

Configuring BGP Routing

1

This chapter describes how to configure Border Gateway Protocol (BGP) routing on your ERX system.

Topic	Page
Overview	1-2
References	1-11
Features	1-13
Before You Configure BGP	1-13
Configuration Tasks	1-14
Basic Configuration	1-14
Advertising Routes	1-39
Configuring BGP Routing Policy	1-51
Selecting the Best Path	1-87
Interactions Between BGP and IGP	1-110
Configuring BGP Peer Groups	1-120
Managing a Large-Scale AS	1-122
Configuring BGP Multicasting	1-132
Using BGP Routes for Other Protocols	1-135
Configuring BGP/MPLS VPNs	1-135
Monitoring BGP	1-136

Overview

The Border Gateway Protocol (BGP) provides loop-free interdomain routing between 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 1-1 lists the terms and their variant spellings.

Table 1-1 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
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 could 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 1-1, 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.

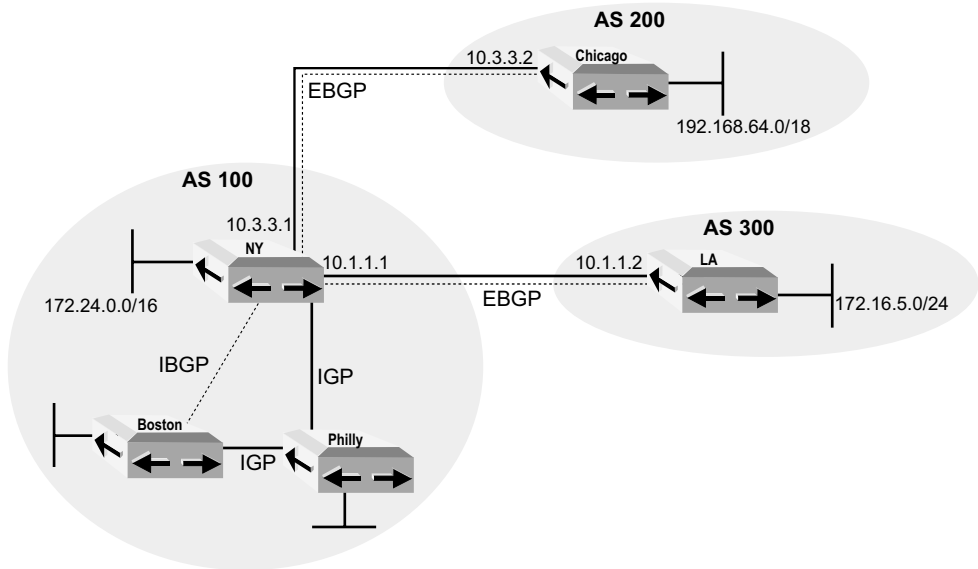


Figure 1-1 BGP peers

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 1-2 shows an example of the exchange of information between routers running IBGP and EBGP across multiple ASs.

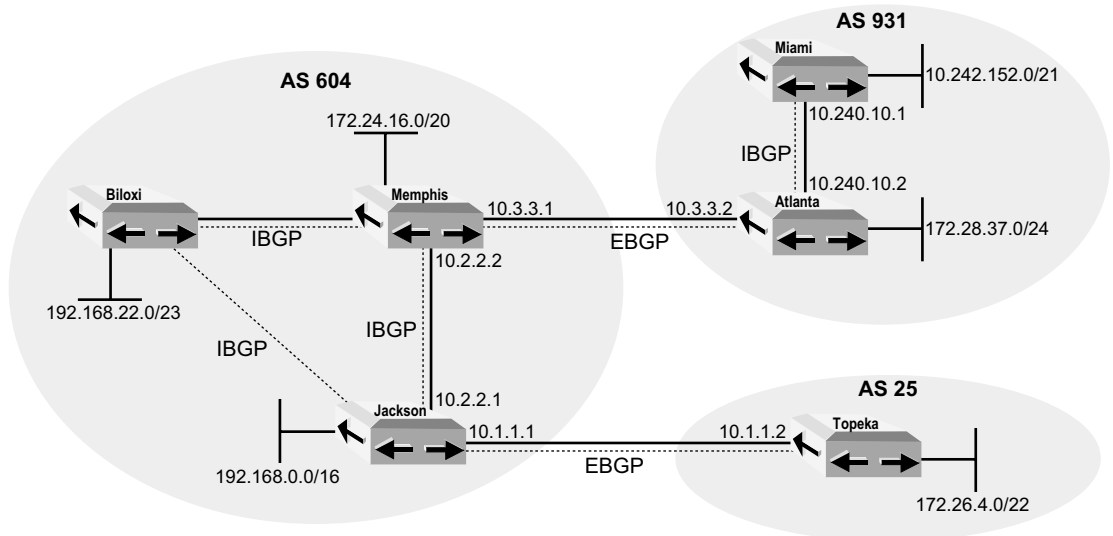


Figure 1-2 Internal and external BGP

Internal 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 internal gateway protocol (IGP) such as the following:

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

Figure 1-3 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.

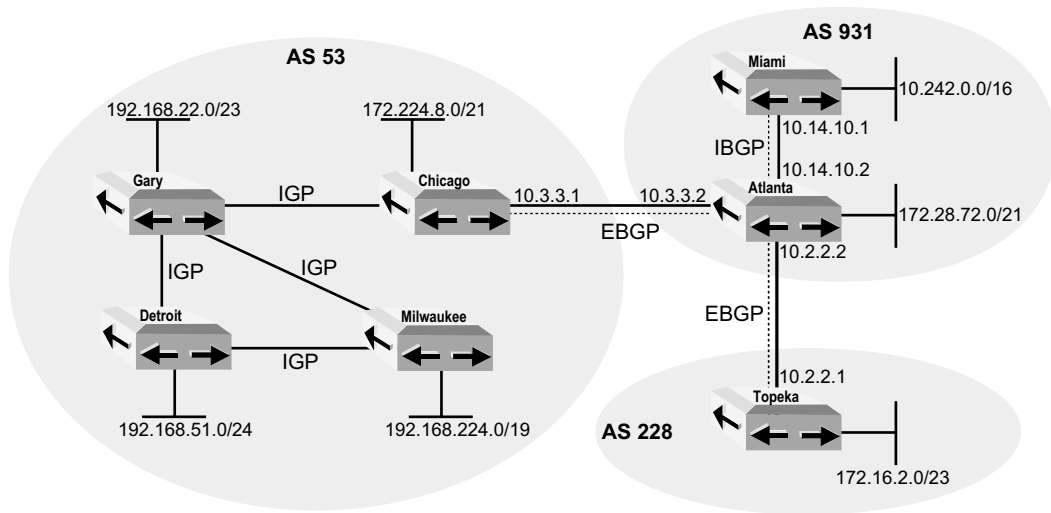


Figure 1-3 Internal gateway protocols

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 check 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.
- 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 system. 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), 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 via 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 would have to advertise a route to each address, as shown in Figure 1-4.

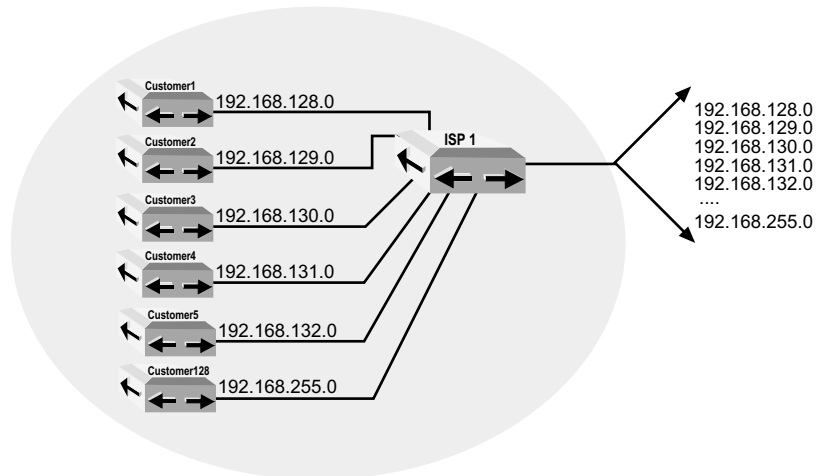


Figure 1-4 Routing without CIDR

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 1-5.

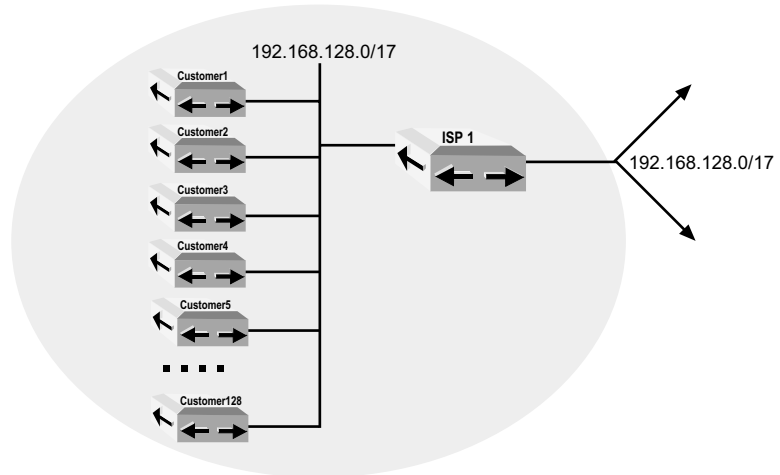


Figure 1-5 Routing with CIDR

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 those routes to use (the “best” route) based on the path attributes. BGP as implemented on the ERX system 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 p 1-87 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 system 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 1-6 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 1-6 illustrates how the ISP permits traffic to transit across its backbone from its own customers or to its own customers.

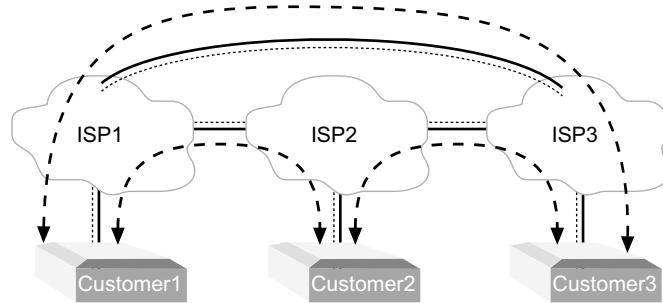


Figure 1-6 Transit service

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

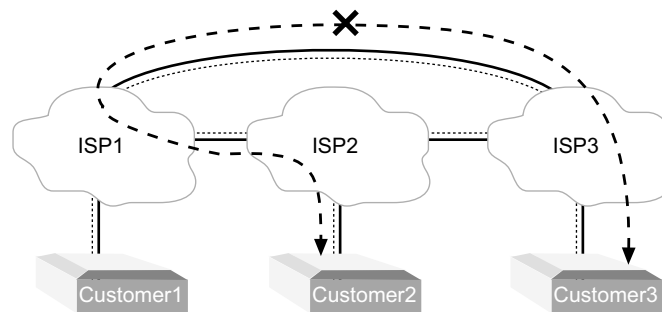


Figure 1-7 Nontransit service

References

If you would like more information about the BGP protocol, consult the following resources:

- Address Prefix Based Outbound Route Filter for BGP-4 – draft-chen-bgp-prefix-orf-04.txt (October 2002 expiration)
- BGP Extended Communities Attribute – draft-ietf-idr-bgp-ext-communities-05.txt (November 2002 expiration)
- BGP/MPLS VPNs – draft-ietf-ppvpn-rfc2547bis-03.txt (April 2003 expiration)
- BGP support for four-octet AS number space – draft-ietf-idr-as4bytes-05.txt (November 2002 expiration)

- Cooperative Route Filtering Capability for BGP-4 – draft-ietf-idr-route-filter-06.txt (November 2002 expiration)
- Dynamic Capability for BGP-4 – draft-ietf-idr-dynamic-cap-02.txt (October 2002 expiration)
- *ERX Release Notes, Appendix A, System Maximums* – refer to the Release Notes corresponding to your software release for information on maximum values.
- 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 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 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 3065 – Autonomous System Confederations for BGP (February 2001)



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

The system supports the following BGP features:

BGP-4	Soft-reconfiguration inbound
Communities	Update source
Confederations	Route mapping and attribute manipulation
BGP neighbor setup	Route redistribution
EBGP multihop	Route reflectors
Next-hop self	Route dampening (also referred to as route damping)
Advertisement intervals	Synchronization enabling and disabling
BGP route origins	BGP aggregation
BGP peer groups	Highly scalable BGP architecture to support hundreds of BGP peers and hundreds of thousands of routes
BGP access lists	BGP/MPLS VPNs
BGP Multicast	

Before You Configure BGP

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

Refer to the *ERX Physical and Link Layers Configuration Guide* for information on how to configure appropriate interfaces. Refer to *ERX*

Routing Protocols Configuration Guide, Vol. 1, Chapter 1, Configuring Routing Policy, for information on 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.

