IP Routing: BGP Configuration Guide, Cisco IOS Release 12.2SY

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Cisco BGP Overview

Border Gateway Protocol (BGP) is an interdomain routing protocol designed to provide loop-free routing between separate routing domains that contain independent routing policies (autonomous systems). The Cisco IOS software implementation of BGP version 4 includes support for 4-byte autonomous system numbers and multiprotocol extensions to allow BGP to carry routing information for IP multicast routes and multiple Layer 3 protocol address families including IP Version 4 (IPv4), IP Version 6 (IPv6), Virtual Private Networks version 4 (VPNv4), Connectionless Network Services (CLNS), and Layer 2 VPN (L2VPN). This module contains conceptual material to help you understand how BGP is implemented in Cisco IOS software.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Cisco BGP

This document assumes knowledge of CLNS, IPv4, IPv6, multicast, VPNv4, and Interior Gateway Protocols (IGPs). The amount of knowledge required for each technology is dependent on your deployment.

Restrictions for Cisco BGP

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

Information About Cisco BGP

BGP Version 4 Functional Overview

BGP is an interdomain routing protocol designed to provide loop-free routing links between organizations. BGP is designed to run over a reliable transport protocol; it uses TCP (Port 179) as the transport protocol because TCP is a connection-oriented protocol. The destination TCP port is assigned 179, and the local port assigned a random port number. Cisco IOSsoftware supports BGP version 4 and it is this version that has been used by Internet Service Providers to help build the Internet. RFC 1771 introduced and discussed a number of new BGP features to allow the protocol to scale for Internet use. RFC 2858 introduced multiprotocol extensions to allow BGP to carry routing information for IP multicast routes and multiple Layer 3 protocol address families including IPv4, IPV6, and CLNS.

BGP is mainly used to connect a local network to an external network to gain access to the Internet or to connect to other organizations. When connecting to an external organization, external BGP (eBGP) peering sessions are created. Although BGP is referred to as an exterior gateway protocol (EGP) many networks within an organization are becoming so complex that BGP can be used to simplify the internal network used within the organization. BGP peers within the same organization exchange routing information through internal BGP (iBGP) peering sessions. For more details about configuring BGP peer sessions and other tasks to build a basic BGP network, see the "Configuring a Basic BGP Network" module.

BGP uses a path-vector routing algorithm to exchange network reachability information with other BGP speaking networking devices. Network reachability information is exchanged between BGP peers in routing updates. Network reachability information contains the network number, path specific attributes, and the list of autonomous system numbers that a route must transit through to reach a destination network. This list is contained in the AS-path attribute. BGP prevents routing loops by rejecting any routing update that contains the local autonomous system number because this indicates that the route has already travelled through that autonomous system and a loop would therefore be created. The BGP path-vector routing algorithm is a combination of the distance-vector routing algorithm and the AS-path loop detection. For more details about configuration tasks to configure various options involving BGP neighbor peer sessions, see the "Configuring BGP Neighbor Session Options" module.

BGP selects a single path, by default, as the best path to a destination host or network. The best path selection algorithm analyzes path attributes to determine which route is installed as the best path in the BGP routing table. Each path carries well-known mandatory, well-know discretionary, and optional transitive attributes that are used in BGP best path analysis. Cisco IOS software provides the ability to influence BGP path selection by altering some of these attributes using the command-line interface (CLI.) BGP path selection can also be influenced through standard BGP policy configuration. For more details about using BGP to influence path selection and configuring BGP policies to filter traffic, see the "Connecting to a Service Provider Using External BGP" module.

BGP uses the best-path selection algorithm to find a set of equally good routes. These routes are the potential multipaths. In Cisco IOS Release 12.2(33)SRD and later releases, when there are more equally good multipaths available than the maximum permitted number, then the oldest paths are selected as multipaths.

BGP can be used to help manage complex internal networks by interfacing with Interior Gateway Protocols (IGPs). Internal BGP can help with issues such as scaling the existing IGPs to match the traffic demands while maintaining network efficiency. For more details about configuring advanced BGP features including tasks to configure iBGP peering sessions, see the "Configuring Advanced BGP Features" module.

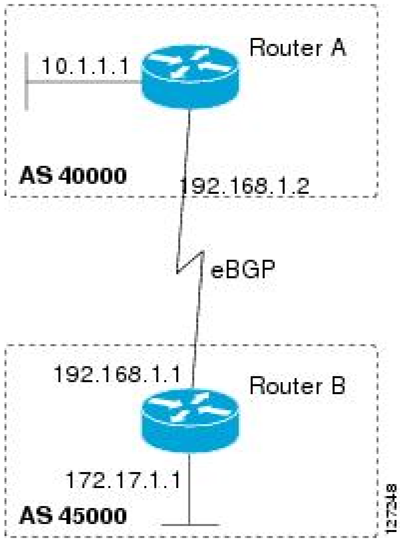
BGP Autonomous Systems

An autonomous system is a network controlled by a single technical administration entity. BGP autonomous systems are used to divide global external networks into individual routing domains where local routing policies are applied. This organization simplifies routing domain administration and simplifies consistent policy configuration. Consistent policy configuration is important to allow BGP to efficiently process routes to destination networks.

Each routing domain can support multiple routing protocols. However, each routing protocol is administrated separately. Other routing protocols can dynamically exchange routing information with BGP through redistribution. Separate BGP autonomous systems dynamically exchange routing information through eBGP peering sessions. BGP peers within the same autonomous system exchange routing information through iBGP peering sessions.

The figure below illustrates two routers in separate autonomous systems that can be connected using BGP. Router A and Router B are Internet service provider (ISP) routers in separate routing domains that use public autonomous system numbers. These routers carry traffic across the Internet. Router A and Router B are connected through eBGP peering sessions.

Figure 1 BGP Topology with Two Autonomous Systems



Each public autonomous system that directly connects to the Internet is assigned a unique number that identifies both the BGP routing process and the autonomous system.

BGP Autonomous System Number Formats

Prior to January 2009, BGP autonomous system numbers that were allocated to companies were two-octet numbers in the range from 1 to 65535 as described in RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*. Due to increased demand for autonomous system numbers, the Internet Assigned Number Authority (IANA) will start in January 2009 to allocate four-octet autonomous system numbers in the range from 65536 to 4294967295. RFC 5396, *Textual Representation of Autonomous System (AS) Numbers*, documents three methods of representing autonomous system numbers. Cisco has implemented the following two methods:

- Asplain--Decimal value notation where both 2-byte and 4-byte autonomous system numbers are represented by their decimal value. For example, 65526 is a 2-byte autonomous system number and 234567 is a 4-byte autonomous system number.
- Asdot--Autonomous system dot notation where 2-byte autonomous system numbers are represented by their decimal value and 4-byte autonomous system numbers are represented by a dot notation. For example, 65526 is a 2-byte autonomous system number and 1.169031 is a 4-byte autonomous system number (this is dot notation for the 234567 decimal number).

For details about the third method of representing autonomous system numbers, see RFC 5396.

Asdot Only Autonomous System Number Formatting

In Cisco IOS Release 12.0(32)S12, 12.4(24)T, and later releases, the 4-octet (4-byte) autonomous system numbers are entered and displayed only in asdot notation, for example, 1.10 or 45000.64000. When using regular expressions to match 4-byte autonomous system numbers the asdot format includes a period which is a special character in regular expressions. A backslash must be entered before the period for example, 1\.14, to ensure the regular expression match does not fail. The table below shows the format in which 2-byte and 4-byte autonomous system numbers are configured, matched in regular expressions, and displayed in **show** command output in Cisco IOS images where only asdot formatting is available.

Table 1 Asdot Only 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Asplain as Default Autonomous System Number Formatting

In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain and asdot format. In addition, the default format for matching 4-byte autonomous system numbers in regular expressions is asplain, so you must ensure that any regular expressions to match 4-byte autonomous system numbers are written in the asplain format. If you want to change the default **show** command output to display 4-byte autonomous system numbers in the asdot format, use the **bgp asnotation dot** command under router configuration mode. When the asdot format is enabled as the default, any regular expressions to match 4-byte autonomous system numbers must be written using the asdot format, or the regular expression match will fail. The tables below show that although you can configure 4-byte autonomous system numbers in either asplain or asdot format, only one format is used to display **show** command output and control 4-byte autonomous system number matching for regular expressions, and the default is asplain format. To display 4-byte autonomous system numbers in **show** command output and to control matching for regular expressions in the asdot format, you must configure the **bgp asnotation dot** command. After enabling the **bgp asnotation dot** command, a hard reset must be initiated for all BGP sessions by entering the **clear ip bgp *** command.

Note If you are upgrading to an image that supports 4-byte autonomous system numbers, you can still use 2-byte autonomous system numbers. The **show** command output and regular expression match are not changed and remain in asplain (decimal value) format for 2-byte autonomous system numbers regardless of the format configured for 4-byte autonomous system numbers.

Table 2 Default Asplain 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 65536 to 4294967295
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 65536 to 4294967295

Table 3 Asdot 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Reserved and Private Autonomous System Numbers

In Cisco IOS Release 12.0(32)S12, 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, 12.4(24)T, and later releases, the Cisco implementation of BGP supports RFC 4893. RFC 4893 was developed to allow BGP to support a gradual transition from 2-byte autonomous system numbers to 4-byte autonomous system numbers. A new reserved (private) autonomous system number, 23456, was created by RFC 4893 and this number cannot be configured as an autonomous system number in the Cisco IOS CLI.

RFC 5398, *Autonomous System (AS) Number Reservation for Documentation Use*, describes new reserved autonomous system numbers for documentation purposes. Use of the reserved numbers allow configuration examples to be accurately documented and avoids conflict with production networks if these configurations are literally copied. The reserved numbers are documented in the IANA autonomous system number registry. Reserved 2-byte autonomous system numbers are in the contiguous block, 64496 to 64511 and reserved 4-byte autonomous system numbers are from 65536 to 65551 inclusive.

Private 2-byte autonomous system numbers are still valid in the range from 64512 to 65534 with 65535 being reserved for special use. Private autonomous system numbers can be used for internal routing domains but must be translated for traffic that is routed out to the Internet. BGP should not be configured to advertise private autonomous system numbers to external networks. Cisco IOS software does not remove private autonomous system numbers from routing updates by default. We recommend that ISPs filter private autonomous system numbers.

Note Autonomous system number assignment for public and private networks is governed by the IANA. For information about autonomous-system numbers, including reserved number assignment, or to apply to register an autonomous system number, see the following URL: http://www.iana.org/.

Classless Interdomain Routing

BGP version 4 supports classless interdomain routing (CIDR). CIDR eliminates classful network boundaries, providing more efficient usage of the IPv4 address space. CIDR provides a method to reduce the size of routing tables by configuring aggregate routes (or supernets). CIDR processes a prefix as an IP address and bit mask (bits are processed from left to right) to define each network. A prefix can represent a network, subnetwork, supernet, or single host route. For example, using classful IP addressing, the IP address 192.168.2.1 is defined as a single host in the Class C network 192.168.2.0. Using CIDR the IP address can be shown as 192.168.2.1/16, which defines a network (or supernet) of 192.168.0.0. CIDR is enabled by default for all routing protocols in Cisco IOS software. Enabling CIDR affects how packets are forwarded but it does not change the operation of BGP.

Multiprotocol BGP

Cisco IOS software supports multiprotocol BGP extensions as defined in RFC 2858, *Multiprotocol Extensions for BGP-4* . The extensions introduced in this RFC allow BGP to carry routing information for multiple network-layer protocols, including CLNS, IPv4, IPv6, and VPNv4. These extensions are backward-compatible to enable routers that do not support multiprotocol extensions to communicate with those routers that do support multiprotocol extensions. Multiprotocol BGP carries routing information for multiple network-layer protocols and IP multicast routes. BGP carries different sets of routes depending on the protocol. For example, BGP can carry one set of routes for IPv4 unicast routing, one set of routes for IPv4 multicast routing, and one set of routes for MPLS VPNv4 routes.

Note A multiprotocol BGP network is backward-compatible with a BGP network, but BGP peers that do not support multiprotocol extensions cannot forward routing information, such as address family identifier information, that the multiprotocol extensions carry.

Benefits of Using Multiprotocol BGP Versus BGP

In complex networks with multiple network layer protocols, multiprotocol BGP must be used. In less complex networks we recommend using multiprotocol BGP because it offers the following benefits:

- All of the BGP commands and routing policy capabilities of BGP can be applied to multiprotocol BGP.
- A network can carry routing information for multiple network layer protocol address families (for example, IP Version 4 or VPN Version 4) as specified in RFC 1700, *Assigned Numbers* .
- A network can support incongruent unicast and multicast topologies.
- A multiprotocol BGP network is backward compatible because the routers that support the multiprotocol extensions can interoperate with routers that do not support the extensions.

In summary, multiprotocol BGP support for multiple network layer protocol address families provides a flexible and scalable infrastructure that allows you to define independent policy and peering configurations on a per-address family basis.

Multiprotocol BGP Extensions for IP Multicast

The routes associated with multicast routing are used by the Protocol Independent Multicast (PIM) feature to build data distribution trees. Multiprotocol BGP is useful when you want a link dedicated to multicast traffic, perhaps to limit which resources are used for which traffic. For example, you want all multicast traffic exchanged at one network access point (NAP). Multiprotocol BGP allows you to have a unicast routing topology different from a multicast routing topology that allows you more control over your network and resources.

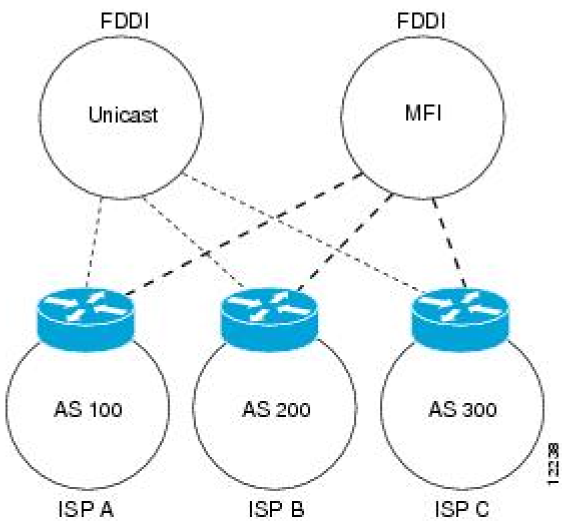
In BGP, the only way to perform interdomain multicast routing is to use the BGP infrastructure that is in place for unicast routing. If the routers are not multicast-capable, or there are differing policies about where multicast traffic should flow, multicast routing cannot be supported without multiprotocol BGP.

A multicast routing protocol, such as PIM, uses both the multicast and unicast BGP database to source the route, perform Reverse Path Forwarding (RPF) lookups for multicast-capable sources, and build a multicast distribution tree (MDT). The multicast table is the primary source for the router, but if the route is not found in the multicast table then the unicast table is searched. Although multicast can be performed with unicast BGP, multicast BGP routes allow an alternative topology to be used for RPF.

