import-map
  Export route map: test1-export-map (can not filter routes)
  Global import route map: test3-global-import-map
  103 routes imported from global table (max 5000 routes allowed)
  Global export route map: test4-global-export-map
  Local-RIB version 7. FIB version 7.
                                                 Messages  Messages  Prefixes
Neighbor           AS State       Up/down time      Sent  Received  Received
11.11.11.11       400 Established     00:36:19        78        81         2
12.12.12.12       400 Established     00:36:21        78        78         1
103.103.103.3     300 Established     00:36:34        85        80         2
```

- Example 2—Status of next hop reachability checking is displayed only if you specify **vpnv4**.

```
host1#show ip bgp vpnv4 all summary
Local router ID 10.13.5.19, local AS 100
  Administrative state is Start
  BGP Operational state is Up
 ...
  Default IPv4-unicast is enabled
  Redistribution of iBGP routes is disabled
  Check reachability of next-hops for VPN routes is enabled
 ...
```

- Example 3—Status of fields related to enabling local AS numbers to be received in routes

```
host1#show ip bgp summary fields remote-as state rib-version
send-queue-length more-in-queue
                                                Send More
Neighbor           AS State         RIB Ver    Q InQ
2.2.2.2           100 Established         2    0 no
```

show ip community-list

- Use to display routes that are permitted by a BGP community list.
- Example

```
host1#show ip community-list
Community List 1:
  permit 752877569 (11488:1)
  permit 752877570 (11488:2)
  permit 752877571 (11488:3)
  permit 752877572 (11488:4)
Community List 2:
  permit 4294967043 (local-as)
```

undebug ip bgp

- Use to disable the display of information about BGP logs that was previously enabled with the **debug ip bgp** command.
- Example

```
host1#undebug ip bgp
```

- There is no **no** version.