Topic	Page
Basic Configuration	1-14
Advertising Routes	1-39
Configuring BGP Routing Policy	1-51
Selecting the Best Path	1-87
Interactions Between BGP and IGP	1-110
Configuring BGP Peer Groups	1-120
Managing a Large-Scale AS	1-122
Configuring BGP/MPLS VPNs	1-135

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.
- Example

```
host1(config)#router bgp 100
```
- This command takes effect immediately.
- 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:

- The commands listed in Table 1-2 configure parameters for the BGP process globally, regardless of address family.

Table 1-2 Commands affecting BGP globally

bgp advertise-inactive	bgp log-neighbor-changes
bgp always-compare-med	bgp maxas-limit
bgp bestpath med confed	bgp redistribute-internal
bgp bestpath missing-as-worst	bgp router-id
bgp client-to-client reflection	bgp shutdown
bgp cluster-id	ip bgp-community new-format
bgp confederation identifier	maximum-paths
bgp confederation peers	overload shutdown
bgp default local-preference	rib-out disable
bgp enforce-first-as	router bgp
bgp fast-external-falover	timers bgp

- The commands listed in Table 1-3 configure parameters for all address families within the current VRF context.

Table 1-3 Commands affecting all address families in a VRF

distance bgp	synchronization
--------------	-----------------

- The commands listed in Table 1-4 configure parameters only for the current address family context.

Table 1-4 Commands affecting the current address family

address family	ip route-type
aggregate-address	network
bgp dampening	redistribute
default-information originate	table-map
disable-dynamic-redistribute	

- The commands listed in Table 1-5 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, it is not configurable per address family:

```

host1(config-router)#neighbor 1.1.3.4 remote-as 1234
host1(config-router)#neighbor 1.2.3.4 ebgp-multihop 5
host1(config-router)#address-family ipv4 multicast
host1(config-router-af)#neighbor 1.2.3.4 ebgp-multihop ?
                                     ^
% Invalid input detected at '^' marker.
host1(config-router-af)#exit-address-family

```

Table 1-5 Commands affecting all address families for the specified peer or peer group

neighbor advertisement -interval	neighbor remote-as
neighbor capability	neighbor rib-out disable
neighbor description	neighbor shutdown
neighbor ebgp-multihop	neighbor timers
neighbor maximum-update-size	neighbor update-source
neighbor password	neighbor weight

- The commands listed in Table 1-6 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 1-6 Commands affecting only the current address family for the specified peer or peer group

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

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 1-7, 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 to the **default** version.

Table 1-7 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
<ul style="list-style-type: none"> neighbor activate neighbor as-override neighbor ebgp-multihop neighbor next-hop-self neighbor remove-private-as neighbor route-reflector-client neighbor send-community neighbor soft-reconfiguration inbound 	<ul style="list-style-type: none"> neighbor default-originate^a neighbor rib-out disable^b neighbor shutdown^c 	<ul style="list-style-type: none"> neighbor advertisement -interval neighbor allowas-in neighbor description neighbor distribute-list neighbor filter-list neighbor maximum-prefix neighbor maximum-update-size neighbor password neighbor prefix-list neighbor prefix-tree neighbor remote-as neighbor route-map neighbor unsuppress-map neighbor update-source neighbor weight 	<ul style="list-style-type: none"> neighbor timers

a. The **neighbor default-originate** command inherits global values set by the **default-information originate** command.

b. The **neighbor rib-out disable** command inherits global values set by the **rib-out disable** command.

c. The **neighbor shutdown** command inherits global values set by the **bgp shutdown** command.

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:

neighbor as-override	neighbor next-hop-self	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



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 system 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 system's 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 1-8. Routers LA and SanJose are IBGP peers within AS 873. Router SanJose has an EBGp peer, router Boston, in AS 17.

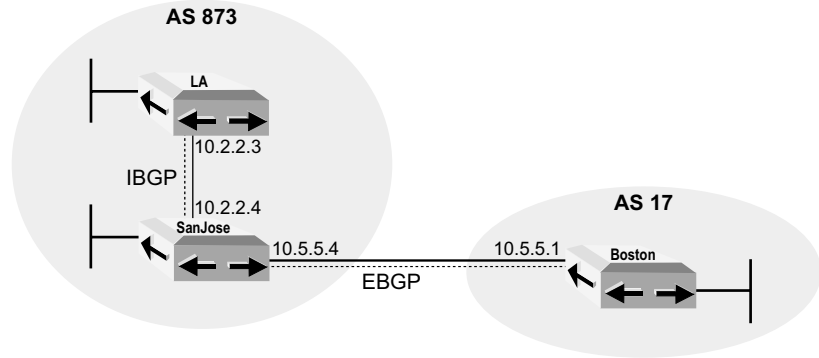


Figure 1-8 Configuring neighbors

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 considered external.
- 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.

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

- Associates 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.

- Example

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

- This command takes effect immediately.
- 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.
- 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

```

- This command takes effect immediately.
- 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.
- Use the **no** version to restore the interface assignment to the closest interface.

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 1-9, router Boston and router LA are connected together via router NY, rather than by a direct connection. Routers Boston and LA are configured as external peers with the **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.

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
```

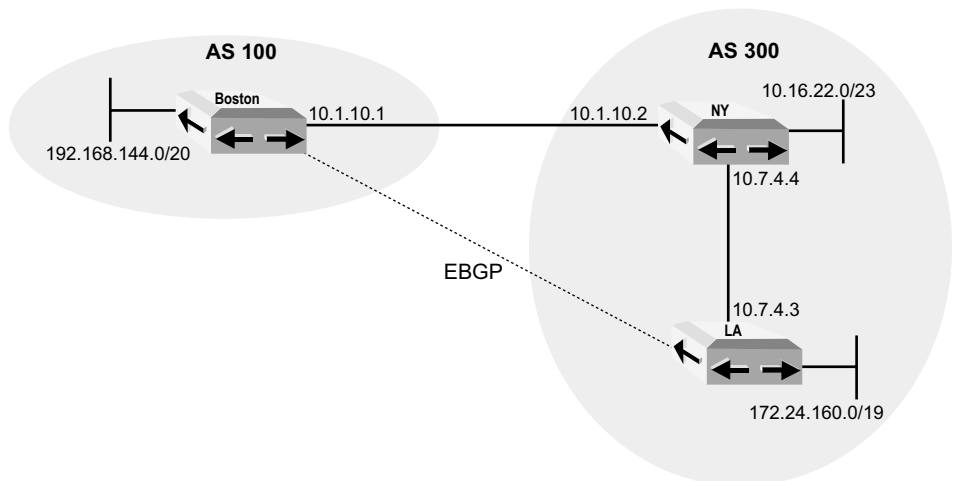


Figure 1-9 Using EBGP-multihop

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.
- 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.

Controlling the Number of Prefixes

As your 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 should not include the private AS numbers in the path. Configure your 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 RIB, 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 RIB and therefore is displayed via 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 RIB, 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 system does not make the connection between the BGP peers.

The MD5 authentication feature uses the MD5 algorithm. When you specify this command, the system 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 Memory Limits for BGP Tables

You can use the **limits** command to configure the memory limits of the internal BGP tables stored by the system. We recommend that you do not change these limits unless absolutely necessary. In most BGP environments, modifying the limits is unnecessary.

You can set the limits for one or more of the following BGP tables:

Address Family	Peer
Aggregated Routes	Peer Address Family
BGP Destinations	Peer Group

Dampening	Peer Group Address Family
Network Layer Reachability Entry	Received Routes
Network Layer Reachability Information	Redistributed Routes
Network Routes	RIB-Out
Next Hop	Route Flap History
Path Attributes	VRF

limits

- Use to set memory limits for the BGP internal tables maintained by BGP software.
- Specify any limit as an integer in the range 0–214783648.
- All limits have a default value of 5000000, except the **rib-out** limit, which defaults to 1000000.
- Example


```
host1(config-router)#limits vrf 1250000
host1(config-router)#limits next-hop 7500000
```
- This command takes effect immediately.
- Use the **no** version to return the memory limits to their default values.

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 via 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.
- Example

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

- 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.
- 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.

bgp shutdown

- Use to shut down BGP globally.
- Example

```
host1(config-router)#bgp shutdown
```
- This command takes effect immediately.
- 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.

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

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.

- Example

```
host1(config-router)#overload shutdown
```
- This command takes effect immediately.
- 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 only has to apply outbound policy 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 system 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 only sends the update message if the new information is different than 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 does not happen for peer groups, because their Adj-RIBs-Out tables stores 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* later in this chapter for information on 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 RIB. 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 only change outbound policy 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 contains all routes actually sent to the peer. In contrast, if you enable storage for a peer group, the peer group's Adj-RIBs-Out 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.
- Example

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

- This command takes effect immediately and automatically bounces the BGP session(s) if the command changes the current configuration.
- 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.
- Example

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

- This command takes effect immediately and automatically bounces the BGP session if the command changes the current configuration.
- 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*. Our BGP implementation defines three different types of address families:

- Unicast IPv4 – If you do not explicitly specify the address family, the system employs 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 a general description of multicasting, see *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 3, Configuring IP Multicasting*.
- VPN IPv4 – If you specify the VPN-IPv4 address family, you can configure the system to provide IPv4 VPN services via an MPLS backbone. These VPNs are often referred to as BGP/MPLS VPNs. For detailed information, see *Chapter 3, Configuring BGP/MPLS VPNs*.

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 Specify an address family within which the router should exchange addresses.
- 2 Specify individual neighbors or peer groups that will exchange routes only from within the current address family.
- 3 Configure BGP parameters for the address family.
- 4 Exit Address Family Configuration mode.

address-family

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

```
host1:vr1(config-router)#address-family ipv4 multicast
host1:vr1(config-router)#address-family vpnv4 vrf vr2
host1:vr1(config-router)#address-family ipv4 unicast
```
- Creating an address family for a VRF automatically disables both synchronization and automatic summarization for that VRF.
- This command takes effect immediately.
- 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 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.
- Example

```
host1:vr1(config-router-af)#neighbor 192.168.1.158 activate
```
- 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 RIB of the newly activated address family.
- Use the **no** version to indicate that routes of the current address family should not 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.

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. Refer to *Understanding the Origin Attribute* (p 1-99) for more information on origins. Figure 1-10 shows a network structure of three autonomous systems, each with a router that originates certain prefixes.