It is possible to configure BGP peers that exchange both unicast and multicast Network Layer Reachability Information (NLRI) where multiprotocol BGP routes can be redistributed into BGP. Multiprotocol extensions, however, will be ignored by any peers that do not support multiprotocol BGP. When PIM builds a multicast distribution tree through a unicast BGP network (because the route through the unicast network is the most attractive), the RPF check may fail, preventing the MDT from being built. If the unicast network runs multiprotocol BGP, peering can be configured using the appropriate multicast address family. The multicast address family configuration enables multiprotocol BGP to carry the multicast information and the RPF lookup will succeed.

The figure below illustrates a simple example of unicast and multicast topologies that are incongruent; these topologies cannot exchange information without implementing multiprotocol BGP. Autonomous systems 100, 200, and 300 are each connected to two NAPs that are FDDI rings. One is used for unicast peering (and therefore the exchanging of unicast traffic). The Multicast Friendly Interconnect (MFI) ring is used for multicast peering (and therefore the exchanging of multicast traffic). Each router is unicast- and multicast-capable.

Figure 2 Incongruent Unicast and Multicast Routes

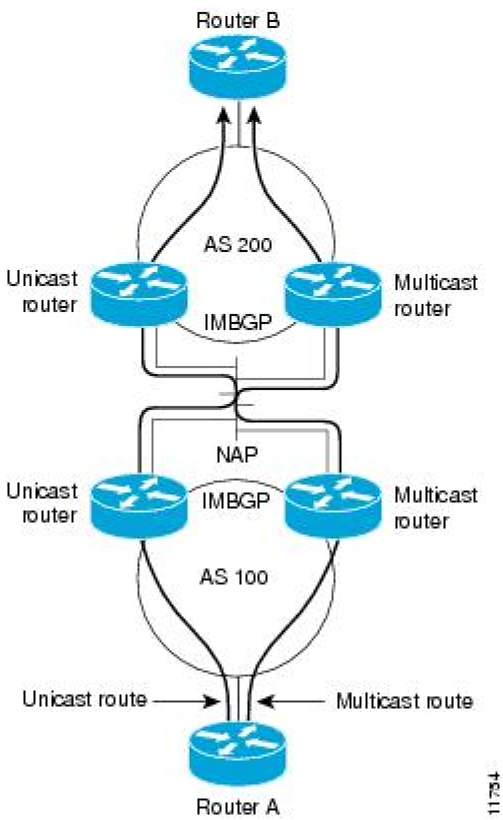


The figure below is a topology of unicast-only routers and multicast-only routers. The two routers on the left are unicast-only routers (that is, they do not support or are not configured to perform multicast routing). The two routers on the right are multicast-only routers. Routers A and B support both unicast and multicast routing. The unicast-only and multicast-only routers are connected to a single NAP.

In the figure below, only unicast traffic can travel from Router A to the unicast routers to Router B and back. Multicast traffic could not flow on that path, because multicast routing is not configured on the unicast routers and therefore the BGP routing table does not contain any multicast routes. On the multicast routers, multicast routes are enabled and BGP builds a separate routing table to hold the multicast routes. Multicast traffic uses the path from Router A to the multicast routers to Router B and back.

The figure below illustrates a multiprotocol BGP environment with a separate unicast route and multicast route from Router A to Router B. Multiprotocol BGP allows these routes to be noncongruent. Both of the autonomous systems must be configured for internal multiprotocol BGP in the figure.

Figure 3 Multicast BGP Environment



For more information about IP multicast, see the "Configuring IP Multicast" configuration library.

NLRI Configuration CLI

BGP was designed to carry only unicast IPv4 routing information. BGP configuration used the Network NLRI format CLI in Cisco IOS software. The NLRI format offers only limited support for multicast routing information and does not support multiple network layer protocols. We do not recommend using NLRI format CLI for BGP configuration.

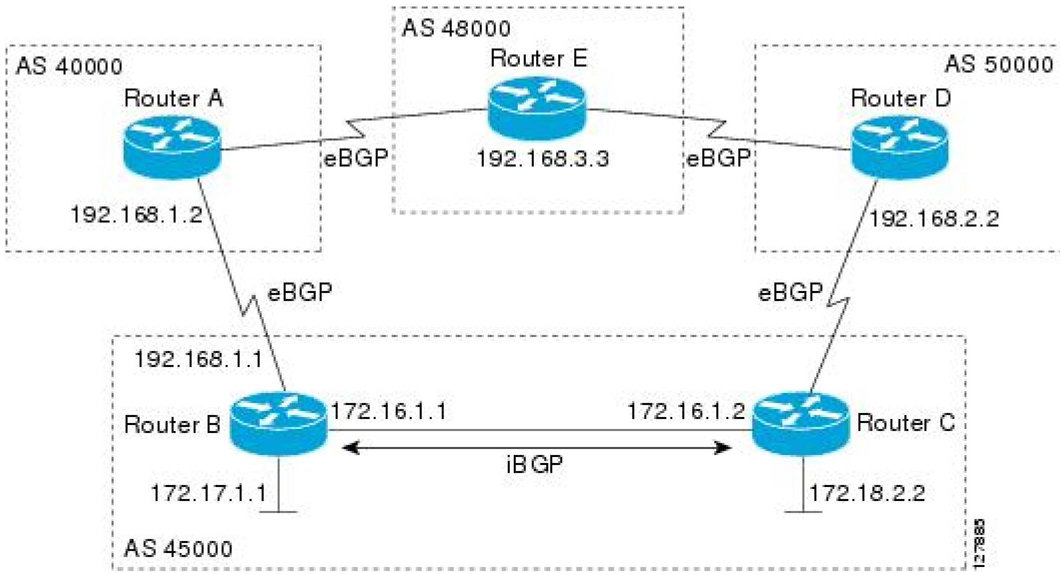
Using the BGP hybrid CLI feature, you can configure commands in the address family VPNv4 format and save these command configurations without modifying an existing NLRI formatted configuration. If you want to use other address family configurations such as IPv4 unicast or multicast, then you must upgrade the configuration using the **bgp upgrade-cli** command.

For more details about using BGP hybrid CLI command, see the "Configuring a Basic BGP Network" module. See the GUID-79FA6FF0-3F89-481E-BA84-ACD428597B7E and GUID-C0429A81-02C9-469F-B18B-FC64CBB62A67 concepts for more information about address family configuration format and the limitations of the NLRI CLI format.

Cisco BGP Address Family Model

The Cisco BGP address family identifier (AFI) model was introduced with multiprotocol BGP and is designed to be modular and scalable, and to support multiple AFI and subsequent address family identifier (SAFI) configurations. Networks are increasing in complexity and many companies are now using BGP to connect to many autonomous systems, as shown in the network topology in the figure below. Each of the separate autonomous systems shown in the figure below may be running several routing protocols such as Multiprotocol Label Switching (MPLS) and IPv6 and require both unicast and multicast routes to be transported via BGP.

Figure 4 BGP Network Topology for Multiple Address Families



The Cisco BGP AFI model introduced new command-line interface (CLI) commands supported by a new internal structure. Multiprotocol BGP carries routing information for multiple network layer protocols and IP multicast routes. This routing information is carried in the AFI model as appended BGP attributes (multiprotocol extensions). Each address family maintains a separate BGP database, which allows you to configure BGP policy on per-address family basis. SAFI configurations are subsets of the parent AFI. SAFIs can be used to refine BGP policy configurations.

The AFI model was created because of scalability limitations of the NLRI format. A router that is configured in NLRI format has IPv4 unicast but limited multicast capabilities. Networks that are configured in the NLRI format have the following limitations:

- No support for AFI and SAFI configuration information. Many new BGP (and other protocols such as MPLS) features are supported only in AFI and SAFI configuration modes and cannot be configured in NLRI configuration modes.
- No support for IPv6. A router that is configured in the NLRI format cannot establish peering with an IPv6 neighbor.
- Limited support for multicast interdomain routing and incongruent multicast and unicast topologies. In the NLRI format, not all configuration options are available and there is no support for VPNv4. The NLRI format configurations can be more complex than configurations that support the AFI model. If the routers in the infrastructure do not have multicast capabilities, or if policies differ as to where multicast traffic is configured to flow, multicast routing cannot be supported.

The AFI model in multiprotocol BGP supports multiple AFIs and SAFIs, all NLRI-based commands and policy configurations, and is backward compatible with routers that support only the NLRI format. A router that is configured using the AFI model has the following features:

- AFI and SAFI information and configurations are supported. A router that is configured using the AFI model can carry routing information for multiple network layer protocol address families (for example, IPv4 and IPv6).
- AFI configuration is similar in all address families, making the CLI syntax easier to use than the NLRI format syntax.
- All BGP routing policy capabilities and commands are supported.
- Congruent unicast and multicast topologies that have different policies (BGP filtering configurations) are supported, as are incongruent multicast and unicast topologies.
- CLNS is supported.
- Interoperation between routers that support only the NLRI format (AFI-based networks are backward compatible) is supported. This includes both IPv4 unicast and multicast NLRI peers.
- Virtual Private Networks (VPNs) and VPN routing and forwarding (VRF) instances are supported. Unicast IPv4 for VRFs can be configured from a specific address family IPv4 VRF; this configuration update is integrated into the BGP VPNv4 database.

Within a specific address family configuration mode, the question mark (?) online help function can be used to display supported commands. The BGP commands supported in address family configuration mode configure the same functionality as the BGP commands supported in router configuration mode; however, the BGP commands in router configuration mode configure functionality only for the IPv4 unicast address prefix. To configure BGP commands and functionality for other address family prefixes (for example, the IPv4 multicast or IPv6 unicast address prefixes), you must enter address family configuration mode for those address prefixes.

The BGP address family model consists of four address families in Cisco IOS software; IPv4, IPv6, CLNS, and VPNv4. In Cisco IOS Release 12.2(33)SRB, and later releases, support for the L2VPN address family was introduced, and within the L2VPN address family the VPLS SAFI is supported. Within the IPv4 and IPv6 address families SAFIs such as Multicast Distribution Tree (MDT), tunnel, and VRF exist. The table below shows the list of SAFIs supported by Cisco IOS software. To ensure compatibility between networks running all types of AFI and SAFI configuration, we recommend configuring BGP on Cisco IOS devices using the multiprotocol BGP address family model.

Table 4 SAFIs Supported by Cisco IOS Software

SAFI Field Value	Description	Reference
1	NLRI used for unicast forwarding.	RFC 2858
2	NLRI used for multicast forwarding.	RFC 2858
3	NLRI used for both unicast and multicast forwarding.	RFC 2858
4	NLRI with MPLS labels.	RFC 3107
64	Tunnel SAFI.	draft-nalawade-kapoor-tunnel-safi -01.txt

SAFI Field Value	Description	Reference
65	Virtual Private LAN Service (VPLS).	--
66	BGP MDT SAFI.	draft-nalawade-idr-mdt-safi-00.txt
128	MPLS-labeled VPN address.	RFC-ietf-l3vpn-rfc2547bis-03.txt

IPv4 Address Family

The IPv4 address family is used to identify routing sessions for protocols such as BGP that use standard IP version 4 address prefixes. Unicast or multicast address prefixes can be specified within the IPv4 address family. Routing information for address family IPv4 unicast is advertised by default when a BGP peer is configured unless the advertisement of unicast IPv4 information is explicitly turned off.

VRF instances can also be associated with IPv4 AFI configuration mode commands.

In Cisco IOS Release 12.0(28)S, the tunnel SAFI was introduced to support multipoint tunneling IPv4 routing sessions. The tunnel SAFI is used to advertise the tunnel endpoints and the SAFI specific attributes that contain the tunnel type and tunnel capabilities. Redistribution of tunnel endpoints into the BGP IPv4 tunnel SAFI table occurs automatically when the tunnel address family is configured. However, peers need to be activated under the tunnel address family before the sessions can exchange tunnel information.

In Cisco IOS Release 12.0(29)S, the multicast distribution tree (MDT) SAFI was introduced to support multicast VPN architectures. The MDT SAFI is a transitive multicast capable connector attribute that is defined as an IPv4 address family in BGP. The MDT address family session operates as a SAFI under the IPv4 multicast address family, and is configured on provider edge (PE) routers to establish VPN peering sessions with customer edge (CE) routers that support inter-AS multicast VPN peering sessions.

IPv6 Address Family

The IPv6 address family is used to identify routing sessions for protocols such as BGP that use standard IPv6 address prefixes. Unicast or multicast address prefixes can be specified within the IPv6 address family.

Note Routing information for address family IPv4 unicast is advertised by default when you configure a BGP peer unless you explicitly turn off the advertisement of unicast IPv4 information.

CLNS Address Family

The CLNS address family is used to identify routing sessions for protocols such as BGP that use standard network service access point (NSAP) address prefixes. Unicast address prefixes are the default when NSAP address prefixes are configured.

CLNS routes are used in networks where CLNS addresses are configured. This is typically a telecommunications Data Communications Network (DCN). Peering is established using IP addresses, but update messages contain CLNS routes.

For more details about configuring BGP support for CLNS, which provides the ability to scale CLNS networks, see the "Configuring Multiprotocol BGP (MP-BGP) support for CLNS" module.

VPNv4 Address Family

The VPNv4 multicast address family is used to identify routing sessions for protocols such as BGP that use standard VPN Version 4 address prefixes. Unicast address prefixes are the default when VPNv4 address prefixes are configured. VPNv4 routes are the same as IPv4 routes, but VPNv4 routes have a route descriptor (RD) prepended that allows replication of prefixes. It is possible to associate every different RD with a different VPN. Each VPN needs its own set of prefixes.

Companies use an IP VPN as the foundation for deploying or administering value-added services including applications and data hosting network commerce, and telephony services to business customers.

In private LANs, IP-based intranets have fundamentally changed the way companies conduct their business. Companies are moving their business applications to their intranets to extend over a WAN. Companies are also addressing the needs of their customers, suppliers, and partners by using extranets (an intranet that encompasses multiple businesses). With extranets, companies reduce business process costs by facilitating supply-chain automation, electronic data interchange (EDI), and other forms of network commerce. To take advantage of this business opportunity, service providers must have an IP VPN infrastructure that delivers private network services to businesses over a public infrastructure.

VPNs, when used with MPLS, allow several sites to transparently interconnect through a service provider's network. One service provider network can support several different IP VPNs. Each of these appears to its users as a private network, separate from all other networks. Within a VPN, each site can send IP packets to any other site in the same VPN. Each VPN is associated with one or more VPN VRFs. VPNv4 routes are a superset of routes from all VRFs, and route injection is done per VRF under the specific VRF address family. The router maintains a separate routing and Cisco Express Forwarding (CEF) table for each VRF. This prevents information from being sent outside the VPN and allows the same subnet to be used in several VPNs without causing duplicate IP address problems. The router using BGP distributes the VPN routing information using the BGP extended communities.

The VPN address space is isolated from the global address space by design. BGP distributes reachability information for VPN-IPv4 prefixes for each VPN using the VPNv4 multiprotocol extensions to ensure that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.