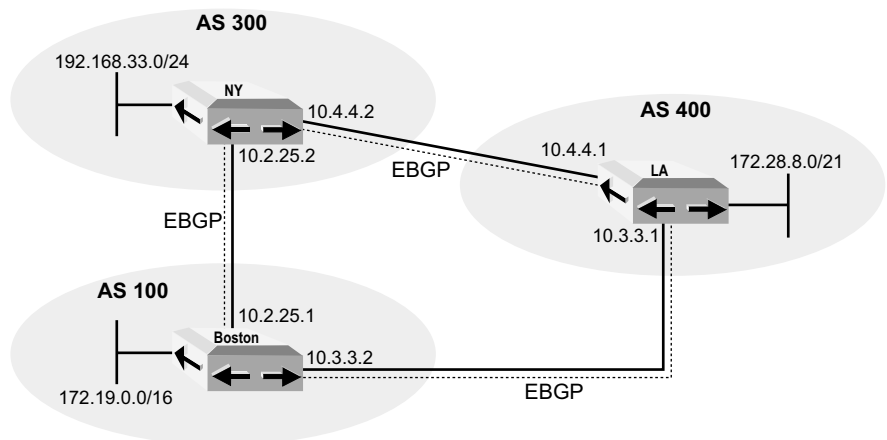


Figure 1-10 Prefixes originating in an AS

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.
- Specify a *network-number* and an optional *network-mask*.
- 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* (p 1-118) 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.

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. Once redistributed, BGP advertises the routes. When you redistribute routes, BGP sets the origin attribute for the route to

Incomplete. Refer to *Understanding the Origin Attribute* (p 1-99) for more information on 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 1-11:

```
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
```

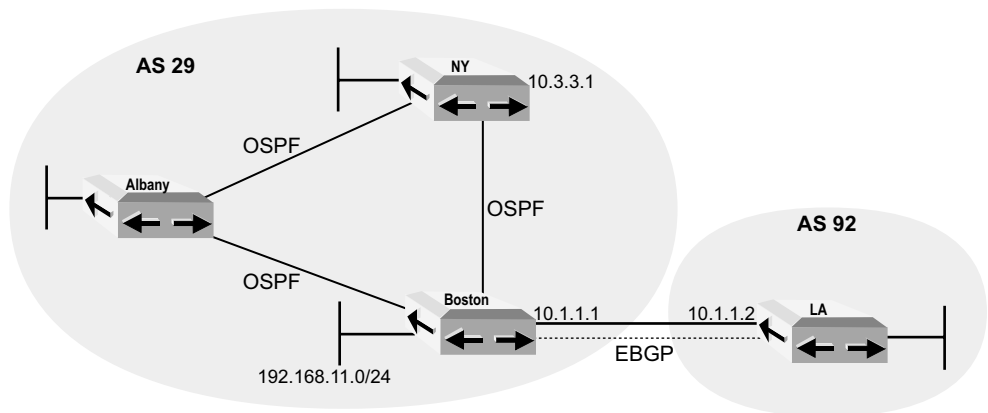


Figure 1-11 Redistributing routes into BGP

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.
- Example

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

- This command takes effect immediately.
- Use the **no** version to reenale 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 1-12, router NY originates the default route 0.0.0.0/0 to router Albany only. Router Chicago does not receive the 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
```

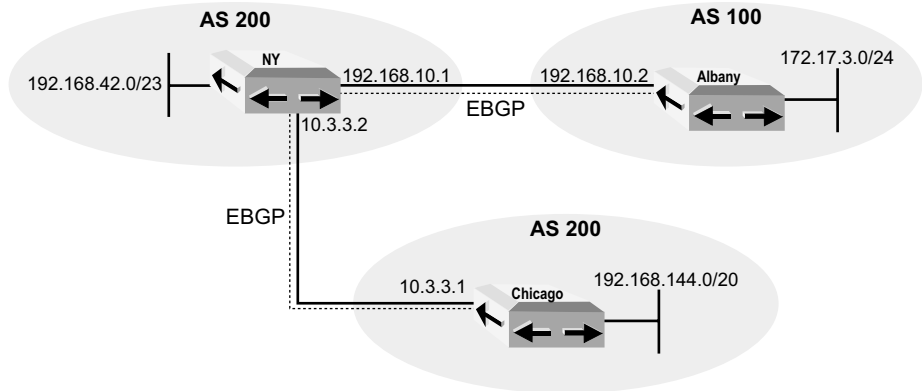


Figure 1-12 Advertising a default route

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.
- Example


```
host1(config)#router bgp 100
host1(config-router)#default-information originate
```
- This command takes effect immediately.
- 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 1-13, 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
```

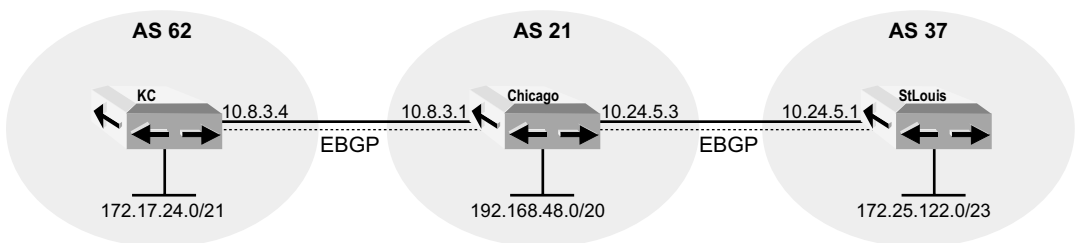


Figure 1-13 Setting a static default route

ip route

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

neighbor default-originate

- Use to allow 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.
- Outbound route maps are not applied to this 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.
- 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 Sending 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 2.2.2.2
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. Figure 1-14 illustrates a network structure where you might use aggregation.

The following commands configure router LA and router SanJose so that router SanJose advertises an 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.

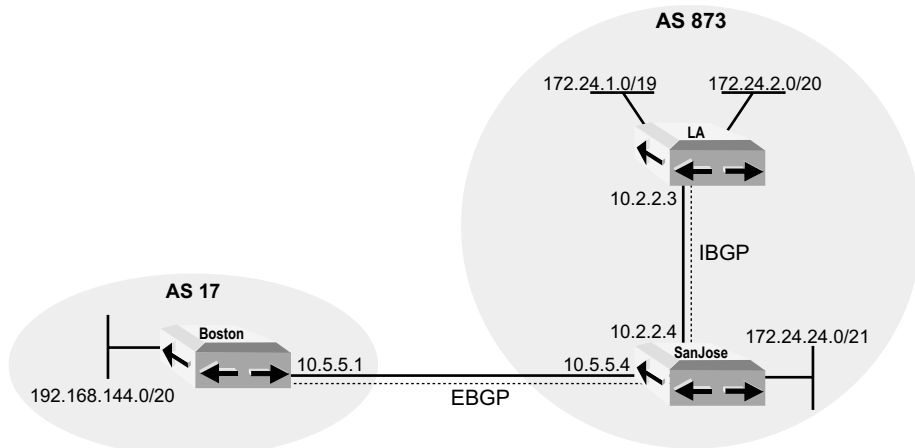


Figure 1-14 Configuring aggregate addresses

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 should 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 1-14 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 1-14, 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 would not be 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.
- You must specify an aggregate IP address (*address*) and aggregate IP mask (*mask*).
- 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 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 RIB, 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.

Configuring BGP Routing Policy

Routing policy determines how the system 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 system.

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

Access lists	Prefix trees
Community lists	Route maps
Prefix lists	

The remainder of this section provides detailed information on using these features with BGP. Before proceeding, please see *ERX Routing Protocols Configuration Guide, Vol. 1, 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 1-8.

Table 1-8 Commands that create match-and-set route maps

aggregate-address attribute-map	redistribute route-map
bgp dampening route-map	table-map
neighbor route-map in	vrf import-map

Table 1-8 Commands that create match-and-set route maps (continued)

neighbor route-map out	vrf export-map
------------------------	----------------

BGP supports the clauses listed in Table 1-9 for match-and-set route maps.

Table 1-9 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
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 1-10. You can use any of the match clauses listed in Table 1-9 in these route maps. Set clauses have no effect in these route maps.

Table 1-10 Commands that create match-only route maps

aggregate advertise-map	aggregate support-map
-------------------------	-----------------------

BGP does not support the clauses listed in Table 1-11. However, see the section later in this chapter, Applying Table Maps, for exceptions for route maps applied via the **table-map** command.

Table 1-11 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-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.
- 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 RIB, 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 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.
- 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 nycl 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 nycl 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 nycl 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 nycl 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.
- 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 nycl 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 nycl 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 nycl 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 range from 0 to 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 nycl 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. You can use only the set clauses listed in Table 1-12 in these route maps.

Table 1-12 Set clauses supported in route maps applied via the **table-map** command

set distance	set metric-type
set ip next-hop	set route-type
set level	set tag
set metric	

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 1-12.
- 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 ip access-list** and **show config** 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 1-15.

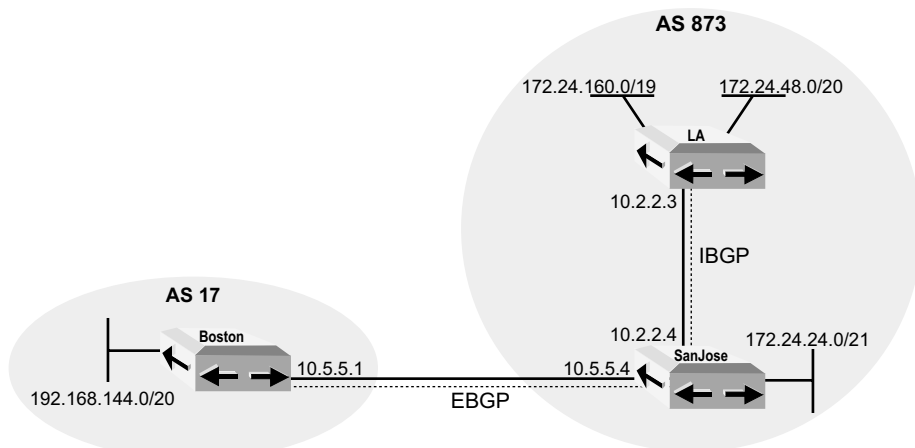


Figure 1-15 Filtering with access lists

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 1-16. 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`.

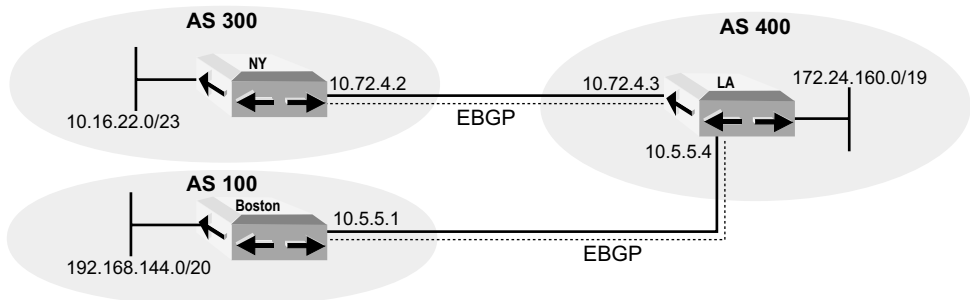


Figure 1-16 Filtering routes with an access list

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 RIB, 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 RIB, 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 RIB, 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 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 ERX Routing Protocols Configuration Guide, Vol. 1, 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 1-17.