RFC 3107 specifies how to add label information to multiprotocol BGP address families using a SAFI. The Cisco IOS implementation of MPLS uses RFC 3107 to provide support for sending IPv4 routes with a label. VPNv4 routes implicitly have a label associated with each route.

L2VPN Address Family

In Cisco IOS Release 12.2(33)SRB and later releases, support for the L2VPN address family is introduced. L2VPN is defined as a secure network that operates inside an unsecured network by using an encryption technology such as IP security (IPsec) or Generic Routing Encapsulation (GRE). The L2VPN address family is configured under BGP routing configuration mode, and within the L2VPN address family the VPLS subsequent address family identifier (SAFI) is supported.

BGP support for the L2VPN address family introduces a BGP-based autodiscovery mechanism to distribute L2VPN endpoint provisioning information. BGP uses a separate L2VPN routing information base (RIB) to store endpoint provisioning information, which is updated each time any Layer 2 VFI is configured. Prefix and path information is stored in the L2VPN database, allowing BGP to make best-path decisions. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, the endpoint information is used to set up a pseudowire mesh to support L2VPN-based services.

The BGP autodiscovery mechanism facilitates the setting up of L2VPN services, which are an integral part of the Cisco IOS Virtual Private LAN Service (VPLS) feature. VPLS enables flexibility in deploying services by connecting geographically dispersed sites as a large LAN over high-speed Ethernet in a robust and scalable IP MPLS network. For more details about VPLS, see the "VPLS Autodiscovery: BGP Based" feature.

Under L2VPN address family the following BGP command-line interface (CLI) commands are supported:

- **bgp scan-time**
- **bgp nexthop**
- **neighbor activate**
- **neighbor advertisement-interval**
- **neighbor allowas-in**
- **neighbor capability**
- **neighbor inherit**
- **neighbor peer-group**
- **neighbor maximum-prefix**
- **neighbor next-hop-self**
- **neighbor next-hop-unchanged**
- **neighbor remove-private-as**
- **neighbor route-map**
- **neighbor route-reflector-client**
- **neighbor send-community**
- **neighbor soft-reconfiguration**
- **neighbor soo**
- **neighbor weight**

Note For route reflectors using L2VPNs, the **neighbor next-hop-self** and **neighbor next-hop-unchanged** commands are not supported.

For route maps used within BGP, all commands related to prefix processing, tag processing, and automated tag processing are ignored when used under L2VPN address family configuration. All other route map commands are supported.

BGP multipaths and confederations are not supported under the L2VPN address family.

For details on configuring BGP under the L2VPN address family, see the "BGP Support for the L2VPN Address Family" feature in Cisco IOS Release 12.2(33)SRB.

BGP CLI Removal Considerations

BGP CLI configuration can become quite complex even in smaller BGP networks. If you need to remove any CLI configuration, you must consider all the implications of removing the CLI. Analyze the current running configuration to determine the current BGP neighbor relationships, any address family considerations, and even other routing protocols that are configured. Many BGP CLI commands affect other parts of the CLI configuration. For example, in the following configuration, a route map is used to match a BGP autonomous system number and then set the matched routes with another autonomous system number for EIGRP:

```
route-map bgp-to-eigrp permit 10
 match tag 50000
 set tag 65000
```

BGP neighbors in three different autonomous systems are configured and activated:

```
router bgp 45000
 bgp log-neighbor-changes
 address-family ipv4
  neighbor 172.16.1.2 remote-as 45000
  neighbor 192.168.1.2 remote-as 40000
  neighbor 192.168.3.2 remote-as 50000
  neighbor 172.16.1.2 activate
  neighbor 192.168.1.2 activate
  neighbor 192.168.3.2 activate
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
```

An EIGRP routing process is then configured and BGP routes are redistributed into EIGRP with a route map filtering the routes:

```
router eigrp 100
 redistribute bgp 45000 metric 10000 100 255 1 1500 route-map bgp-to-eigrp
 no auto-summary
 exit
```

If you later decide to remove the route map, you will use the **no** form of the **route-map** command. Almost every configuration command has a **no** form, and the **no** form generally disables a function. However, in this configuration example, if you disable only the route map, the route redistribution will continue, but without the filtering or matching from the route map. Redistribution without the route map may cause unexpected results in your network. When you remove an access list or route map, you must also review the commands that referenced that access list or route map to consider whether the command will give you the behavior you intended.

The following configuration will remove both the route map and the redistribution:

```
configure terminal
 no route-map bgp-to-eigrp
 router eigrp 100
  no redistribute bgp 45000
  end
```

For details on configuring the removal of BGP CLI configuration, see the "Configuring a Basic BGP Network" module.

Where to Go Next

Proceed to the "Configuring a Basic BGP Network" module.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*
Configuring basic BGP tasks	"Configuring a Basic BGP Network" module

Related Topic	Document Title
Configuring BGP neighbor session options	"Configuring BGP Neighbor Session Options" module
Configuring BGP to connect to a service provider	"Connecting to a Service Provider Using External BGP" module
Configuring internal BGP (iBGP) tasks	"Configuring Internal BGP Features" module
Configuring advanced BGP features	"Configuring Advanced BGP Features" module
Configuring Multiprotocol BGP with CLNS	"Configuring Multiprotocol BGP (MP-BGP) Support for CLNS" module
Configuring basic IP multicast tasks	"Configuring Basic IP Multicast" module

Standards

Standard	Title
MDT SAFI	MDT SAFI

MIBs

MIB	MIBs Link
CISCO-BGP4-MIB	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1700	*Assigned Numbers*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 3107	*Carrying Label Information in BGP-4*
RFC 4271	A Border Gateway Protocol 4 (BGP-4)
RFC 4893	BGP Support for Four-Octet AS Number Space
RFC 5396	Textual Representation of Autonomous System (AS) Numbers
RFC 5398	Autonomous System (AS) Number Reservation for Documentation Use

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Cisco BGP Overview

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 5 Feature Information for Cisco BGP Overview

Feature Name	Releases	Feature Information
BGP Support for 4-Byte ASN	12.0(32)S12 12.0(32)SY8 12.0(33)S3 12.2(33)SRE 12.2(33)XNE 12.2(33)SXI1 12.4(24)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Support for 4-Byte ASN feature introduced support for 4-byte autonomous system numbers. Because of increased demand for autonomous system numbers, in January 2009 the IANA will start to allocate 4-byte autonomous system numbers in the range from 65536 to 4294967295. In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, and 12.2(33)SXI1, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default regular expression match and output display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain format and the asdot format as described in RFC 5396. To change the default regular expression match and output display of 4-byte autonomous system numbers to asdot format, use the **bgp asnotation dot** command. In Cisco IOS Release 12.0(32)S12, and 12.4(24)T, the Cisco implementation of 4-byte autonomous system numbers uses asdot as the only configuration format, regular expression match, and output display, with no asplain support. The following commands were introduced or modified by this feature: **bgp asnotation dot**, **bgp confederation identifier**, **bgp confederation peers**, all **clear ip bgp**commands that configure an autonomous system number, **ip as-path access-list**, **ip extcommunity-list**, **match**

Feature Name	Releases	Feature Information
		source-protocol, **neighbor local-as**, **neighbor remote-as**, **neighbor soo**, **redistribute (**IP**)**, **router bgp**, **route-target**, **set as-path**, **set extcommunity**, **set origin**, **soo**, all **show ip bgp** commands that display an autonomous system number, and **show ip extcommunity-list**.
BGP Support for the L2VPN Address Family	12.2(33)SRB	BGP Support for the L2VPN address family introduced a BGP-based autodiscovery mechanism to distribute L2VPN endpoint provisioning information. BGP uses a separate L2VPN routing information base (RIB) to store endpoint provisioning information which is updated each time any Layer 2 VFI is configured. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, the endpoint information is used to set up a Pseudowire mesh to support L2VPN-based services. The following commands were introduced or modified by this feature: **address-family l2vpn**, **show ip bgp l2vpn**.

Feature Name	Releases	Feature Information
Configuring Multiprotocol BGP Support for CLNS	12.2(33)SRB	The Multiprotocol BGP (MP-BGP) Support for CLNS feature provides the ability to scale Connectionless Network Service (CLNS) networks. The multiprotocol extensions of Border Gateway Protocol (BGP) add the ability to interconnect separate Open System Interconnection (OSI) routing domains without merging the routing domains, thus providing the capability to build very large OSI networks. The following commands were introduced or modified by this feature: **clear bgp nsap**, **clear bgp nsap dampening**, **clear bgp nsap external**, **clear bgp nsap flap-statistics**, **clear bgp nsap peer-group**, **debug bgp nsap**, **debug bgp nsap dampening**, **debug bgp nsap updates**, **neighbor prefix-list, network (BGP and multiprotocol BGP)**, **redistribute (BGP to ISO ISIS)**, **redistribute (ISO ISIS to BGP)**, **show bgp nsap**, **show bgp nsap community**, **show bgp nsap community-list**, **show bgp nsap dampened-paths**, **show bgp nsap filter-list**, **show bgp nsap flap-statistics**, **show bgp nsap inconsistent-as**, **show bgp nsap neighbors**, **show bgp nsap paths**, **show bgp nsap quote-regexp**, **show bgp nsap regexp**, **show bgp nsap summary**.

Feature Name	Releases	Feature Information
Multiprotocol BGP	Cisco IOS XE 3.1.0SG	Cisco IOS software supports multiprotocol BGP extensions as defined in RFC 2858, *Multiprotocol Extensions for BGP-4* . The extensions introduced in this RFC allow BGP to carry routing information for multiple network layer protocols including CLNS, IPv4, IPv6, and VPNv4. These extensions are backward compatible to enable routers that do not support multiprotocol extensions to communicate with those routers that do support multiprotocol extensions. Multiprotocol BGP carries routing information for multiple network layer protocols and IP multicast routes.

Configuring a Basic BGP Network

This module describes the basic tasks to configure a basic Border Gateway Protocol (BGP) network. BGP is an interdomain routing protocol that is designed to provide loop-free routing between organizations. The Cisco IOS implementation of the neighbor and address family commands is explained. This module also contains tasks to configure and customize BGP peers, implement BGP route aggregation, configure BGP route origination, and define BGP backdoor routes. BGP peer group definition is documented, peer session templates are introduced, and update groups are explained,

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring a Basic BGP Network

Before configuring a basic BGP network, you should be familiar with the "Cisco BGP Overview" module.

Restrictions for Configuring a Basic BGP Network

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

Information About Configuring a Basic BGP Network

BGP Version 4

Border Gateway Protocol (BGP) is an interdomain routing protocol designed to provide loop-free routing between separate routing domains that contain independent routing policies (autonomous systems). The Cisco IOSsoftware implementation of BGP version 4 includes multiprotocol extensions to allow BGP to carry routing information for IP multicast routes and multiple Layer 3 protocol address families including IP Version 4 (IPv4), IP Version 6 (IPv6), and Virtual Private Networks version 4 (VPNv4).

BGP is mainly used to connect a local network to an external network to gain access to the Internet or to connect to other organizations. When connecting to an external organization, external BGP (eBGP) peering sessions are created. Although BGP is referred to as an exterior gateway protocol (EGP) many networks within an organization are becoming so complex that BGP can be used to simplify the internal network used within the organization. BGP peers within the same organization exchange routing information through internal BGP (iBGP) peering sessions.

Note BGP requires more configuration than other routing protocols, and the effects of any configuration changes must be fully understood. Incorrect configuration can create routing loops and negatively impact normal network operation.

BGP Router ID

BGP uses a router ID to identify BGP-speaking peers. The BGP router ID is a 32-bit value that is often represented by an IPv4 address. By default, the Cisco IOS software sets the router ID to the IPv4 address of a loopback interface on the router. If no loopback interface is configured on the router, then the software chooses the highest IPv4 address configured to a physical interface on the router to represent the BGP router ID. The BGP router ID must be unique to the BGP peers in a network.

BGP-Speaker and Peer Relationships

A BGP-speaking router does not discover another BGP-speaking device automatically. A network administrator usually manually configures the relationships between BGP-speaking routers. A peer device is a BGP-speaking router that has an active TCP connection to another BGP-speaking device. This relationship between BGP devices is often referred to as a neighbor but, as this can imply the idea that the BGP devices are directly connected with no other router in between, the term neighbor will be avoided whenever possible in this document. A BGP speaker is the local router and a peer is any other BGP-speaking network device.

When a TCP connection is established between peers, each BGP peer initially exchanges all its routes--the complete BGP routing table--with the other peer. After this initial exchange only incremental updates are sent when there has been a topology change in the network, or when a routing policy has been implemented or modified. In the periods of inactivity between these updates, peers exchange special messages called keepalives.

A BGP autonomous system is a network controlled by a single technical administration entity. Peer routers are called external peers when they are in different autonomous systems and internal peers when they are in the same autonomous system. Usually, external peers are adjacent and share a subnet; internal peers may be anywhere in the same autonomous system.

For more details about external BGP peers, see the "Connecting to a Service Provider Using External BGP" module. For more details about internal BGP peers, see the "Configuring Internal BGP Features" module.

BGP Autonomous System Number Formats

Prior to January 2009, BGP autonomous system numbers that were allocated to companies were 2-octet numbers in the range from 1 to 65535 as described in RFC 4271, *A Border Gateway Protocol 4 (BGP-4)* . Due to increased demand for autonomous system numbers, the Internet Assigned Number Authority (IANA) will start in January 2009 to allocate four-octet autonomous system numbers in the range from 65536 to 4294967295. RFC 5396, *Textual Representation of Autonomous System (AS) Numbers* , documents three methods of representing autonomous system numbers. Cisco has implemented the following two methods:

- Asplain--Decimal value notation where both 2-byte and 4-byte autonomous system numbers are represented by their decimal value. For example, 65526 is a 2-byte autonomous system number and 234567 is a 4-byte autonomous system number.
- Asdot--Autonomous system dot notation where 2-byte autonomous system numbers are represented by their decimal value and 4-byte autonomous system numbers are represented by a dot notation. For example, 65526 is a 2-byte autonomous system number and 1.169031 is a 4-byte autonomous system number (this is dot notation for the 234567 decimal number).

For details about the third method of representing autonomous system numbers, see RFC 5396.

Asdot Only Autonomous System Number Formatting

In Cisco IOS Release 12.0(32)S12, 12.4(24)T, and later releases, the 4-octet (4-byte) autonomous system numbers are entered and displayed only in asdot notation, for example, 1.10 or 45000.64000. When using regular expressions to match 4-byte autonomous system numbers the asdot format includes a period which is a special character in regular expressions. A backslash must be entered before the period (for example, 1\.14) to ensure the regular expression match does not fail. The table below shows the format in which 2-byte and 4-byte autonomous system numbers are configured, matched in regular expressions, and displayed in **show** command output in Cisco IOS images where only asdot formatting is available.