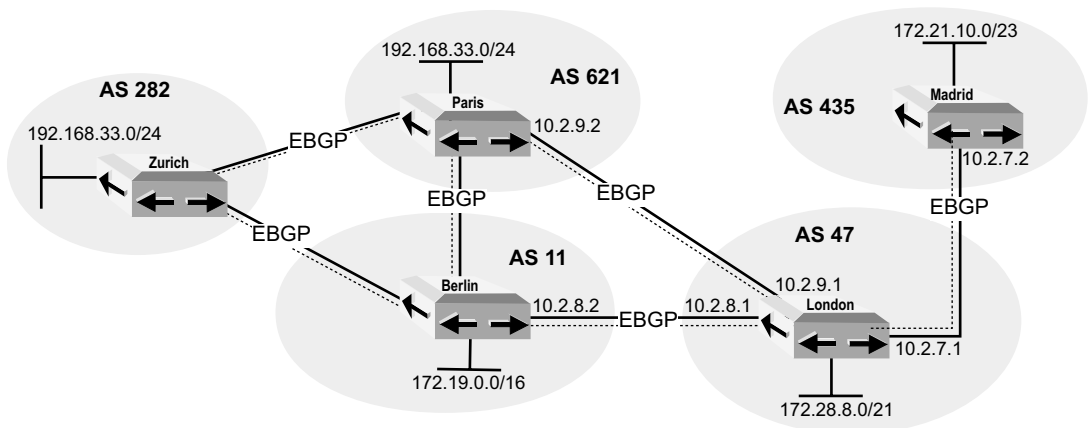


Figure 1-17 Filtering with AS-path access lists

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 1-18:

```
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$
```

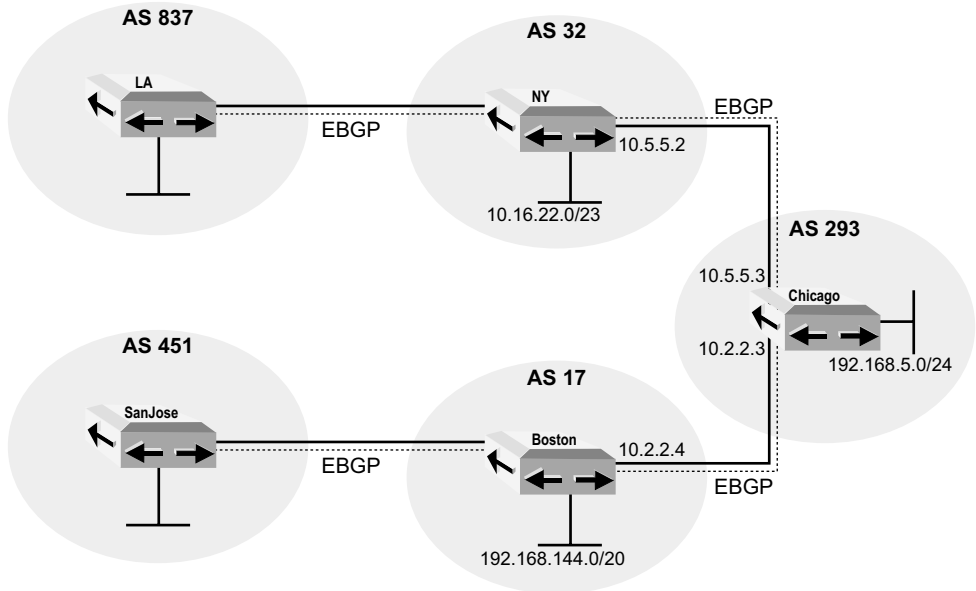


Figure 1-18 Assigning a filter list

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 RIB, 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 1-19, 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
```

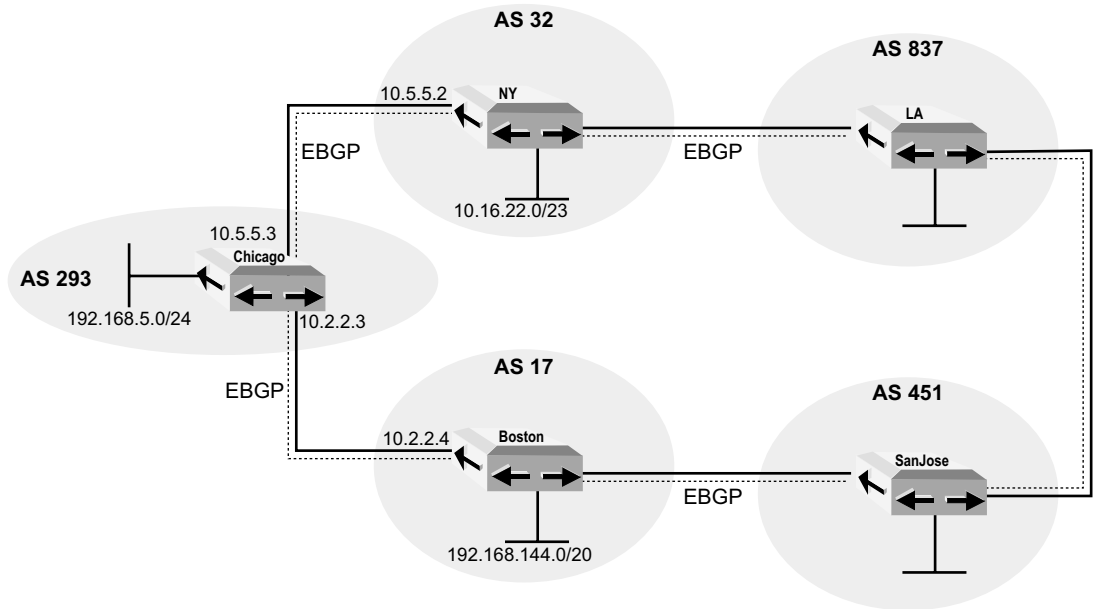


Figure 1-19 Route map filtering

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

Refer to the commands and guidelines beginning on p 1-53 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 1-13 describes how a BGP speaker handles a route based on the setting of its community attribute.

Table 1-13 Action based on well-known community membership

If the well-known community is...	The BGP speaker...
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 could 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 1-20. The following sample configurations illustrate some of the capabilities of using the community attribute.

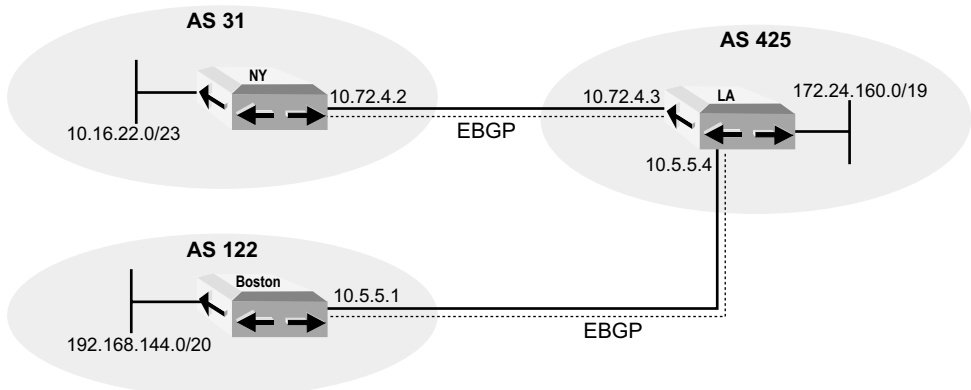


Figure 1-20 Communities

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 via 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 should 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 RIB, 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 nycl 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 1-21.

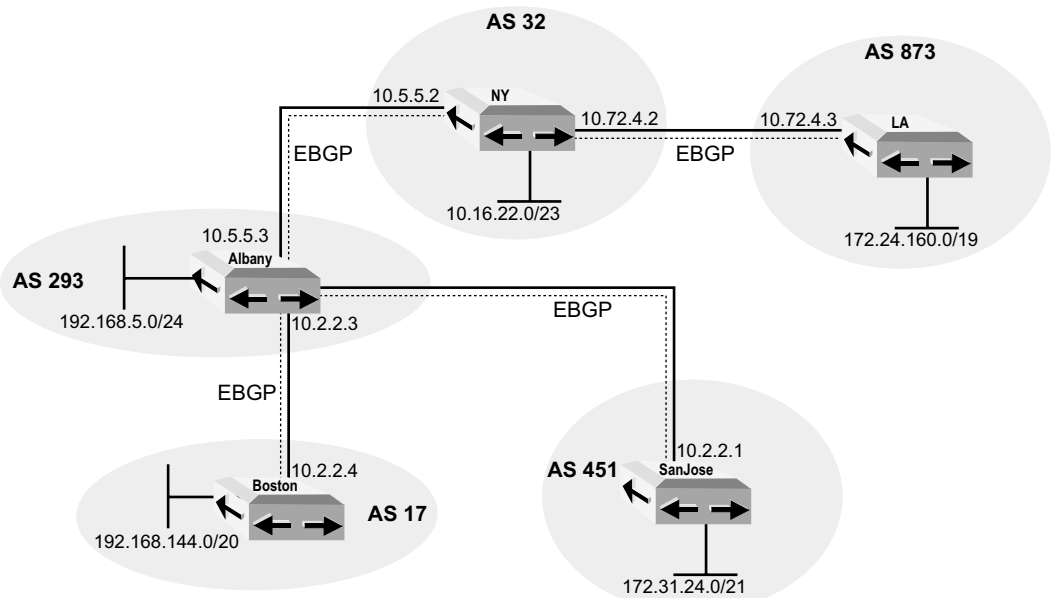


Figure 1-21 Community lists

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

- Creates 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 system 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-05.txt (November 2002 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** command 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 **clear ip bgp** 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 reapplied 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.

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-list** 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 system 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 system 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 system 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 system. 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 system 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.

- Example

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

- When issued with the **orf** keyword, this command takes effect immediately and automatically bounces the BGP session.
- Use the **no** version to prevent advertisement of 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 RIB, 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 system stores a penalty value with each route. Each time the route flaps, the system increases the penalty by 1000. If the penalty for a route reaches a configured *suppress* value, the system suppresses the route. That is, the system 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 system once again advertises the route. You can specify a *max-suppress-time* for route suppression; once this interval passes, the system 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 system 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* – when this period expires, the penalty assigned to a route is decreased by half

- › *reuse* – when 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 via 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 ip bgp dampening

- Use to clear 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.
- Examples

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

```
host1#clear ip bgp dampening
```

To clear dampening information for all routes in VRF dogwood:

```
host1#clear ip bgp vrf dogwood dampening
```

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

```
host1#clear ip bgp ipv4 unicast dampening
```

To clear dampening information for a specific route:

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

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