Table 6 Asdot Only 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Asplain as Default Autonomous System Number Formatting

In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain and asdot format. In addition, the default format for matching 4-byte autonomous system numbers in regular expressions is asplain, so you must ensure that any regular expressions to match 4-byte autonomous system numbers are written in the asplain format. If you want to change the default **show** command output to display 4-byte autonomous system numbers in the asdot format, use the **bgp asnotation dot** command under router configuration mode. When the asdot format is enabled as the default, any regular expressions to match 4-byte autonomous system numbers must be written using the asdot format, or the regular expression match will fail. The tables below show that although you can configure 4-byte autonomous system numbers in either asplain or asdot format, only one format is used to display **show** command output and control 4-byte autonomous system number matching for regular expressions, and the default is asplain format. To display 4-byte autonomous system numbers in **show** command output and to control matching for regular expressions in the asdot format, you must configure the **bgp asnotation dot** command. After enabling the **bgp asnotation dot** command, a hard reset must be initiated for all BGP sessions by entering the **clear ip bgp *** command.

Note If you are upgrading to an image that supports 4-byte autonomous system numbers, you can still use 2-byte autonomous system numbers. The **show** command output and regular expression match are not changed and remain in asplain (decimal value) format for 2-byte autonomous system numbers regardless of the format configured for 4-byte autonomous system numbers.

Table 7 Default Asplain 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 65536 to 4294967295
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 65536 to 4294967295

Table 8 Asdot 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Reserved and Private Autonomous System Numbers

In Cisco IOS Release 12.0(32)S12, 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, 12.4(24)T, and later releases, the Cisco implementation of BGP supports RFC 4893. RFC 4893 was developed to allow BGP to support a gradual transition from 2-byte autonomous system numbers to 4-byte autonomous system numbers. A new reserved (private) autonomous system number, 23456, was created by RFC 4893 and this number cannot be configured as an autonomous system number in the Cisco IOS CLI.

RFC 5398, *Autonomous System (AS) Number Reservation for Documentation Use* , describes new reserved autonomous system numbers for documentation purposes. Use of the reserved numbers allow configuration examples to be accurately documented and avoids conflict with production networks if these configurations are literally copied. The reserved numbers are documented in the IANA autonomous system number registry. Reserved 2-byte autonomous system numbers are in the contiguous block, 64496 to 64511 and reserved 4-byte autonomous system numbers are from 65536 to 65551 inclusive.

Private 2-byte autonomous system numbers are still valid in the range from 64512 to 65534 with 65535 being reserved for special use. Private autonomous system numbers can be used for internal routing domains but must be translated for traffic that is routed out to the Internet. BGP should not be configured to advertise private autonomous system numbers to external networks. Cisco IOS software does not remove private autonomous system numbers from routing updates by default. We recommend that ISPs filter private autonomous system numbers.

Note Autonomous system number assignment for public and private networks is governed by the IANA. For information about autonomous-system numbers, including reserved number assignment, or to apply to register an autonomous system number, see the following URL: http://www.iana.org/.

Cisco Implementation of 4-Byte Autonomous System Numbers

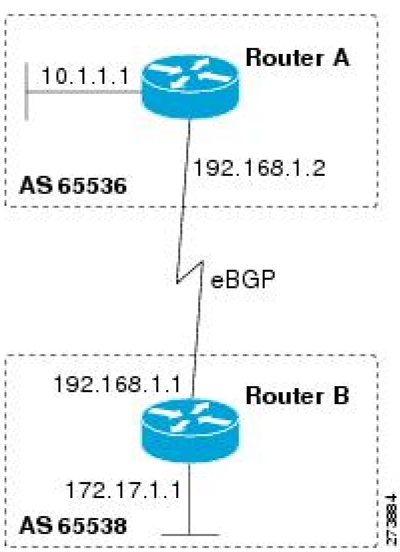
In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, the Cisco implementation of 4-byte autonomous system numbers uses asplain--65538 for example--as the default regular expression match and output display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain format and the asdot format as described in RFC 5396. To change the default regular expression match and output display of 4-byte autonomous system numbers to asdot format, use the **bgp asnotation dot** command followed by the **clear ip bgp *** command to perform a hard reset of all current BGP sessions.

In Cisco IOS Release 12.0(32)S12, and 12.4(24)T, the Cisco implementation of 4-byte autonomous system numbers uses asdot--1.2 for example--as the only configuration format, regular expression match, and output display, with no asplain support. For an example of BGP peers in two autonomous systems using 4-byte numbers, see the figure below. To view a configuration example of the configuration between three neighbor peers in separate 4-byte autonomous systems configured using asdot notation, see the Examples Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers, page 106.

Cisco also supports RFC 4893, which was developed to allow BGP to support a gradual transition from 2-byte autonomous system numbers to 4-byte autonomous system numbers. To ensure a smooth transition, we recommend that all BGP speakers within an autonomous system that is identified using a 4-byte autonomous system number be upgraded to support 4-byte autonomous system numbers.

Note A new private autonomous system number, 23456, was created by RFC 4893, and this number cannot be configured as an autonomous system number in the Cisco IOS CLI.

Figure 5 BGP Peers in Two Autonomous Systems Using 4-Byte Numbers



BGP Peer Session Establishment

When a BGP routing process establishes a peering session with a peer it goes through the following state changes:

- Idle--Initial state the BGP routing process enters when the routing process is enabled or when the router is reset. In this state, the router waits for a start event, such as a peering configuration with a remote peer. After the router receives a TCP connection request from a remote peer, the router initiates another start event to wait for a timer before starting a TCP connection to a remote peer. If the router is reset then the peer is reset and the BGP routing process returns to the Idle state.
- Connect--The BGP routing process detects that a peer is trying to establish a TCP session with the local BGP speaker.
- Active--In this state, the BGP routing process tries to establish a TCP session with a peer router using the ConnectRetry timer. Start events are ignored while the BGP routing process is in the Active state. If the BGP routing process is reconfigured or if an error occurs, the BGP routing process will release system resources and return to an Idle state.
- OpenSent--The TCP connection is established and the BGP routing process sends an OPEN message to the remote peer, and transitions to the OpenSent state. The BGP routing process can receive other OPEN messages in this state. If the connection fails, the BGP routing process transitions to the Active state.
- OpenReceive--The BGP routing process receives the OPEN message from the remote peer and waits for an initial keepalive message from the remote peer. When a keepalive message is received, the BGP routing process transitions to the Established state. If a notification message is received, the BGP routing process transitions to the Idle state. If an error or configuration change occurs that affects the peering session, the BGP routing process sends a notification message with the Finite State Machine (FSM) error code and then transitions to the Idle state.
- Established--The initial keepalive is received from the remote peer. Peering is now established with the remote neighbor and the BGP routing process starts exchanging update message with the remote peer. The hold timer restarts when an update or keepalive message is received. If the BGP process receives an error notification, it will transition to the Idle state.

Cisco Implementation of BGP Global and Address Family Configuration Commands

The address family model for configuring BGP is based on splitting apart the configuration for each address family. All commands that are independent of the address family are grouped together at the beginning (highest level) of the configuration, and these are followed by separate submodes for commands specific to each address family (with the exception that commands relating to IPv4 unicast can also be entered at the beginning of the configuration). When a network operator configures BGP, the flow of BGP configuration categories is represented by the following bullets in order:

- Global configuration--Configuration that is applied to BGP in general, rather than to specific neighbors. For example, the **network**, **redistribute**, and **bgp bestpath** commands.
- Address family-dependent configuration--Configuration that applies to a specific address family such as policy on an individual neighbor.

The relationship between BGP global and BGP address family-dependent configuration categories is shown in the table below.

Table 9 Relationships Between BGP Configuration Categories

BGP Configuration Category	Configuration Sets Within Category
Global address family-independent	One set of global address family-independent configurations

BGP Configuration Category	Configuration Sets Within Category
Address family-dependent	One set of global address family-dependent configurations per address family

Note Address family configuration must be entered within the address family submode to which it applies.

The following is an example of BGP configuration statements showing the grouping of global address family-independent and address family-dependent commands.

```
router bgp <AS>
 ! AF independent part
 neighbor <ip-address> <command> ! Session config; AF independent
 address-family ipv4 unicast
  ! AF dependant part
  neighbor <ip-address> <command> ! Policy config; AF dependant
  exit-address-family
 address-family ipv4 multicast
  ! AF dependant part
  neighbor <ip-address> <command> ! Policy config; AF dependant
  exit-address-family
 address-family ipv4 unicast vrf <vrf-name>
  ! VRF specific AS independent commands
  ! VRF specific AS dependant commands
  neighbor <ip-address> <command> ! Session config; AF independent
  neighbor <ip-address> <command> ! Policy config; AF dependant
  exit-address-family
```

The following example shows actual BGP commands that match the BGP configuration statements in the previous example:

```
router bgp 45000
 router-id 172.17.1.99
 bgp log-neighbor-changes
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.3.2 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.168.1.2 activate
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
 address-family ipv4 multicast
  neighbor 192.168.3.2 activate
  neighbor 192.168.3.2 advertisement-interval 25
  network 172.16.1.0 mask 255.255.255.0
  exit-address-family
 address-family ipv4 vrf vpn1
  neighbor 192.168.3.2 activate
  network 172.21.1.0 mask 255.255.255.0
  exit-address-family
```

In Cisco IOS Releases 12.0(22)S, 12.2(15)T, and later releases, the **bgp upgrade-cli** command simplifies the migration of BGP networks and existing configurations from the network layer reachability information (NLRI) format to the address family format. Network operators can configure commands in the address family identifier (AFI) format and save these command configurations to existing NLRI formatted configurations. The BGP hybrid command-line interface (CLI) does not add support for complete AFI and NLRI integration because of the limitations of the NLRI format. For complete support of AFI commands and features, we recommend upgrading existing NLRI configurations with the **bgp upgrade-cli** command. For a configuration example of migrating BGP configurations from the NLRI format to the address family format, see .

BGP Session Reset

Whenever there is a change in the routing policy due to a configuration change, BGP peering sessions must be reset using the **clear ip bgp** command. Cisco IOS software support the following three mechanisms to reset BGP peering sessions:

- Hard reset--A hard reset tears down the specified peering sessions including the TCP connection and deletes routes coming from the specified peer.
- Soft reset--A soft reset uses stored prefix information to reconfigure and activate BGP routing tables without tearing down existing peering sessions. Soft reconfiguration uses stored update information, at the cost of additional memory for storing the updates, to allow you to apply new BGP policy without disrupting the network. Soft reconfiguration can be configured for inbound or outbound sessions.
- Dynamic inbound soft reset--The route refresh capability, as defined in RFC 2918, allows the local router to reset inbound routing tables dynamically by exchanging route refresh requests to supporting peers. The route refresh capability does not store update information locally for non disruptive policy changes. It instead relies on dynamic exchange with supporting peers. Route refresh must first be advertised through BGP capability negotiation between peers. All BGP routers must support the route refresh capability. To determine if a BGP router supports this capability, use the **show ip bgp neighbors**command. The following message is displayed in the output when the router supports the route refresh capability:

```
Received route refresh capability from peer.
```

The **bgp soft-reconfig-backup** command was introduced to configure BGP to perform inbound soft reconfiguration for peers that do not support the route refresh capability. The configuration of this command allows you to configure BGP to store updates (soft reconfiguration) only as necessary. Peers that support the route refresh capability are unaffected by the configuration of this command.

BGP Route Aggregation

BGP peers store and exchange routing information and the amount of routing information increases as more BGP speakers are configured. The use of route aggregation reduces the amount of information involved. Aggregation is the process of combining the attributes of several different routes so that only a single route is advertised. Aggregate prefixes use the classless interdomain routing (CIDR) principle to combine contiguous networks into one classless set of IP addresses that can be summarized in routing tables. Fewer routes now need to be advertised.

Two methods are available in BGP to implement route aggregation. You can redistribute an aggregated route into BGP or you can use a form of conditional aggregation. Basic route redistribution involves creating an aggregate route and then redistributing the routes into BGP. Conditional aggregation involves creating an aggregate route and then advertising or suppressing the advertising of certain routes on the basis of route maps, autonomous system set path (AS-SET) information, or summary information.

The **bgp suppress-inactive** command configures BGP to not advertise inactive routes to any BGP peer. A BGP routing process can advertise routes that are not installed in the routing information database (RIB) to BGP peers by default. A route that is not installed into the RIB is an inactive route. Inactive route advertisement can occur, for example, when routes are advertised through common route aggregation. Inactive route advertisements can be suppressed to provide more consistent data forwarding.

BGP Aggregation Route AS-SET Information Generation

AS-SET information can be generated when BGP routes are aggregated using the **aggregate-address** command. The path advertised for such a route is an AS-SET consisting of all the elements, including the communities, contained in all the paths that are being summarized. If the AS-PATHs to be aggregated are identical, only the AS-PATH is advertised. The ATOMIC-AGGREGATE attribute, set by default for the **aggregate-address** command, is not added to the AS-SET.

Routing Policy Change Management

Routing policies for a peer include all the configurations for elements such as route map, distribute list, prefix list, and filter list that may impact inbound or outbound routing table updates. Whenever there is a change in the routing policy, the BGP session must be soft cleared, or soft reset, for the new policy to take effect. Performing inbound reset enables the new inbound policy configured on the router to take effect. Performing outbound reset causes the new local outbound policy configured on the router to take effect without resetting the BGP session. As a new set of updates is sent during outbound policy reset, a new inbound policy of the neighbor can also take effect. This means that after changing inbound policy you must do an inbound reset on the local router or an outbound reset on the peer router. Outbound policy changes require an outbound reset on the local router or an inbound reset on the peer router.

There are two types of reset: hard reset and soft reset. The table below lists their advantages and disadvantages.

Table 10 Advantages and Disadvantages of Hard and Soft Resets

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

Type of Reset	Advantages	Disadvantages
Configured inbound soft reset (uses the **neighbor soft-reconfiguration** router configuration command)	Can be used when both BGP routers do not support the automatic route refresh capability. In Cisco IOS Release 12.3(14)T, the **bgp soft-reconfig-backup** command was introduced to configure inbound soft reconfiguration for peers that do not support the route refresh capability.	Requires preconfiguration. Stores all received (inbound) routing policy updates without modification; is memory-intensive. Recommended only when absolutely necessary, such as when both BGP routers do not support the automatic route refresh capability. **Note** Does not reset outbound routing table updates.

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

A soft reset updates the routing table for inbound and outbound routing updates. Cisco IOS Release 12.1 and later releases support soft reset without any prior configuration. This soft reset allows the dynamic exchange of route refresh requests and routing information between BGP routers, and the subsequent readvertisement of the respective outbound routing table. There are two types of soft reset:

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

To use soft reset without preconfiguration, both BGP peers must support the soft route refresh capability, which is advertised in the OPEN message sent when the peers establish a TCP session. Routers running Cisco IOS releases prior to Release 12.1 do not support the route refresh capability and must clear the BGP session using the **neighbor soft-reconfiguration** router configuration command. Clearing the BGP session in this way will have a negative impact upon network operations and should be used only as a last resort.