```
host1#clear ip bgp dampening 192.168.5.0
```

- This command takes effect immediately.
- 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** command to test the outcome of a BGP policy. The command output displays the routes that would be advertised or accepted if the specified policy were 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 (via the **neighbor soft reconfiguration in** command) must be in effect.



Note: The output of the **test ip bgp neighbor** command is always speculative. It does not reflect the current state of the system.

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 system.



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 *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*.
- 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.
- 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 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.

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.

The 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 via EBGP over one learned via IBGP.
- 9 Select the route with the lowest IGP cost to the next hop.
- 10 Select the route received from the peer with the lowest BGP router ID.

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. Systems 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 via 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 1-22. 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.

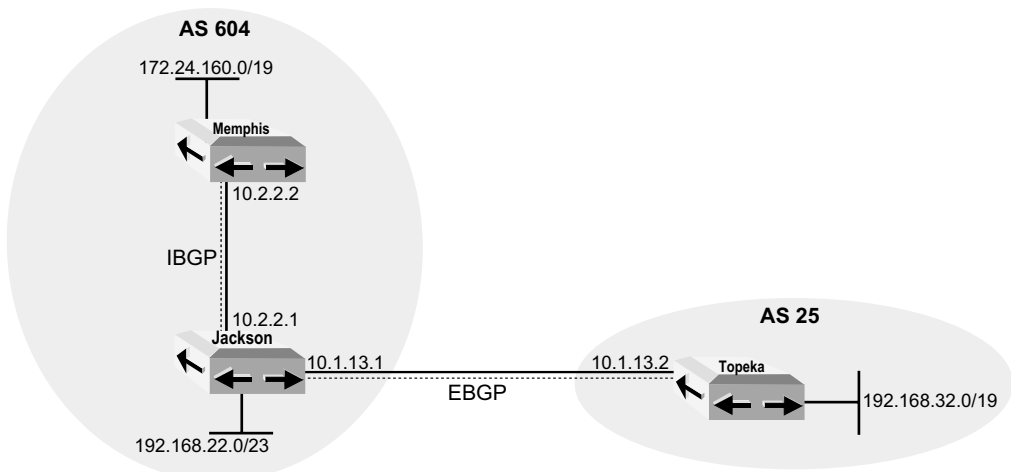


Figure 1-22 Configuring next-hop processing

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 via 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 via router Jackson, and the traffic can be forwarded.

The following commands configure the routers as shown in Figure 1-22:

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 considered external.
- 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 1-23.

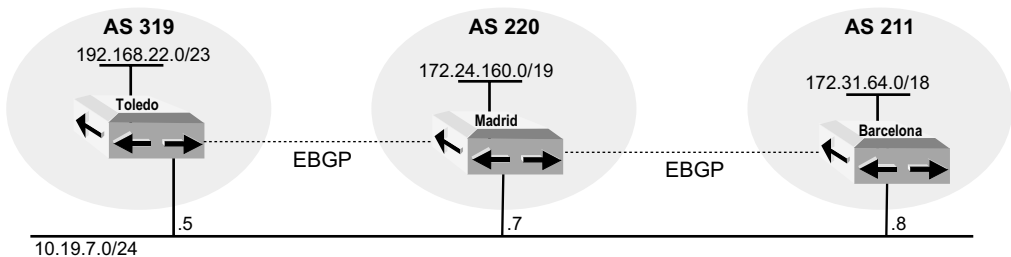


Figure 1-23 Next-hop behavior for broadcast multiaccess media.

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 1-24, which shows the same routers on a Frame Relay—NBMA—network.

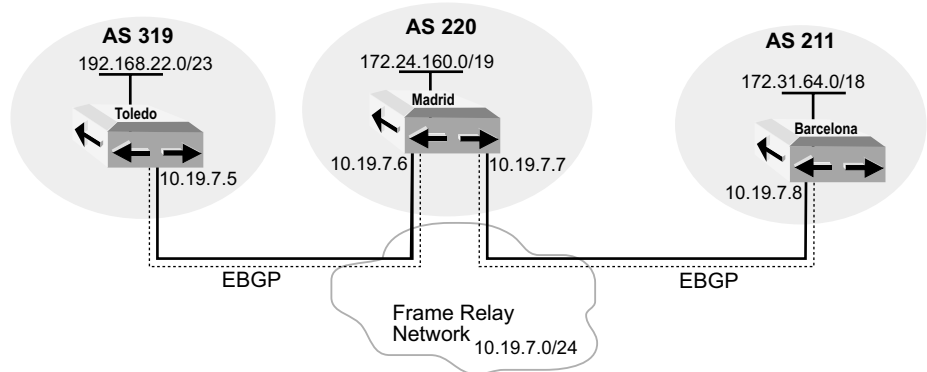


Figure 1-24 Next-hop behavior for nonbroadcast multiaccess media.

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 RIB, 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 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 1-25, 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.

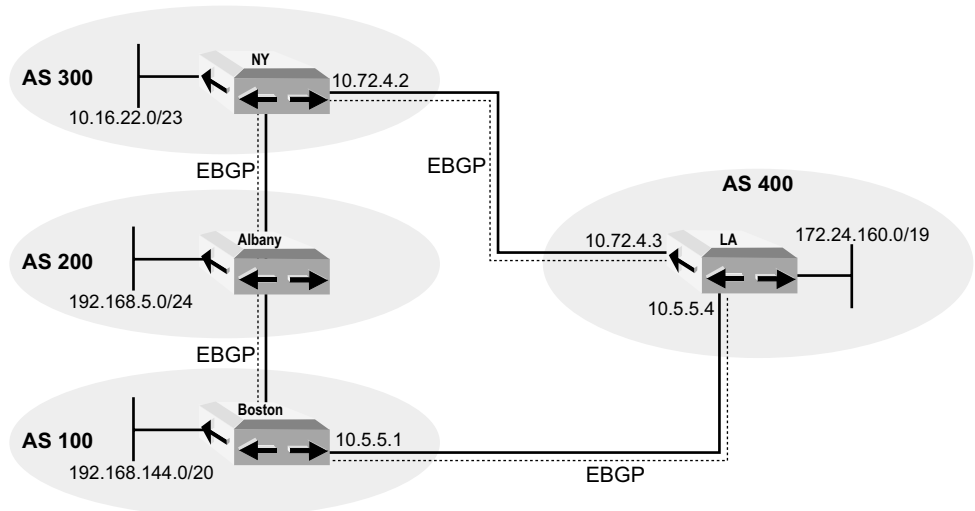


Figure 1-25 Assigning a weight to a neighbor connection

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 *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 1, Configuring Routing Policy* for more information on 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

- Applies an AS-path access list to advertisements inbound from or outbound to the specified neighbor, or assigns 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 RIB, 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 RIB, 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* (p 1-62) for more information on 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 1-26, AS 873 receives updates for network 192.168.5.0/24 from AS 32 and AS 17.

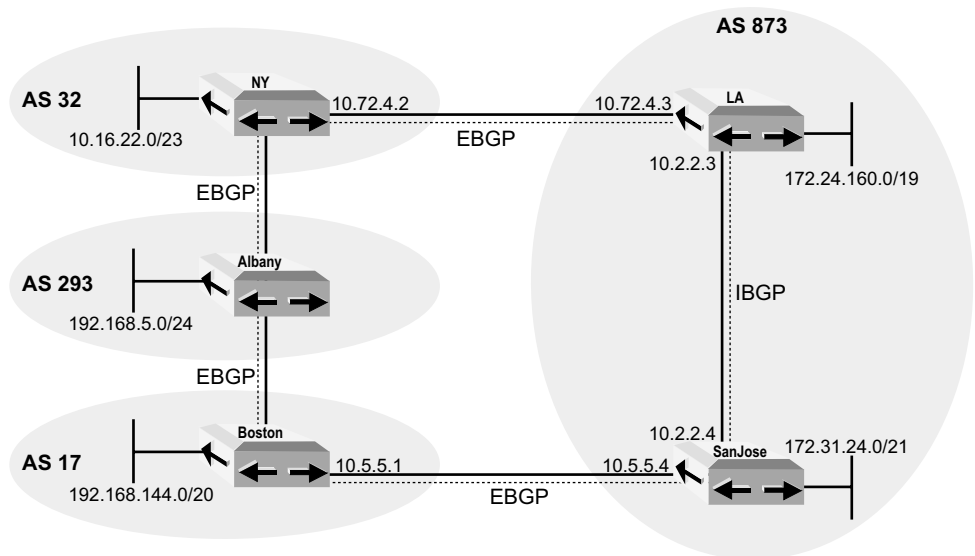


Figure 1-26 Configuring the local-preference attribute

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 via 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 via an IGP and, therefore, is internal to the originating AS. All routes advertised via the **network** command have an origin of IGP.
- EGP – indicates that the route was learned via 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 via the **redistribute** command—such as static routes—have an origin of Incomplete. An origin of Incomplete occurs when a route is redistributed into BGP.

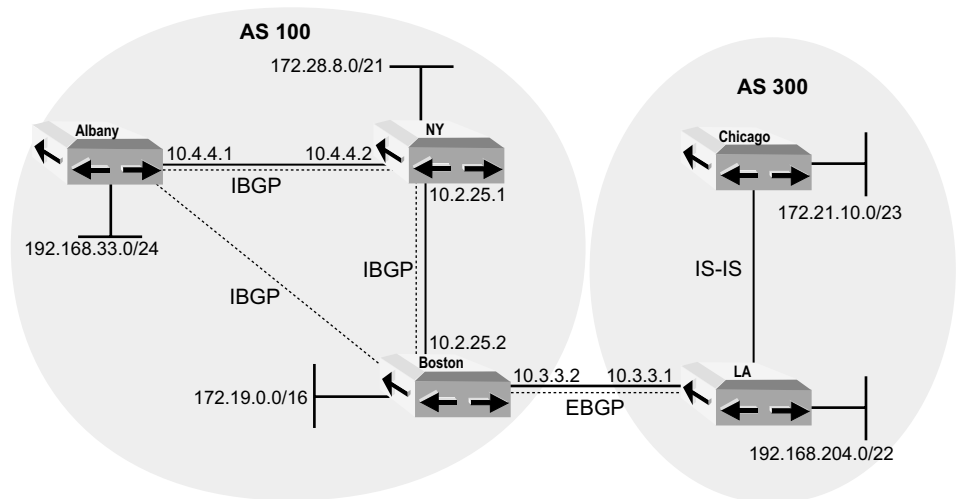


Figure 1-27 The origin attribute

Consider the sample topology shown in Figure 1-27. Because routers Albany and Boston are not directly connected, they learn the path to each other via 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 1-27:

- 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 via EBGP from router LA.
- 3 Router NY learns about route 172.21.10.0/23 via IBGP from router Boston.

The value of the origin attribute for a given route remains the same, regardless of where you examine it. Table 1-14 shows this for all the routes known to routers NY and LA.

Table 1-14 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

Table 1-14 Origin and AS-path for routes viewed on different routers (continued)

Route	Router	Origin	AS Path
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 systems shown in Figure 1-28:

- 1 Route 172.21.10.0/23 is injected into BGP via 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.

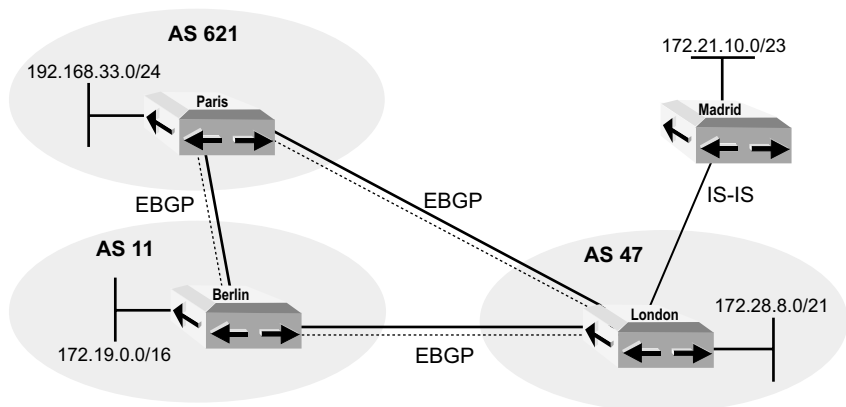


Figure 1-28 AS-path attributes

A routing loop would exist 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 tell 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 1-29, 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.

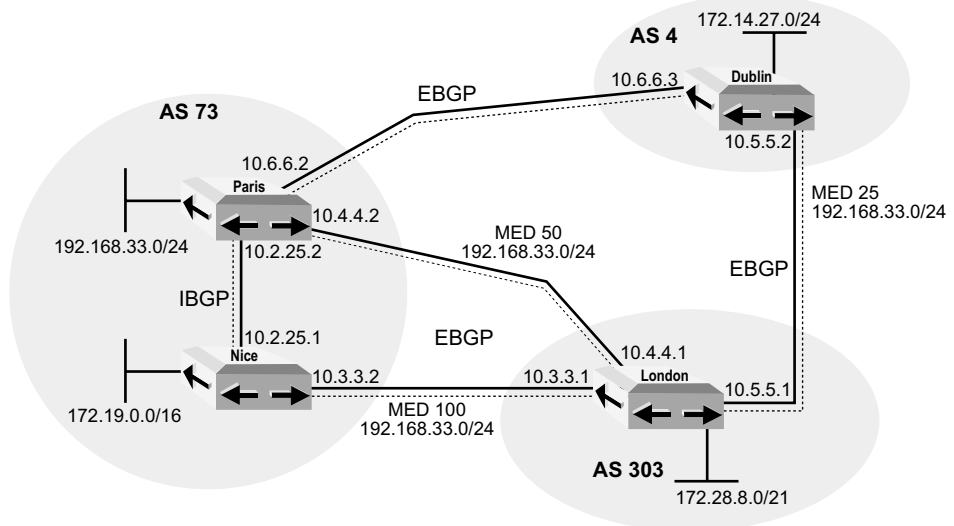


Figure 1-29 Configuring the MED

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 consider 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 nycl 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 considered to have originated in different places.) You can use the **bgp bestpath med confed** command to force the consideration of MED values within a confederation.

bgp bestpath med confed

- Use to specify that BGP consider 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 to the **no** version to restore the default, where the MED is not considered.

Example 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 considers the MED when comparing Route 1 with Route 2. However, BGP does not consider the MED when comparing Route 3 with either Route 1 or Route 2, because Route 3 originates outside the confederation.

Capability Negotiation

The system accepts connections from peers that perform capability negotiation. Capabilities are negotiated using the open messages that are exchanged when the session is established. The system supports the following capabilities:

- Cisco-proprietary route refresh – capability code 128
- Cooperative route filtering– capability code 3
- Dynamic capability renegotiation – capability code 66
- Four-octet AS numbers – capability code 65
- 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 system 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.

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 system 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 system 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.

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.

Route Refresh

If the system 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 is enabled by default.
- You can specify the **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.
- Example

```
host1(config-router)#neighbor 10.6.2.5 capability orf
prefix-list both
```
- If you issue the **route-refresh** or **route-refresh-cisco** keywords, 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 **dynamic-capability-negotiation**, **four-octet-as-numbers**, or **negotiation** keywords, takes effect immediately and bounces the session.
- If you issue the **orf** keyword, takes effect immediately and automatically bounces the BGP session.
- 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.

Interactions Between BGP and IGPs

Interactions between BGP and an internal gateway protocol are more likely to occur in an enterprise system than in a service provider system. You can also encounter interactions when configuring small test systems. 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 1-30, 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.

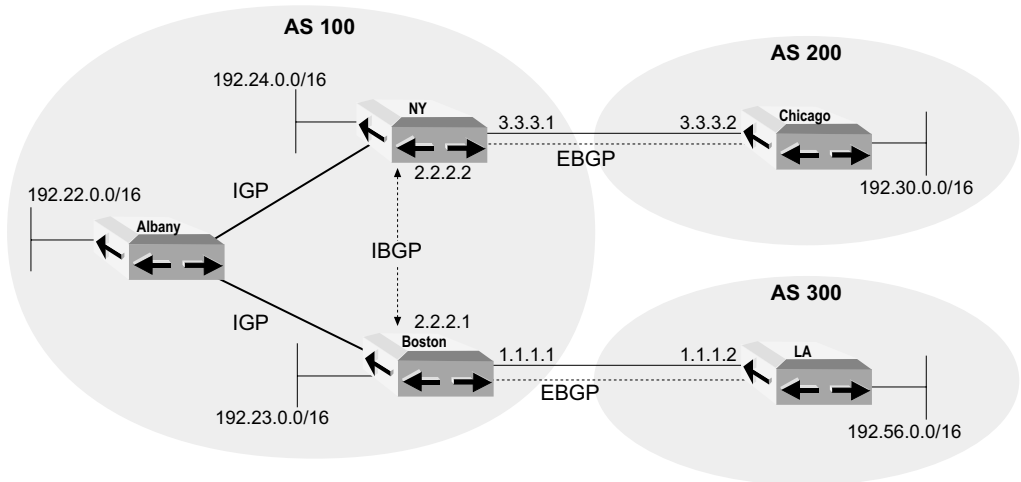


Figure 1-30 Synchronization

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 via 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. In 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 1-30, 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 via EBGP to router Chicago.

In practice, service providers rarely redistribute BGP into an IGP because existing IGP's cannot handle the full Internet routing table (about 100,000 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 via 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 1-31 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.

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.

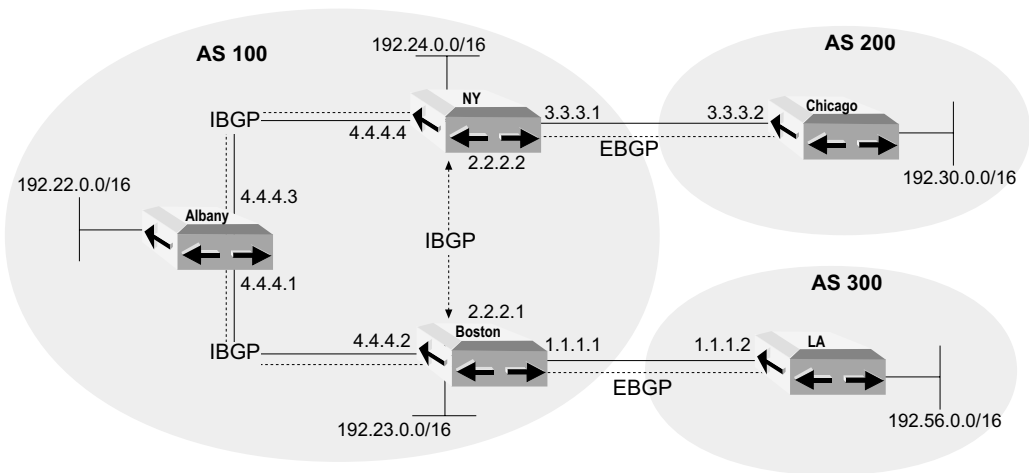


Figure 1-31 Disabling synchronization

The following commands show how to configure routers Boston, NY, and Chicago (shown in Figure 1-31) 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 via 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 Chicago without learning about 192.30.00/16 via 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 ranging from 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 1-15, default distances are provided for each type of source from which a route can be learned.

Table 1-15 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* (p 1-49) 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 via 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 via BGP. Consider the network structure shown in Figure 1-32.

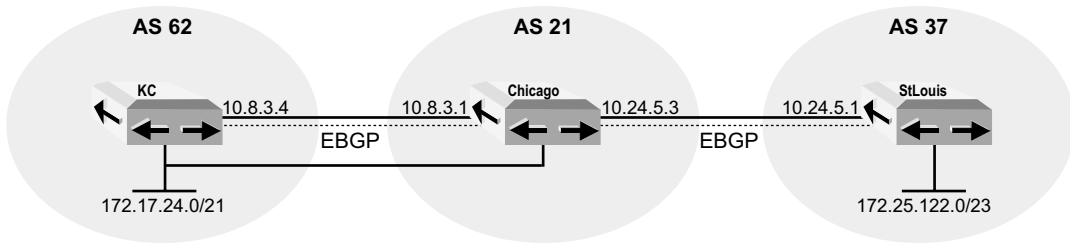


Figure 1-32 Administrative distances

Suppose router KC originates 172.17.24.0/21 and advertises the route to router Chicago via 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 via 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 1-33. Router Chicago originates prefix 192.168.11.0/24 and advertises it via EBGP to router Albany. Router Albany advertises the route to router Boston via IBGP.

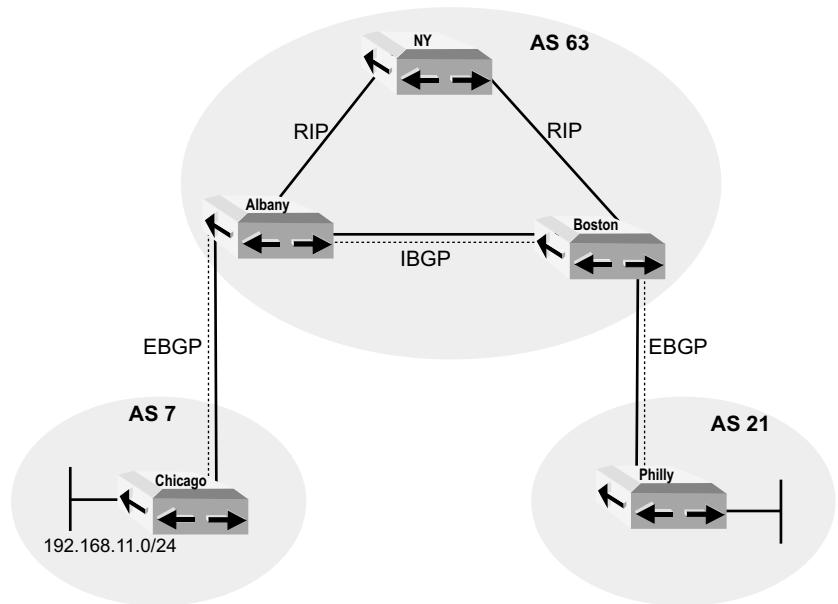


Figure 1-33 Administrative distance and synchronization

Router Albany also redistributes the route into the internal gateway protocol, RIP, which informs router NY of the route. Router NY propagates the route to router Boston via 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.

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 via any IGP appears in the IP routing table. In other words, routers connected via 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 via an IGP protocol. Consider the network structure shown in Figure 1-34.

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 via OSPF from router Boston, and one via EBGP from router LA through router San Diego. As was shown in Table 1-15, 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 via EBGP is preferred over the OSPF backdoor path with its distance of 110.

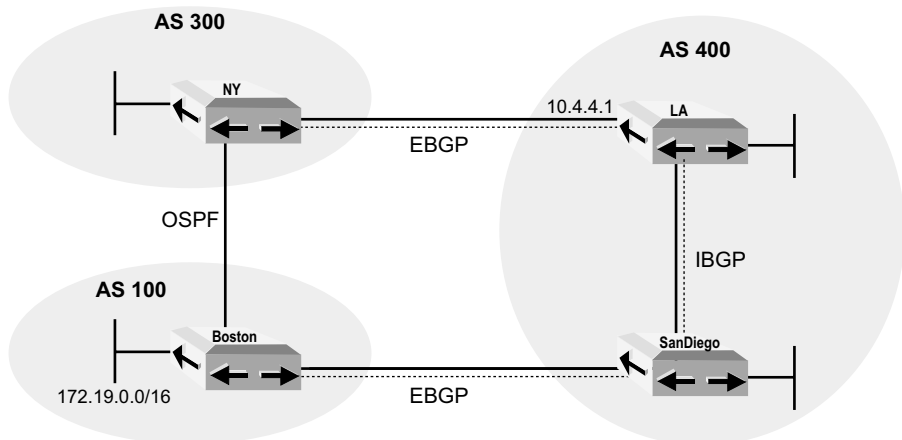


Figure 1-34 Backdoor route

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 via 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 1-34, 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.
- Example

```
host1(config-router)#network 10.53.42.0 backdoor
```
- This command takes effect immediately.
- 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 system installs the single best route in the IP routing table. If you set the value greater than 1, the system installs that number of parallel routes.

maximum-paths

- Use to set the maximum number of equal cost multipaths.
- Specify a value in the range 1–6; the default value is 1.
- The maximum number applies only to routes learned from external peers unless you use the **ibgp** keyword, in which case the maximum number applies only to routes received from internal 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 address family, or a VRF specified in the context of an IPv4 unicast address family.
- This command does not support VPNv4 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.

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 1-35 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
```

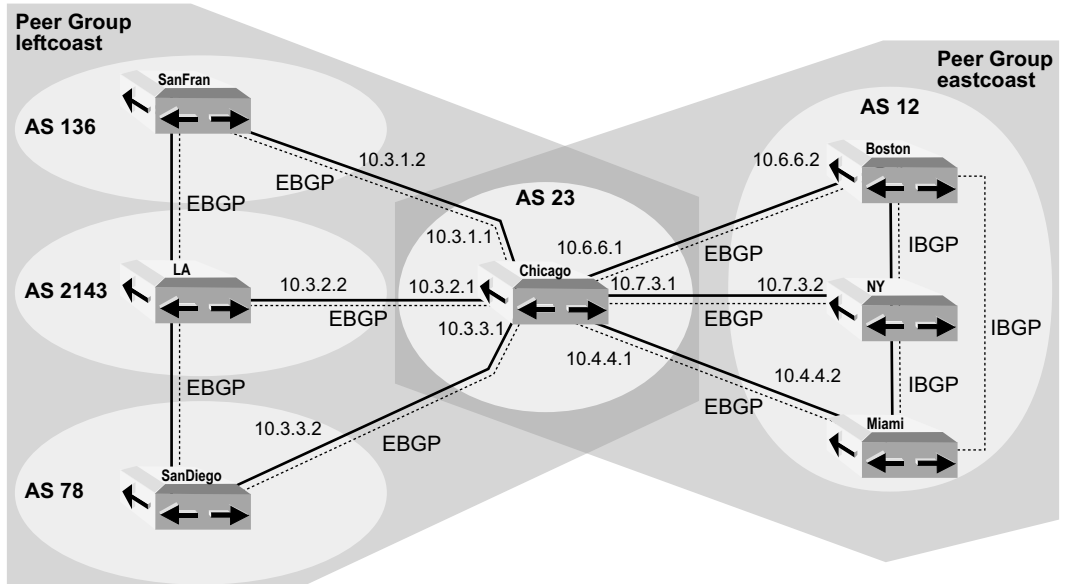


Figure 1-35 BGP peer groups

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.