Conditional BGP Route Injection

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

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

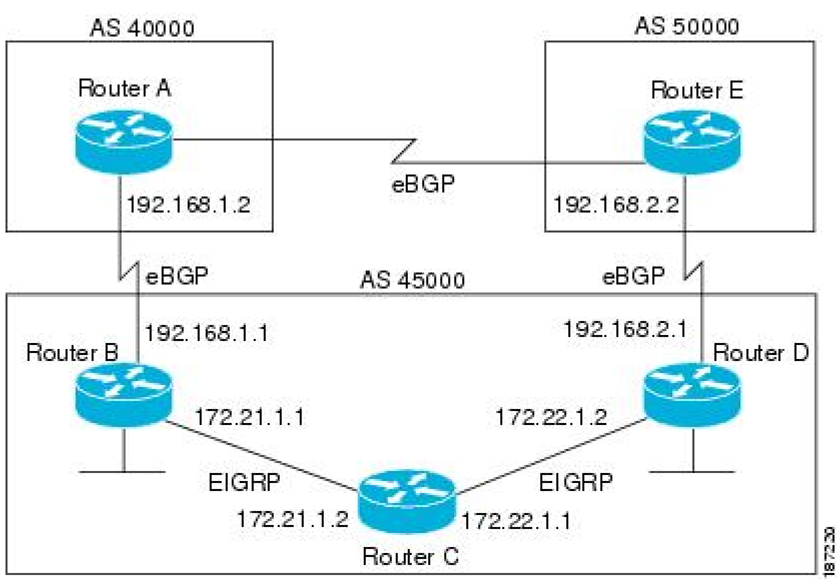
BGP Peer Groups

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

BGP Backdoor Routes

In a BGP network topology with two border routers using eBGP to communicate to a number of different autonomous systems, using eBGP to communicate between the two border routers may not be the most efficient routing method. In the figure below, Router B as a BGP speaker will receive a route to Router D through eBGP, but this route will traverse at least two autonomous systems. Router B and Router D are also connected through an Enhanced Interior Gateway Routing Protocol (EIGRP) network (any IGP can be used here) and this route has a shorter path. EIGRP routes, however, have a default administrative distance of 90 and eBGP routes have a default administrative distance of 20, so BGP will prefer the eBGP route. Changing the default administrative distances is not recommended because changing the administrative distance may lead to routing loops. To cause BGP to prefer the EIGRP route, you can use the **network backdoor** command. BGP treats the network specified by the **network backdoor** command as a locally assigned network, except that it does not advertise the specified network in BGP updates. In the figure below, this means that Router B will communicate to Router D using the shorter EIGRP route instead of the longer eBGP route.

Figure 6 BGP Backdoor Route Topology



Peer Groups and BGP Update Messages

In Cisco IOS software releases prior to Release 12.0(24)S, 12.2(18)S, or 12.3(4)T, BGP update messages were grouped based on peer group configurations. This method of grouping neighbors for BGP update message generation reduced the amount of system processing resources needed to scan the routing table. This method, however, had the following limitations:

- All neighbors that shared peer group configuration also had to share outbound routing policies.
- All neighbors had to belong to the same peer group and address family. Neighbors configured in different address families could not belong to different peer groups.

These limitations existed to balance optimal update generation and replication against peer group configuration. These limitations could cause the network operator to configure smaller peer groups, which reduced the efficiency of update message generation and limited the scalability of neighbor configuration.

BGP Update Group

The introduction of the BGP (dynamic) update group in Cisco IOS Releases 12.0(24)S, 12.2(18)S, 12.3(4)T, or 12.2(27)SBC, provides a different type of BGP peer grouping from existing BGP peer groups. Existing peer groups are not affected but peers with the same outbound policy configured that are not members of a current peer group can be grouped into an update group. The members of this update group will use the same update generation engine. When BGP update groups are configured an algorithm dynamically calculates the BGP update group membership based on outbound policies. Optimal BGP update message generation occurs automatically and independently. BGP neighbor configuration is no longer restricted by outbound routing policies, and update groups can belong to different address families.

BGP Dynamic Update Group Configuration

In Cisco IOS Release 12.0(24)S, 12.2(18)S, 12.3(4)T, 12.2(27)SBC, and later releases, a new algorithm was introduced that dynamically calculates and optimizes update groups of neighbors that share the same outbound policies and can share the same update messages. No configuration is required to enable the BGP dynamic update group and the algorithm runs automatically. When a change to outbound policy occurs, the router automatically recalculates update group memberships and applies the changes by triggering an outbound soft reset after a 1-minute timer expires. This behavior is designed to provide the network operator with time to change the configuration if a mistake is made. You can manually enable an outbound soft reset before the timer expires by entering the **clear ip bgp** *ip-address* **soft out**command.

Note In Cisco IOS Release 12.0(22)S, 12.2(14)S, 12.3(2)T, and prior releases, the update group recalculation delay timer is set to 3 minutes.

For the best optimization of BGP update group generation, we recommend that the network operator keeps outbound routing policy the same for neighbors that have similar outbound policies.

BGP Peer Templates

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

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

There are two types of peer templates:

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

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

Note A BGP neighbor cannot be configured to work with both peer groups and peer templates. A BGP neighbor can be configured to belong only to a peer group or to inherit policies from peer templates.

The following restrictions apply to the peer policy templates:

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

Inheritance in Peer Templates

The inheritance capability is a key component of peer template operation. Inheritance in a peer template is similar to node and tree structures commonly found in general computing, for example, file and directory trees. A peer template can directly or indirectly inherit the configuration from another peer template. The directly inherited peer template represents the tree in the structure. The indirectly inherited peer template represents a node in the tree. Because each node also supports inheritance, branches can be created that apply the configurations of all indirectly inherited peer templates within a chain back to the directly inherited peer template or the source of the tree. This structure eliminates the need to repeat configuration statements that are commonly reapplied to groups of neighbors because common configuration statements can be applied once and then indirectly inherited by peer templates that are applied to neighbor groups with common configurations. Configuration statements that are duplicated separately within a node and a tree are filtered out at the source of the tree by the directly inherited template. A directly inherited template will overwrite any indirectly inherited statements that are duplicated in the directly inherited template.

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

When BGP neighbors use inherited peer templates it can be difficult to determine which policies are associated with a specific template. In Cisco IOS 12.0(25)S, 12.4(11)T, 12.2(33)SRB, 12.2(33)SB, and later releases, the **detail** keyword was added to the **show ip bgp template peer-policy** command to display the detailed configuration of local and inherited policies associated with a specific template.

Peer Session Templates

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

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

General session commands can be configured once in a peer session template and then applied to many neighbors through the direct application of a peer session template or through indirect inheritance from a peer session template. The configuration of peer session templates simplifies the configuration of general session commands that are commonly applied to all neighbors within an autonomous system.

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

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

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

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

```
template peer-session SESSION-TEMPLATE-ONE
```

```
remote-as 1
exit peer-session
```

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

Peer Policy Templates

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

- **advertisement-interval**
- **allowas-in**
- **as-override**
- **capability**
- **default-originate**
- **distribute-list**
- **dmzlink-bw**
- **exit-peer-policy**
- **filter-list**
- **inherit peer-policy**
- **maximum-prefix**
- **next-hop-self**
- **next-hop-unchanged**
- **prefix-list**
- **remove-private-as**
- **route-map**
- **route-reflector-client**
- **send-community**
- **send-label**
- **soft-reconfiguration**
- **unsuppress-map**
- **weight**

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

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

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

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

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

BGP IPv6 Neighbor Activation Under the IPv4 Address Family

Prior to Cisco IOS Release 12.2(33)SRE4, by default, both IPv6 and IPv4 capability is exchanged with a BGP peer that has an IPv6 address. When an IPv6 peer is configured, that neighbor is automatically activated under the IPv4 unicast address family.

Beginning with Cisco IOS Release 12.2(33)SRE4, when a *new* IPv6 neighbor is being configured, it is no longer automatically activated under the IPv4 address family. You can manually activate the IPv6 neighbor under the IPv4 address family if, for example, you have a dual stack environment and want to send IPv6 and IPv4 prefixes.

If you do not want an *existing* IPv6 peer to be activated under the IPv4 address family, you can manually deactivate the peer with the **no neighbor activate** command. Until then, existing configurations that activate an IPv6 neighbor under the IPv4 unicast address family will continue to try to establish a session.

How to Configure a Basic BGP Network

Configuring a basic BGP network consists of a few required tasks and many optional tasks. A BGP routing process must be configured and BGP peers must be configured, preferably using the address family configuration model. If the BGP peers are part of a VPN network, the BGP peers must be configured using the IPv4 VRF address family task. The other tasks in the following list are optional:

- Configuring a BGP Routing Process, page 40
- Configuring a BGP Peer, page 43
- Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers, page 46
- Modifying the Default Output and Regular Expression Match Format for 4-Byte Autonomous System Numbers, page 50
- Configuring a BGP Peer for the IPv4 VRF Address Family, page 54
- Customizing a BGP Peer, page 57
- Removing BGP Configuration Commands Using a Redistribution, page 63
- Monitoring and Maintaining Basic BGP, page 65
- Aggregating Route Prefixes Using BGP, page 70
- Originating BGP Routes, page 80
- Configuring a BGP Peer Group, page 88
- Configuring Peer Session Templates, page 90

- Configuring Peer Policy Templates, page 96
- Monitoring and Maintaining BGP Dynamic Update Groups, page 103

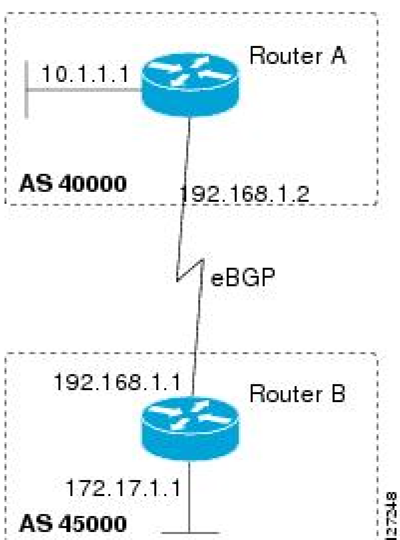
Configuring a BGP Routing Process

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

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

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

Figure 7 BGP Topology with Two Autonomous Systems



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
5. **bgp router-id** *ip-address*
6. **timers bgp** *keepalive holdtime*
7. **bgp fast-external-fallover**
8. **bgp log-neighbor-changes**
9. **end**
10. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Configures a BGP routing process, and enters router configuration mode for the specified routing process. • Use the *autonomous-system-number* argument to specify an integer, from 0 and 65534, that identifies the router to other BGP speakers.
Step 4	**network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*] **Example:** `Router(config-router)# network 10.1.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.

	Command or Action	Purpose
Step 5	**bgp router-id** *ip-address* **Example:** `Router(config-router)# bgp router-id 10.1.1.99`	(Optional) Configures a fixed 32-bit router ID as the identifier of the local router running BGP. • Use the *ip-address* argument to specify a unique router ID within the network. **Note** Configuring a router ID using the **bgp router-id** command resets all active BGP peering sessions.
Step 6	**timers bgp** *keepalive holdtime* **Example:** `Router(config-router)# timers bgp 70 120`	(Optional) Sets BGP network timers. • Use the *keepalive* argument to specify the frequency, in seconds, with which the software sends keepalive messages to its BGP peer. By default, the keepalive timer is set to 60 seconds. • Use the *holdtime* argument to specify the interval, in seconds, after not receiving a keepalive message that the software declares a BGP peer dead. By default, the holdtime timer is set to 180 seconds.
Step 7	**bgp fast-external-fallover** **Example:** `Router(config-router)# bgp fast-external-fallover`	(Optional) Enables the automatic resetting of BGP sessions. • By default, the BGP sessions of any directly adjacent external peers are reset if the link used to reach them goes down.
Step 8	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	(Optional) Enables logging of BGP neighbor status changes (up or down) and neighbor resets. • Use this command for troubleshooting network connectivity problems and measuring network stability. Unexpected neighbor resets might indicate high error rates or high packet loss in the network and should be investigated.
Step 9	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.
Step 10	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp`	(Optional) Displays the entries in the BGP routing table. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following sample output from the **show ip bgp** command shows the BGP routing table for Router A in the figure above after this task has been configured on Router A. You can see an entry for the network 10.1.1.0 that is local to this autonomous system.

```
BGP table version is 12, local router ID is 10.1.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      0.0.0.0                  0         32768 i
```

- Troubleshooting Tips, page 43

Troubleshooting Tips

Use the **ping** command to check basic network connectivity between the BGP routers.

Configuring a BGP Peer

Perform this task to configure BGP between two IPv4 routers (peers). The address family configured here is the default IPv4 unicast address family and the configuration is done at Router A in the figure above. Remember to perform this task for any neighbor routers that are to be BGP peers.

Before you perform this task, perform the Configuring a BGP Routing Process, page 40 task.

Note By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types, such as IPv6 prefixes.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **neighbor** *ip-address* **activate**
7. **end**
8. **show ip bgp** [*network*] [*network-mask*]
9. **show ip bgp neighbors** [*neighbor-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the virtual routing and forwarding (VRF) instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.1 activate`	Enables the neighbor to exchange prefixes for the IPv4 unicast address family with the local router.

	Command or Action	Purpose
Step 7	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.
Step 8	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp`	(Optional) Displays the entries in the BGP routing table. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 9	**show ip bgp neighbors** [*neighbor-address*] **Example:** `Router(config-router-af)# show ip bgp neighbors 192.168.2.2`	(Optional) Displays information about the TCP and BGP connections to neighbors. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following sample output from the **show ip bgp** command shows the BGP routing table for Router A in the figure above after this task has been configured on Router A and Router B. You can now see an entry for the network 172.17.1.0 in autonomous system 45000.

```
BGP table version is 13, local router ID is 10.1.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      0.0.0.0                  0         32768 i
*> 172.17.1.0/24    192.168.1.1              0             0 45000 i
```

The following sample output from the **show ip bgp neighbors** command shows information about the TCP and BGP connections to the BGP neighbor 192.168.1.1 of Router A in the figure above after this task has been configured on Router A:

```
BGP neighbor is 192.168.1.1,  remote AS 45000, external link
  BGP version 4, remote router ID 172.17.1.99
  BGP state = Established, up for 00:06:55
  Last read 00:00:15, last write 00:00:15, hold time is 120, keepalive intervals
  Configured hold time is 120,keepalive interval is 70 seconds, Minimum holdtims
  Neighbor capabilities:
    Route refresh: advertised and received (old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                1          2
    Keepalives:            13         13
    Route Refresh:          0          0
    Total:                 15         16
  Default minimum time between advertisement runs is 30 seconds
```

```
 For address family: IPv4 Unicast
  BGP table version 13, neighbor version 13/0
 Output queue size : 0
  Index 1, Offset 0, Mask 0x2
  1 update-group member
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               1          1 (Consumes 52 bytes)
    Prefixes Total:                 1          1
    Implicit Withdraw:              0          0
    Explicit Withdraw:              0          0
    Used as bestpath:             n/a          1
    Used as multipath:            n/a          0
                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    AS_PATH loop:                       n/a          1
    Bestpath from this peer:              1        n/a
    Total:                                1          1
  Number of NLRIs in the update sent: max 0, min 0
  Connections established 1; dropped 0
  Last reset never
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled
Local host: 192.168.1.2, Local port: 179
Foreign host: 192.168.1.1, Foreign port: 37725
Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)
Event Timers (current time is 0x12F4F2C):
Timer          Starts    Wakeups            Next
Retrans            14          0             0x0
TimeWait            0          0             0x0
AckHold            13          8             0x0
SendWnd             0          0             0x0
KeepAlive           0          0             0x0
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0
iss:  165379618  snduna:  165379963  sndnxt:  165379963     sndwnd:  16040
irs: 3127821601  rcvnxt: 3127821993  rcvwnd:      15993  delrcvwnd:    391
SRTT: 254 ms, RTTO: 619 ms, RTV: 365 ms, KRTT: 0 ms
minRTT: 12 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: passive open, nagle, gen tcbs
IP Precedence value : 6
Datagrams (max data segment is 1460 bytes):
Rcvd: 20 (out of order: 0), with data: 15, total data bytes: 391
Sent: 22 (retransmit: 0, fastretransmit: 0, partialack: 0, Second Congestion: 04
```


- Troubleshooting Tips, page 46
- What to Do Next, page 46

Troubleshooting Tips

Use the **ping** command to verify basic network connectivity between the BGP routers.