For information on the inheritance of configuration values by peer groups and peers, see *Inheritance of Configuration Values* earlier in this chapter.

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, if an AS has 9 BGP peers,

$$\text{IBGP sessions} = \frac{9^2 - 9}{2} = 36$$

BGP provides two alternative configuration strategies to reduce the number of fully meshed peers. You can either:

- 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 1-36 shows a simpler system. 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.

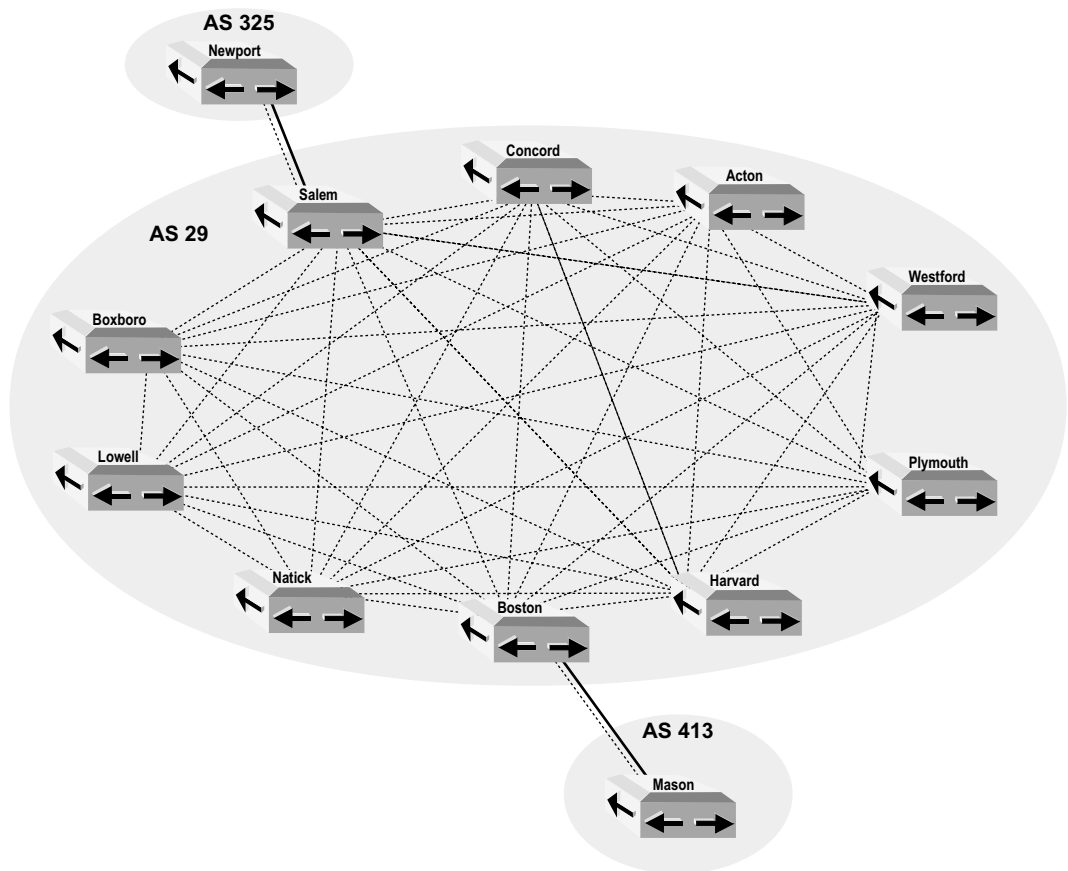


Figure 1-36 A fully meshed autonomous system

Figure 1-37 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.

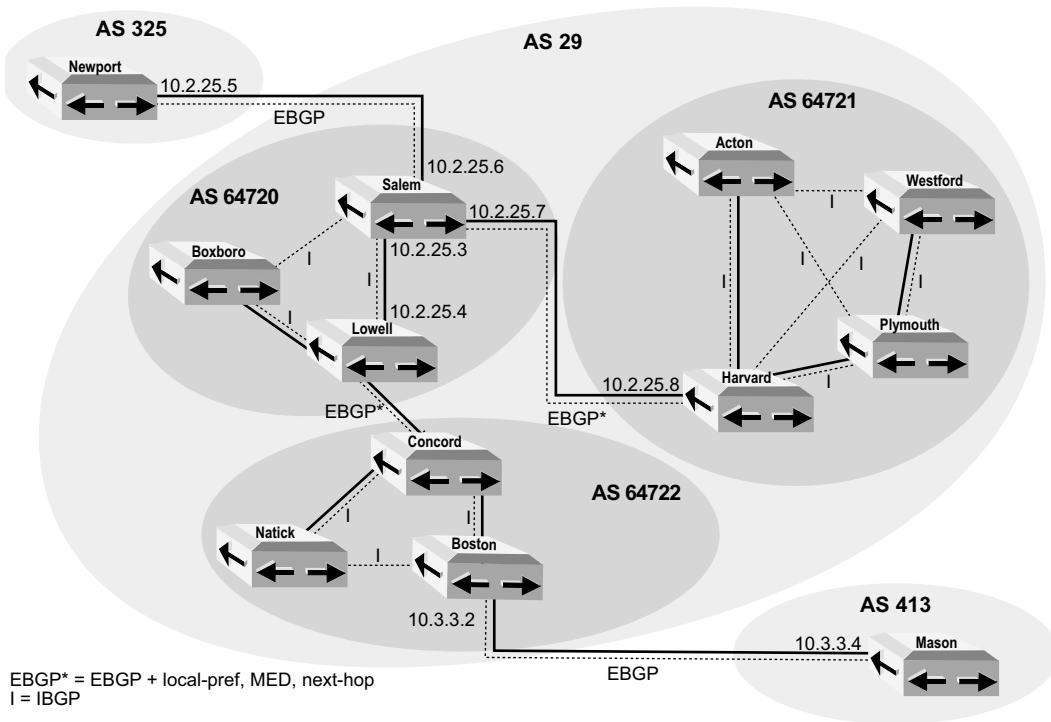


Figure 1-37 A confederation of sub-autonomous systems

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 systems 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

- Enables 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 considered to be 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 1-38 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.

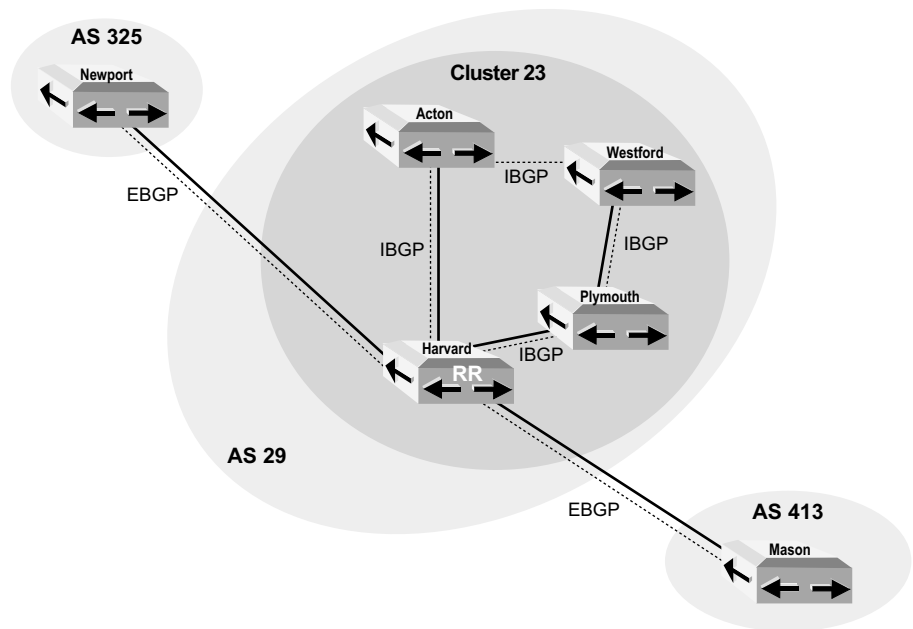


Figure 1-38 Simple route reflection

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 1-38 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 1-39. 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.

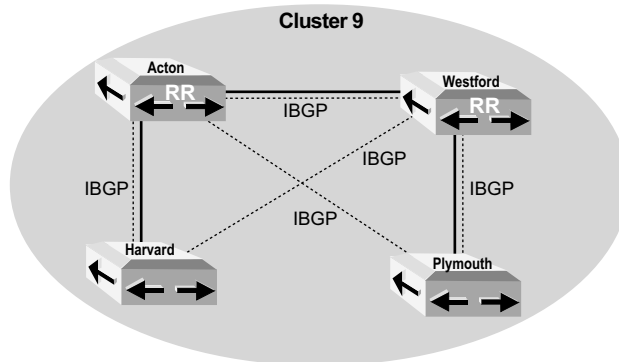


Figure 1-39 Route reflection: logical redundancy

In Figure 1-40, 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.

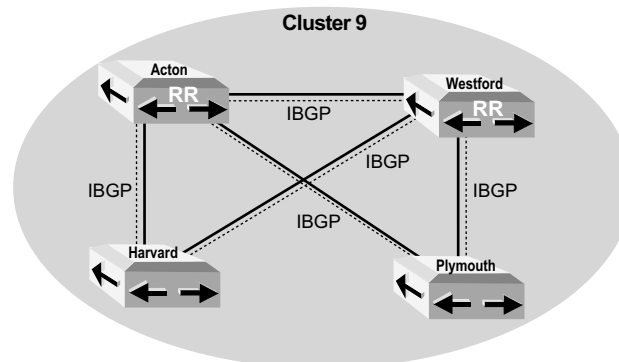


Figure 1-40 Route reflection: physical and logical redundancy

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.

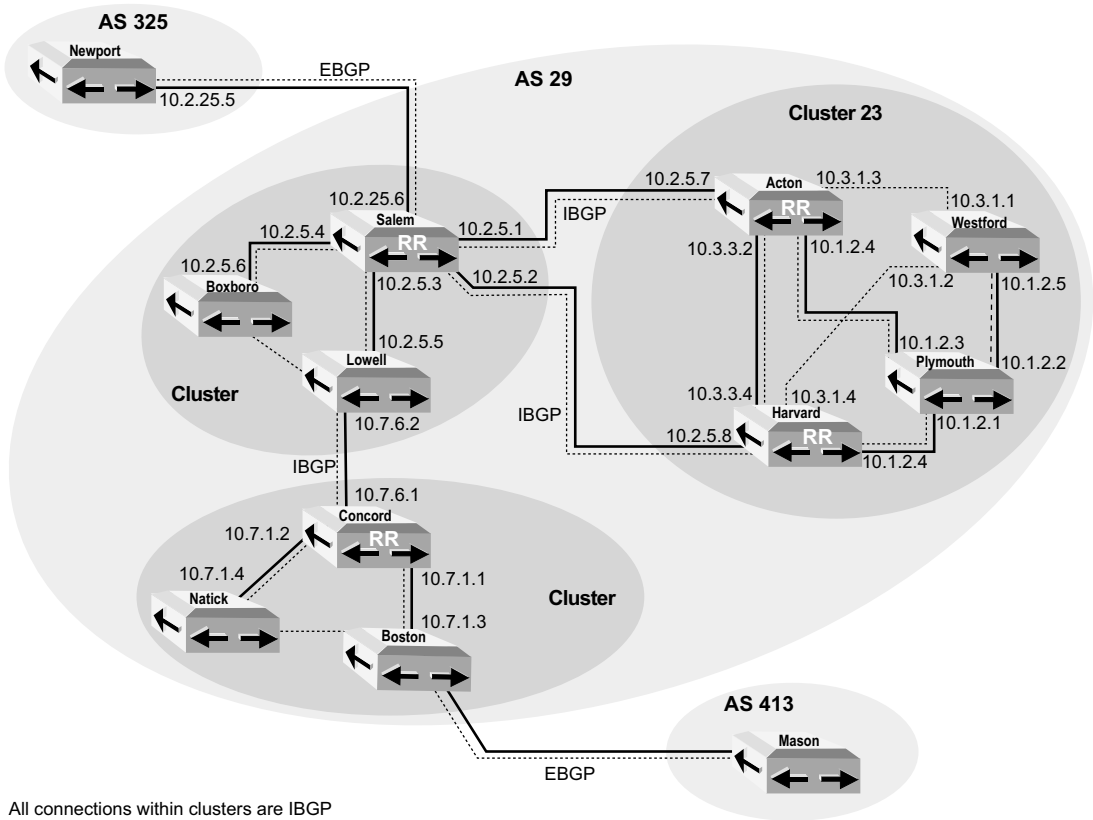
What about routes that a client forwards out of the cluster? No cluster ID is needed, because clients can forward routes only to EBGPeers, that is, to peers outside the AS. Looping between ASs is prevented by the AS-path list.

The following commands configure the route reflectors for the network topology shown in Figure 1-41. You would configure the other routers, whether nonclients or route reflector clients, as usual for IBGP and EBGPeers.

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.



All connections within clusters are IBGP

Figure 1-41 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 RIB, 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 IBGP 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 RIB, 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 *ERX Routing Protocols Configuration Guide, Vol. 1, Chapter 3, Configuring IP Multicasting* for information on 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 – When you enable BGP, the system employs unicast IPv4 addresses by default. You must specify an address family other than unicast IPv4 for the new features.
- Multicast IPv4 – If you specify the multicast IPv4 address family, you can configure the system to exchange routes to multicast sources (as opposed to routes to unicast destinations).

- VPN-IPv4 – If you specify the VPN-IPv4 address family, sometimes referred to as VPNv4, you can configure the system to provide IPv4 VPN services via an MPLS backbone.

As discussed in *Understanding BGP Command Scope* (p 1-15), 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 1-4 to configure parameters that affect the multicast address family globally. You can issue the commands listed in Table 1-6 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 1-5 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 addresses in unicast, multicast, or VPN mode.
- The default setting is to exchange IPv4 addresses in unicast mode from the default router.
- 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
```

- This command takes effect immediately.
- 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.
- Example

```
host1:vr1(config-router-af)#neighbor 192.168.1.158 activate
```
- 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 RIB of the newly activated address family.
- Use the **no** version to indicate that routes of the current address family should 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* in this chapter.

Using BGP Routes for Other Protocols

You can use the **ip route-type** command to specify whether BGP IPv4 unicast routes are available only for other 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

- 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 routes** command to view the routes available for unicast protocols.
- Use the **show ip rpf-routes** command to view the routes available for multicast protocols. It does not display routes available only to unicast protocols.
- By default, BGP IPv4 unicast routes are available only for other unicast routing protocols.

- Example 1

```
host1(config)#router bgp 100
host1(config-router)#ip 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 system to provide IPv4 VPN services via an MPLS backbone. These VPNs are often referred to as

BGP/MPLS VPNs. For detailed information, see *Chapter 3, Configuring BGP/MPLS VPNs*.

Monitoring BGP

Use the **show** commands in this section to monitor BGP activity. 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. Refer to *ERX System Basics Configuration Guide, Chapter 2, Command Line Interface*, for details.

Use the **debug ip bgp** command to get information on 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 on 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
```

Neighbor	AS	State	Up/down time	Messages Sent	Messages Received
1.1.1.1	100	Established	00:07:55	94	92

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

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

- 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
- Examples

```
host1:pe1#show ip bgp
Local BGP identifier 1.1.1.1, local AS 100
  2 routes (104 bytes), 3 distinct destinations (216 bytes)
  1 route selected for route table installation
  11 path attribute entries (1364 bytes)
Local-RIB version 2. Processed 2 changes (0 pending)
```

Status codes: > best, * invalid, s suppressed, d dampened, r rejected, a auto-summarized

Prefix	Peer	Next-hop	MED	LocPrf	Weight	Origin
> 10.88.88.1/32	0.0.0.0	0.0.0.0	1		32768	IGP
> 10.88.88.2/32	10.2.2.2	10.2.2.2	1	100	0	IGP

```
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

```

```
host1:5#show ip bgp 10.66.66.66
```

```

BGP route information for prefix 10.66.66.66/32
Received route learned from peer 10.6.6.6
Route placed in IP forwarding table
Not advertised to any peers
Address Family Identifier (AFI) is ip-v4
Subsequent Address Family Identifier (SAFI) is unicast
Next hop IP address is 10.6.6.6 (not reachable)
Multi-exit discriminator is 1
Local preference is 100
Weight is 0
Origin is IGP
AS path is empty

```

- 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. 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.
- You can use the **default-fields route** command to specify default fields to be displayed by subsequently issued **show ip bgp** commands.
- Example

```

host1#show ip bgp fields intro peer as-path
Local router ID 192.168.1.232, local AS 200
 4 paths, 4 distinct prefixes (272 bytes used)
 4 paths selected for route table installation
 1 path attribute entry (129 bytes used)

```

Prefix	Peer	AS-path
10.6.128.0/17	192.168.1.158	100
10.15.1.1/32	192.168.1.158	100
10.15.1.1/32	192.168.1.232	not present
172.25.2.2/32	192.168.1.158	100
172.25.2.2/32	192.168.1.232	not present
172.16.1.1/32	192.168.1.232	not present
172.16.2.2/32	192.168.1.232	not present
192.168.1.0/24	192.168.1.158	100

192.168.1.0/24 192.168.1.232 not present

- Example

```
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

show ip bgp 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 via the **no rib-out disable** command or the **no neighbor rib-out disable** command. Otherwise, this command returns an error message.
- 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

- 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 ip bgp aggregate-address
```

Prefix	AS set	Summary only	Attribute map	Advertise map
12.0.0.0/8	No	Yes	None	None
158.101.0.0/16	No	No	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.
- 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

- 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
- 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

- 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.
- 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

- Use to display information on dampened routes.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- 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. Once the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default).
 - › 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
```

```
Cutoff 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
```

Prefix	Peer	Status	Figure of Merit	Time until 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

- 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.
- 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

- 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. Once the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default).
 - › 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
```

Prefix	Peer	Status	Figure of Merit	Time until 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

- 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.
- 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 has 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

- Use the **longer-prefixes** option 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.
- 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

- 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)
 - › 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

- 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
    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

```

show ip bgp 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.
- 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

- 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

- 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.
- 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 neighbor 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

- Use to display, after inbound policy is applied, all routes that originate from the specified neighbor.
- Reports whether the indirect next hop of a route is unreachable; if not, displays the IGP cost to the indirect next hop.
- 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 neighbors 10.5.0.48 routes
```

```
Local router ID 192.168.1.153, local AS 100
```

```
72082 paths, 72082 distinct prefixes (5189904 bytes used)
```

```
72082 paths selected for route table installation
```

```
21695 path attribute entries (2966686 bytes used)
```

Prefix	Peer	Next-hop	MED	CalPrf	Weight	Origin
> 3.0.0.0/8	10.5.0.48	10.5.0.48	100	100	100	IGP
> 4.0.0.0/8	10.5.0.48	10.5.0.48	100	100	100	IGP
> 4.17.106.0/24	10.5.0.48	10.5.0.48	100	100	100	IGP
> 4.17.115.0/24	10.5.0.48	10.5.0.48	100	100	100	IGP
> 6.0.0.0/8	10.5.0.48	10.5.0.48	100	100	100	IGP
> 9.2.0.0/16	10.5.0.48	10.5.0.48	100	100	100	IGP
> 9.20.0.0/17	10.5.0.48	10.5.0.48	100	100	100	IGP
> 12.0.0.0/8	10.5.0.48	10.5.0.48	100	100	100	IGP
> 12.0.48.0/20	10.5.0.48	10.5.0.48	100	100	100	IGP
> 12.1.248.0/24	10.5.0.48	10.5.0.48	100	100	100	IGP
> 12.2.6.0/24	10.5.0.48	10.5.0.48	100	100	100	IGP
> 12.2.7.0/24	10.5.0.48	10.5.0.48	100	100	100	IGP
> 12.2.76.0/24	10.5.0.48	10.5.0.48	100	100	100	IGP

show ip bgp network

- Use to display information about networks in an AS.
- Example

```
host1#show ip bgp network
```

```
192.168.1.0/24
192.168.2.0/24
192.168.10.0/24
```

show ip bgp 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 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

- 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 ip bgp 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 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
 - › 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
- Example


```
host1#show ip bgp peer-group
BGP peer group leftcoast
Members: 10.1.1.2 10.2.2.2 10.3.3.2
Options:
  Default originate is disabled
  EBGp multi-hop is disabled
```

```

Next hop self is disabled
Send community is disabled
Private AS number stripping is disabled
Maximum update message size is 1024 octets
Policy:
Neighbor weight is 500
Timers:
Minimum route advertisement interval is 30 seconds

```

show ip bgp 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.

show ip bgp 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.

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 reg ^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     100    200
10.99.1.3/32    10.1.1.2      100   100     100    200 10
10.99.1.4/32    10.1.1.2      100   100     100    200 10 20
```

show ip bgp 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.
- 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
 - › Default local preference – default value for local preference
 - › IGP synchronization – indicates whether synchronization is enabled or disabled
 - › Default originate – indicates whether network 0.0.0.0 is redistributed into BGP
 - › Always compare MED – status of always compare MED
 - › Route dampening – status of route dampening
 - › Router is a route reflector – indicates whether the router has been configured as a route reflector
 - › Cluster ID – cluster IDs
 - › Client-to-client reflection – whether client-to-client reflection is configured
 - › Confederation ID – confederation ID
 - › Confederation peers – confederation peers
 - › 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
- Example

```
host1#show ip bgp summary
Local router ID 192.168.1.218, local AS 100
  Default local preference is 100
  IGP synchronization is enabled
  Default originate is disabled
  Always compare MED is disabled
  Route dampening is disabled
```

Neighbor	AS	Ver.	State	Up/down time	Messages	
					Sent	Received
10.2.1.48	11488	4	Established	0 17:42:57	4102	44926
192.168.1.106	200	4	Established	0 17:42:03	2364	2127

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 on BGP logs that was previously enabled with the **debug ip bgp** command.
- Example

```
host1#undebug ip bgp
```

- There is no **no** version.