What to Do Next

If you have BGP peers in a VPN, proceed to theConfiguring a BGP Peer for the IPv4 VRF Address Family, page 54. If you do not have BGP peers in a VPN, proceed to the Customizing a BGP Peer, page 57.

Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers

Perform this task to configure a BGP routing process and BGP peers when the BGP peers are located in 4-byte autonomous system numbers. The address family configured here is the default IPv4 unicast address family, and the configuration is done at Router B in the figure above (in the "Cisco Implementation of 4-Byte Autonomous System Numbers" section). The 4-byte autonomous system numbers in this task are formatted in the default asplain (decimal value) format; for example, Router B is in autonomous system number 65538 in the figure above. Remember to perform this task for any neighbor routers that are to be BGP peers.

For more details about 4-byte autonomous system number formats, see the BGP Autonomous System Number Formats, page 25 and the Cisco Implementation of 4-Byte Autonomous System Numbers, page 28.

This task requires Cisco IOS Release 12.0(32)SY8, 12.2(33)SXI1, or a later release to be running on the router.

Note By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. Repeat Step 4 to define other BGP neighbors, as required.
6. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
7. **neighbor** *ip-address* **activate**
8. Repeat Step 7 to activate other BGP neighbors, as required.
9. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
10. **end**
11. **show ip bgp** [*network*] [*network-mask*]
12. **show ip bgp summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 65538`	Enters router configuration mode for the specified routing process. • In this example, the 4-byte autonomous system number, 65538, is defined in asplain notation.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 65536`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • In this example, the 4-byte autonomous system number, 65536, is defined in asplain notation.
Step 5	Repeat Step 4 to define other BGP neighbors, as required.	--
Step 6	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the virtual routing and forwarding (VRF) instance to associate with subsequent IPv4 address family configuration mode commands.
Step 7	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.2 activate`	Enables the neighbor to exchange prefixes for the IPv4 unicast address family with the local router.
Step 8	Repeat Step 7 to activate other BGP neighbors, as required.	--

	Command or Action	Purpose
Step 9	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Router(config-router)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.
Step 10	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 11	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp 10.1.1.0`	(Optional) Displays the entries in the BGP routing table. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 12	**show ip bgp summary** **Example:** `Router# show ip bgp summary`	(Optional) Displays the status of all BGP connections.

Examples

The following output from the **show ip bgp** command at Router B shows the BGP routing table entry for network 10.1.1.0 learned from the BGP neighbor at 192.168.1.2 in Router A in the figure above with its 4-byte autonomous system number of 65536 displayed in the default asplain format.

```
RouterB# show ip bgp 10.1.1.0
BGP routing table entry for 10.1.1.0/24, version 2
Paths: (1 available, best #1)
  Advertised to update-groups:
     2
  65536
    192.168.1.2 from 192.168.1.2 (10.1.1.99)
      Origin IGP, metric 0, localpref 100, valid, external, best
```

The following output from the **show ip bgp summary** command shows the 4-byte autonomous system number 65536 for the BGP neighbor 192.168.1.2 of Router A in the figure above after this task has been configured on Router B:

```
RouterB# show ip bgp summary
BGP router identifier 172.17.1.99, local AS number 65538
BGP table version is 3, main routing table version 3
2 network entries using 234 bytes of memory
2 path entries using 104 bytes of memory
3/2 BGP path/bestpath attribute entries using 444 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
```

```
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 806 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 60 secs
Neighbor        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  Stated
192.168.1.2     4       65536       6       6        3    0    0 00:01:33       1
```

• Troubleshooting Tips, page 50

Troubleshooting Tips

Use the **ping** command to verify basic network connectivity between the BGP routers.

Modifying the Default Output and Regular Expression Match Format for 4-Byte Autonomous System Numbers

Perform this task to modify the default output format for 4-byte autonomous system numbers from asplain format to asdot notation format. The **show ip bgp summary** command is used to display the changes in output format for the 4-byte autonomous system numbers.

For more details about 4-byte autonomous system number formats, see the BGP Autonomous System Number Formats, page 25.

This example requires Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, or a later release, to be running on the router.

SUMMARY STEPS

1. **enable**
2. **show ip bgp summary**
3. **configure terminal**
4. **router bgp** *autonomous-system-number*
5. **bgp asnotation dot**
6. **end**
7. **clear ip bgp ***
8. **show ip bgp summary**
9. **show ip bgp regexp** *regexp*
10. **configure terminal**
11. **router bgp** *autonomous-system-number*
12. **no bgp asnotation dot**
13. **end**
14. **clear ip bgp ***

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip bgp summary** **Example:** `Router# show ip bgp summary`	Displays the status of all BGP connections.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 65538`	Enters router configuration mode for the specified routing process. • In this example, the 4-byte autonomous system number, 65538, is defined in asplain notation.
Step 5	**bgp asnotation dot** **Example:** `Router(config-router)# bgp asnotation dot`	Changes the default output format of BGP 4-byte autonomous system numbers from asplain (decimal values) to dot notation. **Note** 4-byte autonomous system numbers can be configured using either asplain format or asdot format. This command affects only the output displayed for **show** commands or the matching of regular expressions.
Step 6	**end** **Example:** `Router(config-router)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 7	**clear ip bgp *** **Example:** `Router# clear ip bgp *`	Clears and resets all current BGP sessions. • In this example, a hard reset is performed to ensure that the 4-byte autonomous system number format change is reflected in all BGP sessions. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 8	**show ip bgp summary** **Example:** `Router# show ip bgp summary`	Displays the status of all BGP connections.
Step 9	**show ip bgp regexp** *regexp* **Example:** `Router# show ip bgp regexp ^1\.0$`	Displays routes that match the autonomous system path regular expression. • In this example, a regular expression to match a 4-byte autonomous system path is configured using asdot format.
Step 10	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 11	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 65538`	Enters router configuration mode for the specified routing process. • In this example, the 4-byte autonomous system number, 65538, is defined in asplain notation.
Step 12	**no bgp asnotation dot** **Example:** `Router(config-router)# no bgp asnotation dot`	Resets the default output format of BGP 4-byte autonomous system numbers back to asplain (decimal values). **Note** 4-byte autonomous system numbers can be configured using either asplain format or asdot format. This command affects only the output displayed for **show** commands or the matching of regular expressions.
Step 13	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 14	**clear ip bgp *** **Example:** `Router# clear ip bgp *`	Clears and resets all current BGP sessions. • In this example, a hard reset is performed to ensure that the 4-byte autonomous system number format change is reflected in all BGP sessions. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following output from the **show ip bgp summary** command shows the default asplain format of the 4-byte autonomous system numbers. Note the asplain format of the 4-byte autonomous system numbers, 65536 and 65550.

```
Router# show ip bgp summary
BGP router identifier 172.17.1.99, local AS number 65538
BGP table version is 1, main routing table version 1
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  Statd
192.168.1.2     4       65536       7       7        1    0    0 00:03:04      0
192.168.3.2     4       65550       4       4        1    0    0 00:00:15      0
```

After the **bgp asnotation dot** command is configured (followed by the **clear ip bgp *** command to perform a hard reset of all current BGP sessions), the output is converted to asdot notation format as shown in the following output from the **show ip bgp summary** command. Note the asdot format of the 4-byte autonomous system numbers, 1.0 and 1.14 (these are the asdot conversions of the 65536 and 65550 autonomous system numbers.

```
Router# show ip bgp summary
BGP router identifier 172.17.1.99, local AS number 1.2
BGP table version is 1, main routing table version 1
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  Statd
192.168.1.2     4         1.0       9       9        1    0    0 00:04:13      0
192.168.3.2     4        1.14       6       6        1    0    0 00:01:24      0
```

After the **bgp asnotation dot** command is configured (followed by the **clear ip bgp *** command to perform a hard reset of all current BGP sessions), the regular expression match format for 4-byte autonomous system paths is changed to asdot notation format. Although a 4-byte autonomous system number can be configured in a regular expression using either asplain format or asdot format, only 4-byte autonomous system numbers configured using the current default format are matched. In the first example below, the **show ip bgp regexp**command is configured with a 4-byte autonomous system number in asplain format. The match fails because the default format is currently asdot format and there is no output. In the second example using asdot format, the match passes and the information about the 4-byte autonomous system path is shown using the asdot notation.

Note The asdot notation uses a period which is a special character in Cisco regular expressions. To remove the special meaning, use a backslash before the period.

```
Router# show ip bgp regexp ^65536$
Router# show ip bgp regexp ^1\.0$
BGP table version is 2, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
```

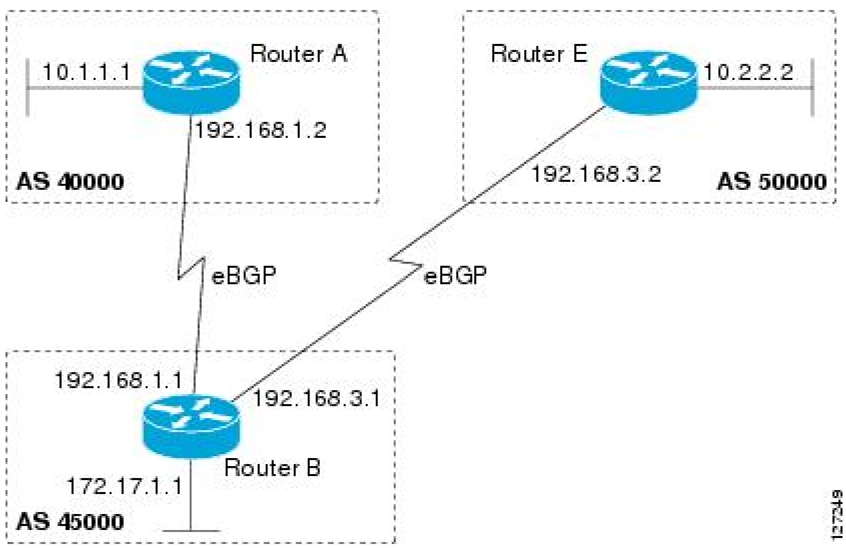
```
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 1.0 i
```

Configuring a BGP Peer for the IPv4 VRF Address Family

Perform this optional task to configure BGP between two IPv4 routers (peers) that must exchange IPv4 VRF information because they exist in a VPN. The address family configured here is the IPv4 VRF address family and the configuration is done at Router B in the figure below with the neighbor 192.168.3.2 at Router E in autonomous system 50000. Remember to perform this task for any neighbor routers that are to be BGP IPv4 VRF address family peers.

This task does not show the complete configuration required for VPN routing. For some complete example configurations and an example configuration showing how to create a VRF with a route-target that uses a 4-byte autonomous system number, see .

Figure 8 BGP Topology for IPv4 VRF Address Family



Before you perform this task, perform the Configuring a BGP Routing Process, page 40 task.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **rd** *route-distinguisher*
5. **route-target** {**import** | **export**| **both**} *route-target-ext-community*
6. **exit**
7. **router bgp** *autonomous-system-number*
8. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
9. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
10. **neighbor** {*ip-address*| *peer-group-name*} **maximum-prefix** *maximum* [*threshold*] [**restart** *restart-interval*] [**warning-only**]
11. **neighbor** *ip-address* **activate**
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf vpn1`	Configures a VRF routing table and enters VRF configuration mode. • Use the *vrf-name* argument to specify a name to be assigned to the VRF.
Step 4	**rd** *route-distinguisher* **Example:** `Router(config-vrf)# rd 45000:5`	Creates routing and forwarding tables and specifies the default route distinguisher for a VPN. • Use the *route-distinguisher* argument to add an 8-byte value to an IPv4 prefix to create a unique VPN IPv4 prefix.

	Command or Action	Purpose
Step 5	**route-target** {**import** | **export**| **both**} *route-target-ext-community* **Example:** `Router(config-vrf)# route-target both 45000:100`	Creates a route target extended community for a VRF. • Use the **import** keyword to import routing information from the target VPN extended community. • Use the **export** keyword to export routing information to the target VPN extended community. • Use the **both** keyword to import both import and export routing information to the target VPN extended community. • Use the *route-target-ext-community* argument to add the route target extended community attributes to the VRF's list of import, export, or both (import and export) route target extended communities.
Step 6	**exit** **Example:** `Router(config-vrf)# exit`	Exits VRF configuration mode and enters global configuration mode.
Step 7	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 8	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 vrf vpn1`	Specifies the IPv4 address family and enters address family configuration mode. • Use the **unicast** keyword to specify the IPv4 unicast address family. By default, the router is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • Use the **multicast** keyword to specify IPv4 multicast address prefixes. • Use the **vrf** keyword and *vrf-name* argument to specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 9	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.3.2 remote-as 45000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.

	Command or Action	Purpose
Step 10	**neighbor** {*ip-address*| *peer-group-name*} **maximum-prefix** *maximum* [*threshold*] [**restart** *restart-interval*] [**warning-only**] **Example:** `Router(config-router-af)# neighbor 192.168.3.2 maximum-prefix 10000 warning-only`	Controls how many prefixes can be received from a neighbor. • Use the *maximum* argument to specify the maximum number of prefixes allowed from the specified neighbor. The number of prefixes that can be configured is limited only by the available system resources on a router. • Use the *threshold* argument to specify an integer representing a percentage of the maximum prefix limit at which the router starts to generate a warning message. • Use the **warning-only** keyword to allow the router to generate a log message when the maximum prefix limit is exceeded, instead of terminating the peering session.
Step 11	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 192.168.3.2 activate`	Enables the neighbor to exchange prefixes for the IPv4 VRF address family with the local router.
Step 12	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.

• Troubleshooting Tips, page 57

Troubleshooting Tips

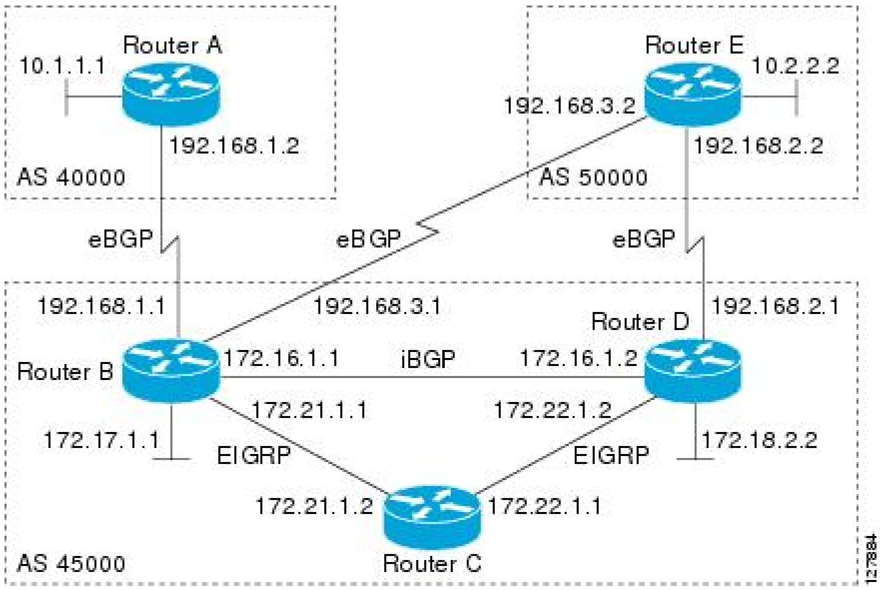
Use the **ping** command to verify basic network connectivity between the BGP routers, and use the **show ip vrf** command to verify that the VRF instance has been created.

Customizing a BGP Peer

Perform this task to customize your BGP peers. Although many of the steps in this task are optional, this task demonstrates how the neighbor and address family configuration command relationships work. Using the example of the IPv4 multicast address family, neighbor address family-independent commands are configured before the IPv4 multicast address family is configured. Commands that are address family-dependent are then configured and the **exit address-family** command is shown. An optional step shows how to disable a neighbor.

The configuration in this task is done at Router B in the figure below and would need to be repeated with appropriate changes to the IP addresses, for example, at Router E to fully configure a BGP process between the two routers.

Figure 9 ***BGP Peer Topology***



Note By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types, such as IPv6 prefixes.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **no bgp default ipv4-unicast**
5. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
6. **neighbor** {*ip-address* | *peer-group-name*} **description** *text*
7. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
8. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
9. **neighbor** {*ip-address* | *peer-group-name*} **activate**
10. **neighbor** {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds*
11. **neighbor** {*ip-address* | *peer-group-name*} **default-originate**[**route-map** *map-name*]
12. **exit-address-family**
13. **neighbor** {*ip-address* | *peer-group-name*} **shutdown**
14. **end**
15. **show ip bgp ipv4 multicast** [*command*]
16. **show ip bgp neighbors** [*neighbor-address*] [**received-routes** | **routes** | **advertised-routes** | **paths** *regexp* | **dampened-routes** | **received prefix-filter**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.

	Command or Action	Purpose
Step 4	**no bgp default ipv4-unicast** **Example:** `Router(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured with the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast**router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **description** *text* **Example:** `Router(config-router)# neighbor 192.168.3.2 description finance`	(Optional) Associates a text description with the specified neighbor.
Step 7	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 multicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 8	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.

	Command or Action	Purpose
Step 9	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.3.2 activate`	Enables the exchange of information with a BGP neighbor.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds* **Example:** `Router(config-router-af)# neighbor 192.168.3.2 advertisement-interval 25`	(Optional) Sets the minimum interval between the sending of BGP routing updates.
Step 11	**neighbor** {*ip-address* | *peer-group-name*} **default-originate**[**route-map** *map-name*] **Example:** `Router(config-router-af)# neighbor 192.168.3.2 default-originate`	(Optional) Permits a BGP speaker--the local router--to send the default route 0.0.0.0 to a peer for use as a default route.
Step 12	**exit-address-family** **Example:** `Router(config-router-af)# exit-address-family`	Exits address family configuration mode and enters router configuration mode.
Step 13	**neighbor** {*ip-address* | *peer-group-name*} **shutdown** **Example:** `Router(config-router)# neighbor 192.168.3.2 shutdown`	(Optional) Disables a BGP peer or peer group. **Note** If you perform this step you will not be able to run either of the subsequent **show** command steps because you have disabled the neighbor.
Step 14	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 15	**show ip bgp ipv4 multicast** [*command*] **Example:** `Router# show ip bgp ipv4 multicast`	(Optional) Displays IPv4 multicast database-related information. • Use the *command* argument to specify any multiprotocol BGP command that is supported. To see the supported commands, use the ? prompt on the CLI.
Step 16	**show ip bgp neighbors** [*neighbor-address*] [**received-routes** | **routes** | **advertised-routes** | **paths** *regexp* | **dampened-routes** | **received prefix-filter**]] **Example:** `Router# show ip bgp neighbors 192.168.3.2`	(Optional) Displays information about the TCP and BGP connections to neighbors.

Examples

The following sample output from the **show ip bgp ipv4 multicast** command shows BGP IPv4 multicast information for Router B in the figure above after this task has been configured on Router B and Router E. Note that the networks local to each router that were configured under IPv4 multicast address family appear in the output table.

```
BGP table version is 3, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.2.0/24      192.168.3.2              0             0 50000 i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
```

The following partial sample output from the **show ip bgp neighbors**command for neighbor 192.168.3.2 shows general BGP information and specific BGP IPv4 multicast address family information about the neighbor. The command was entered on Router B in the figure above after this task had been configured on Router B and Router E.

```
BGP neighbor is 192.168.3.2,  remote AS 50000, external link
 Description: finance
  BGP version 4, remote router ID 10.2.2.99
  BGP state = Established, up for 01:48:27
  Last read 00:00:26, last write 00:00:26, hold time is 120, keepalive intervals
  Configured hold time is 120,keepalive interval is 70 seconds, Minimum holdtims
  Neighbor capabilities:
    Route refresh: advertised and received (old & new)
    Address family IPv4 Unicast: advertised
    Address family IPv4 Multicast: advertised and received
!
 For address family: IPv4 Multicast
  BGP table version 3, neighbor version 3/0
 Output queue size : 0
  Index 1, Offset 0, Mask 0x2
  1 update-group member
    Uses NEXT_HOP attribute for MBGP NLRIs
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               1          1 (Consumes 48 bytes)
    Prefixes Total:                 1          1
    Implicit Withdraw:              0          0
```

```
    Explicit Withdraw:              0          0
    Used as bestpath:             n/a          1
    Used as multipath:            n/a          0
                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Bestpath from this peer:             1        n/a
    Total:                               1          0
  Number of NLRIs in the update sent: max 0, min 0
  Minimum time between advertisement runs is 25 seconds
  Connections established 8; dropped 7
  Last reset 01:48:54, due to User reset
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled
Local host: 192.168.3.1, Local port: 13172
Foreign host: 192.168.3.2, Foreign port: 179
!
```

Removing BGP Configuration Commands Using a Redistribution

BGP CLI configuration can become quite complex even in smaller BGP networks. If you need to remove any CLI configuration, you must consider all the implications of removing the CLI. Analyze the current running configuration to determine the current BGP neighbor relationships, any address family considerations, and even other routing protocols that are configured. Many BGP CLI commands affect other parts of the CLI configuration.

Perform this task to remove all the BGP configuration commands used in a redistribution of BGP routes into EIGRP. A route map can be used to match and set parameters or to filter the redistributed routes to ensure that routing loops are not created when these routes are subsequently advertised by EIGRP. When removing BGP configuration commands you must remember to remove or disable all the related commands. In this example, if the **route-map** command is omitted, then the redistribution will still occur and possibly with unexpected results as the route map filtering has been removed. Omitting just the **redistribute** command would mean that the route map is not applied, but it would leave unused commands in the running configuration.

For more details on BGP CLI removal, see the "BGP CLI Removal Considerations" concept in the "Cisco BGP Overview" module.

To view the redistribution configuration before and after the CLI removal, see the Examples Removing BGP Configuration Commands Using a Redistribution Example, page 111.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no route-map** *map-name*
4. **router eigrp** *autonomous-system-number*
5. **no redistribute** *protocol* [*as-number*]
6. **end**
7. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**no route-map** *map-name* **Example:** `Router(config)# no route-map bgp-to-eigrp`	Removes a route map from the running configuration. • In this example, a route map named bgp-to-eigrp is removed from the configuration.
Step 4	**router eigrp** *autonomous-system-number* **Example:** `Router(config)# router eigrp 100`	Enters router configuration mode for the specified routing process.
Step 5	**no redistribute** *protocol* [*as-number*] **Example:** `Router(config-router)# no redistribute bgp 45000`	Disables the redistribution of routes from one routing domain into another routing domain. • In this example, the configuration of the redistribution of BGP routes into the EIGRP routing process is removed from the running configuration. **Note** If a route map was included in the original **redistribute** command configuration, remember to remove the **route-map** command configuration as in Step 3 in this example task. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 6	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 7	**show running-config** **Example:** `Router# show running-config`	(Optional) Displays the current running configuration on the router. • Use this command to verify that the **redistribute** and **route-map** commands are removed from the router configuration.

Monitoring and Maintaining Basic BGP

The tasks in this section are concerned with the resetting and display of information about basic BGP processes and peer relationships. Once you have defined two routers to be BGP neighbors, they will form a BGP connection and exchange routing information. If you subsequently change a BGP filter, weight, distance, version, or timer, or make a similar configuration change, you may have to reset BGP connections for the configuration change to take effect.

- Configuring Inbound Soft-Reconfiguration When Route Refresh Capability Is Missing, page 65
- Resetting and Displaying Basic BGP Information, page 68

Configuring Inbound Soft-Reconfiguration When Route Refresh Capability Is Missing

Perform this task to configure inbound soft reconfiguration using the **bgp soft-reconfig-backup** command for BGP peers that do not support the route refresh capability. BGP Peers that support the route refresh capability are unaffected by the configuration of this command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp log-neighbor-changes**
5. **bgp soft-reconfig-backup**
6. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
7. **neighbor** {*ip-address* | *peer-group-name*} **soft-reconfiguration**[**inbound**]
8. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
9. Repeat Steps 6 through 8 for every peer that is to be configured with soft-reconfiguration inbound.
10. **exit**
11. **route-map** *map-name* [**permit**| **deny**][**sequence-number**]
12. **set local-preference** *number-value*
13. **end**
14. **show ip bgp neighbors** [*neighbor-address*]
15. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 5	**bgp soft-reconfig-backup** **Example:** `Router(config-router)# bgp soft-reconfig-backup`	Configures a BGP speaker to perform inbound soft reconfiguration for peers that do not support the route refresh capability. • This command is used to configure BGP to perform inbound soft reconfiguration for peers that do not support the route refresh capability. The configuration of this command allows you to configure BGP to store updates (soft reconfiguration) only as necessary. Peers that support the route refresh capability are unaffected by the configuration of this command.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.

	Command or Action	Purpose
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **soft-reconfiguration**[**inbound**] **Example:** `Router(config-router)# neighbor 192.168.1.2 soft-reconfiguration inbound`	Configures the Cisco IOS software to start storing updates. • All the updates received from this neighbor will be stored unmodified, regardless of the inbound policy. When inbound soft reconfiguration is done later, the stored information will be used to generate a new set of inbound updates.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**} **Example:** `Router(config-router)# neighbor 192.168.1.2 route-map LOCAL in`	Applies a route map to incoming or outgoing routes. • In this example, the route map named LOCAL will be applied to incoming routes.
Step 9	Repeat Steps 6 through 8 for every peer that is to be configured with soft-reconfiguration inbound.	--
Step 10	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 11	**route-map** *map-name* [**permit**| **deny**][**sequence-number**] **Example:** `Router(config)# route-map LOCAL permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named LOCAL is created.
Step 12	**set local-preference** *number-value* **Example:** `Router(config-route-map)# set local-preference 200`	Specifies a preference value for the autonomous system path. • In this example, the local preference value is set to 200.
Step 13	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 14	**show ip bgp neighbors** [*neighbor-address*] **Example:** `Router(config-router-af)# show ip bgp neighbors 192.168.1.2`	(Optional) Displays information about the TCP and BGP connections to neighbors. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 15	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp`	(Optional) Displays the entries in the BGP routing table. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following partial output from the **show ip bgp neighbors** command shows information about the TCP and BGP connections to the BGP neighbor 192.168.2.1. This peer supports route refresh.

```
BGP neighbor is 192.168.1.2,  remote AS 40000, external link
 Neighbor capabilities:
   Route refresh: advertised and received(new)
```

The following partial output from the **show ip bgp neighbors** command shows information about the TCP and BGP connections to the BGP neighbor 192.168.3.2. This peer does not support route refresh so the soft-reconfig inbound paths for BGP peer 192.168.3.2 will be stored because there is no other way to update any inbound policy updates.

```
BGP neighbor is 192.168.3.2,  remote AS 50000, external link
 Neighbor capabilities:
   Route refresh: advertised
```

The following sample output from the **show ip bgp** command shows the entry for the network 172.17.1.0. Both BGP peers are advertising 172.17.1.0/24 but only the received-only path is stored for 192.168.3.2.

```
BGP routing table entry for 172.17.1.0/24, version 11
Paths: (3 available, best #3, table Default-IP-Routing-Table, RIB-failure(4))
Flag: 0x820
 Advertised to update-groups:
    1
 50000
   192.168.3.2 from 192.168.3.2 (172.17.1.0)
     Origin incomplete, metric 0, localpref 200, valid, external
 50000, (received-only)
   192.168.3.2 from 192.168.3.2 (172.17.1.0)
     Origin incomplete, metric 0, localpref 100, valid, external
 40000
   192.168.1.2 from 192.168.1.2 (172.16.1.0)
     Origin incomplete, metric 0, localpref 200, valid, external, best
```

Resetting and Displaying Basic BGP Information

Perform this task to reset and display information about basic BGP processes and peer relationships.

SUMMARY STEPS

1. **enable**
2. **clear ip bgp** {***** | *autonomous-system-number* | *neighbor-address*}} [**soft** [**in** | **out**]
3. **show ip bgp** [*network-address*][*network-mask*] [**longer-prefixes**] [**prefix-list** *prefix-list-name* | **route-map** *route-map-name*] [**shorter prefixes** *mask-length*]
4. **show ip bgp neighbors** [*neighbor-address*] [**received-routes** | **routes** | **advertised-routes** | **paths** *regexp* | **dampened-routes** | **received** *prefix-filter*]]
5. **show ip bgp paths**
6. **show ip bgp summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear ip bgp** {*****	*autonomous-system-number*	*neighbor-address*}} [**soft** [**in**	**out**] **Example:** `Router#` **clear ip bgp ***	Clears and resets BGP neighbor sessions: • In the example provided, all BGP neighbor sessions are cleared and reset.
Step 3	**show ip bgp** [*network-address*][*network-mask*] [**longer-prefixes**] [**prefix-list** *prefix-list-name*	**route-map** *route-map-name*] [**shorter prefixes** *mask-length*] **Example:** `Router#` **show ip bgp 10.1.1.0 255.255.255.0**	Displays all the entries in the BGP routing table: • In the example provided, the BGP routing table information for the 10.1.1.0 network is displayed.
Step 4	**show ip bgp neighbors** [*neighbor-address*] [**received-routes**	**routes**	**advertised-routes**	**paths** *regexp*	**dampened-routes**	**received** *prefix-filter*]] **Example:** `Router#` **show ip bgp neighbors 192.168.3.2 advertised-routes**	Displays information about the TCP and BGP connections to neighbors. • In the example provided, the routes advertised from the router to BGP neighbor 192.168.3.2 on another router are displayed.

	Command or Action	Purpose
Step 5	**show ip bgp paths** **Example:** `Router#` **show ip bgp paths**	Displays information about all the BGP paths in the database.
Step 6	**show ip bgp summary** **Example:** `Router#` **show ip bgp summary**	Displays information about the status of all BGP connections.

Aggregating Route Prefixes Using BGP

BGP peers exchange information about local networks but this can quickly lead to large BGP routing tables. CIDR enables the creation of aggregate routes (or *supernets*) to minimize the size of routing tables. Smaller BGP routing tables can reduce the convergence time of the network and improve network performance. Aggregated routes can be configured and advertised using BGP. Some aggregations advertise only summary routes and other methods of aggregating routes allow more specific routes to be forwarded. Aggregation applies only to routes that exist in the BGP routing table. An aggregated route is forwarded if at least one more specific route of the aggregation exists in the BGP routing table. Perform one of the following tasks to aggregate routes within BGP:

- Redistributing a Static Aggregate Route into BGP, page 70
- Configuring Conditional Aggregate Routes Using BGP, page 72
- Suppressing and Unsuppressing Advertising Aggregated Routes Using BGP, page 73
- Suppressing Inactive Route Advertisement Using BGP, page 75
- Conditionally Advertising BGP Routes, page 77

Redistributing a Static Aggregate Route into BGP

Use this task to redistribute a static aggregate route into BPG. A static aggregate route is configured and then redistributed into the BGP routing table. The static route must be configured to point to interface null 0 and the prefix should be a superset of known BGP routes. When a router receives a BGP packet it will use the more specific BGP routes. If the route is not found in the BGP routing table, then the packet will be forwarded to null 0 and discarded.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip route** *prefix mask* {*ip-address* | *interface-type interface-number* [*ip-address*]} [*distance*] [*name*] [**permanent** | **track** *number*] [**tag** *tag*]
4. **router bgp** *autonomous-system-number*
5. **redistribute static**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip route** *prefix mask* {*ip-address* | *interface-type interface-number* [*ip-address*]} [*distance*] [*name*] [**permanent** | **track** *number*] [**tag** *tag*] **Example:** `Router(config)# ip route 172.0.0.0 255.0.0.0 null 0`	Creates a static route.
Step 4	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 5	**redistribute static** **Example:** `Router(config-router)# redistribute static`	Redistributes routes into the BGP routing table.

	Command or Action	Purpose
Step 6	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Configuring Conditional Aggregate Routes Using BGP

Use this task to create an aggregate route entry in the BGP routing table when at least one specific route falls into the specified range. The aggregate route is advertised as originating from your autonomous system. For more information, see the BGP Aggregation Route AS-SET Information Generation, page 31.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **aggregate-address** *address mask* [**as-set**]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.

	Command or Action	Purpose
Step 4	**aggregate-address** *address mask* [**as-set**] **Example:** `Router(config-router)# aggregate-address 172.0.0.0 255.0.0.0 as-set`	Creates an aggregate entry in a BGP routing table. • A specified route must exist in the BGP table. • Use the **aggregate-address** command with no keywords to create an aggregate entry if any more-specific BGP routes are available that fall in the specified range. • Use the **as-set** keyword to specify that the path advertised for this route is an AS-SET. Do not use the **as-set** keyword when aggregating many paths because this route is withdrawn and updated every time the reachability information for the aggregated route changes. **Note** Only partial syntax is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 5	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

Suppressing and Unsuppressing Advertising Aggregated Routes Using BGP

Use this task to create an aggregate route, suppress the advertisement of routes using BGP, and subsequently unsuppress the advertisement of routes. Routes that are suppressed are not advertised to any neighbors, but it is possible to unsuppress routes that were previously suppressed to specific neighbors.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. Do one of the following:
    - **aggregate-address** *address mask* [**summary-only**]
    -
    - **aggregate-address** *address mask* [**suppress-map** *map-name*]
6. **neighbor** {*ip-address* | *peer-group-name*} **unsuppress-map** *map-name*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.

Step 5	Do one of the following:
	
	• **aggregate-address** *address mask* [**summary-only**]
	•
	• **aggregate-address** *address mask* [**suppress-map** *map-name*]
	
	Example:
	
	`Router(config-router)# aggregate-address 172.0.0.0 255.0.0.0 summary-only`
	
	Example:
	
	`Router(config-router)# aggregate-address 172.0.0.0 255.0.0.0 suppress-map map1`
		Creates an aggregate route.
		
		• Use the optional **summary-only** keyword to create the aggregate route (for example, 10.*.*.*) and also suppresses advertisements of more-specific routes to all neighbors.
		• Use the optional **suppress-map**keyword to create the aggregate route but suppress advertisement of specified routes. Routes that are suppressed are not advertised to any neighbors. You can use the **match** clauses of route maps to selectively suppress some more-specific routes of the aggregate and leave others unsuppressed. IP access lists and autonomous system path access lists **match** clauses are supported.
		
		Note Only partial syntax is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

	Command or Action	Purpose
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **unsuppress-map** *map-name* **Example:** `Router(config-router)# neighbor 192.168.1.2 unsuppress map1`	(Optional) Selectively advertises routes previously suppressed by the **aggregate-address** command. • In this example, the routes previously suppressed in Step 5 are advertised to neighbor 192.168.1.2.
Step 7	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

Suppressing Inactive Route Advertisement Using BGP

Perform this task to suppress the advertisement of inactive routes by BGP. In Cisco IOS Release 12.2(25)S, 12.2(33)SXH, and 15.0(1)M, the **bgp suppress-inactive** command was introduced to configure BGP to not advertise inactive routes to any BGP peer. A BGP routing process can advertise routes that are not installed in the RIB to BGP peers by default. A route that is not installed into the RIB is an inactive route. Inactive route advertisement can occur, for example, when routes are advertised through common route aggregation.

Inactive route advertisements can be suppressed to provide more consistent data forwarding. This feature can be configured on a per IPv4 address family basis. For example, when specifying the maximum number of routes that can be configured in a VRF with the **maximum routes** global configuration command, you also suppress inactive route advertisement to prevent inactive routes from being accepted into the VRF after route limit has been exceeded.

This task assumes that BGP is enabled and that peering has been established.

Note Inactive route suppression can be configured only under the IPv4 address family or under a default IPv4 general session.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
5. **bgp suppress-inactive**
6. **end**
7. **show ip bgp rib-failure**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 unicast`	Enter address family configuration mode to configure BGP peers to accept address family specific configurations. • The example creates an IPv4 unicast address family session.
Step 5	**bgp suppress-inactive** **Example:** `Router(config-router-af)# bgp suppress-inactive`	Suppresses BGP advertising of inactive routes. • BGP advertises inactive routes by default. • Entering the **no** form of this command reenables the advertisement of inactive routes.
Step 6	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.
Step 7	**show ip bgp rib-failure** **Example:** `Router# show ip bgp rib-failure`	(Optional) Displays BGP routes that are not installed in the RIB.

Examples

The following example shows output from the **show ip bgp rib-failure** command displaying routes that are not installed in the RIB. The output shows that the displayed routes were not installed because a route or routes with a better administrative distance already exist in the RIB.

```
Router# show ip bgp rib-failure

Network            Next Hop                      RIB-failure   RIB-NH Matches
10.1.15.0/24       10.1.35.5           Higher admin distance              n/a
10.1.16.0/24       10.1.15.1           Higher admin distance              n/a
```

Conditionally Advertising BGP Routes

Perform this task to conditionally advertise selected BGP routes. The routes or prefixes that will be conditionally advertised are defined in two route maps: an **advertise map** and either an **exist map** or **nonexist map**. The route map associated with the **exist map** or **nonexist map** specifies the prefix that the BGP speaker will track. The route map associated with the **advertise map** specifies the prefix that will be advertised to the specified neighbor when the condition is met.

- If a prefix is found to be present in the exist map by the BGP speaker, then the prefix specified by the advertise map is advertised.
- If a prefix is found not to be present in the nonexist map by the BGP speaker, then the prefix specified by the advertise map is advertised.

If the condition is not met, the route is withdrawn and conditional advertisement does not occur. All routes that may be dynamically advertised or not advertised need to exist in the BGP routing table for conditional advertisement to occur. These routes are referenced from an access list or an IP prefix list.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. **neighbor** *ip-address* **advertise-map** *map-name* {**exist-map** *map-name* | **non-exist-map** *map-name*}
6. **exit**
7. **route-map** *map-tag* [**permit**| **deny**][**sequence-number**]
8. **match ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [*access-list-number*...| *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]}
9. **route-map** *map-tag* [**permit**| **deny**][**sequence-number**]
10. **match ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [*access-list-number*...| *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]}
11. **exit**
12. **access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**]
13. **access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**]
14. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 5	**neighbor** *ip-address* **advertise-map** *map-name* {**exist-map** *map-name* | **non-exist-map** *map-name*} **Example:** `Router(config-router)# neighbor 192.168.1.2 advertise-map map1 exist-map map2`	Adds the IP address of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • In this example, the prefix (172.17.0.0) matching the ACL in the advertise map (route map named map1) will be advertised to neighbor only when a prefix (192.168.50.0) matching the ACL in exist map (route-map "map2") is in the local BGP table.
Step 6	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 7	**route-map** *map-tag* [**permit**| **deny**][**sequence-number**] **Example:** `Router(config)# route-map map1 permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named map1 is created.
Step 8	**match ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...*| *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]} **Example:** `Router(config-route-map)# match ip address 1`	Configures the route map to match a prefix that is permitted by a standard access list, an extended access list, or a prefix list. • In this example, the route map is configured to match a prefix permitted by access list 1.
Step 9	**route-map** *map-tag* [**permit**| **deny**][**sequence-number**] **Example:** `Router(config)# route-map map2 permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named map2 is created.
Step 10	**match ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...*| *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]} **Example:** `Router(config-route-map)# match ip address 2`	Configures the route map to match a prefix that is permitted by a standard access list, an extended access list, or a prefix list. • In this example, the route map is configured to match a prefix permitted by access list 2.
Step 11	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 12	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**] **Example:** `Router(config)# access-list 1 permit 172.17.0.0`	Configures a standard access list. • In this example, access list 1 permits advertising of the 172.17.0.0 prefix, depending on other conditions set by the **neighbor advertise-map** command.

	Command or Action	Purpose
Step 13	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**] **Example:** `Router(config)# access-list 2 permit 192.168.50.0`	Configures a standard access list. • In this example, access list 2 permits the 192.168.50.0 to be the prefix of the exist-map.
Step 14	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Originating BGP Routes

Route aggregation is useful to minimize the size of the BGP table but there are situations when you want to add more specific prefixes to the BGP table. Route aggregation can hide more specific routes. Using the **network** command as shown in the "Configuring a BGP Routing Process" section originates routes, and the following optional tasks originate BGP routes for the BGP table for different situations.

- Advertising a Default Route Using BGP, page 80
- Conditionally Injecting BGP Routes, page 82
- Originating BGP Routes Using Backdoor Routes, page 86

Advertising a Default Route Using BGP

Perform this task to advertise a default route to BGP peers. The default route is locally originated. A default route can be useful to simplify configuration or to prevent the router from using too many system resources. If the router is peered with an Internet service provider (ISP), the ISP will carry full routing tables, so configuring a default route into the ISP network saves resources at the local router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length*| **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*]
4. **route-map** *map-tag* [**permit**| **deny**][**sequence-number**]
5. **match ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...* | *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]}
6. **exit**
7. **router bgp** *autonomous-system-number*
8. **neighbor** {*ip-address* | *peer-group-name*} **default-originate**[**route-map** *map-name*]
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length*| **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list DEFAULT permit 10.1.1.0/24`	Configures an IP prefix list. • In this example, prefix list DEFAULT permits advertising of the 10.1.1.0/24. prefix depending on a match set by the **match ip address** command.
Step 4	**route-map** *map-tag* [**permit**| **deny**][**sequence-number**] **Example:** `Router(config)# route-map ROUTE`	Configures a route map and enters route map configuration mode. • In this example, a route map named ROUTE is created.
Step 5	**match ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [*access-list-number*... | *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]} **Example:** `Router(config-route-map)# match ip address prefix-list DEFAULT`	Configures the route map to match a prefix that is permitted by a standard access list, an extended access list, or a prefix list. • In this example, the route map is configured to match a prefix permitted by prefix list DEFAULT.
Step 6	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 7	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **default-originate**[**route-map** *map-name*] **Example:** `Router(config-router)# neighbor 192.168.3.2 default-originate`	(Optional) Permits a BGP speaker--the local router--to send the default route 0.0.0.0 to a peer for use as a default route.
Step 9	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

- Troubleshooting Tips, page 82

Troubleshooting Tips

Use the **show ip route** command on the receiving BGP peer (not on the local router) to verify that the default route has been set. In the output, verify that a line similar to the following showing the default route 0.0.0.0 is present:

```
B*    0.0.0.0/0 [20/0] via 192.168.1.2, 00:03:10
```

Conditionally Injecting BGP Routes

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

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp inject-map** *inject-map-name* **exist-map** *exist-map-name* [**copy-attributes**]
5. **exit**
6. **route-map** *map-tag* [**permit**| **deny**][*sequence-number*]
7. **match ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [*access-list-number*... | *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]}
8. **match ip route-source** {*access-list-number* | *access-list-name*} [*access-list-number*...| *access-list-name*...]
9. **exit**
10. **route-map** *map-tag* [**permit**| **deny**][*sequence-number*]
11. **set ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [access-list-number... | access-list-name] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]}
12. **set community** {*community-number* [**additive**] [*well-known-community*] | **none**}
13. **exit**
14. **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*]
15. Repeat Step 14 for every prefix list to be created.
16. **exit**
17. **show ip bgp injected-paths**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.

	Command or Action	Purpose
Step 4	**bgp inject-map** *inject-map-name* **exist-map** *exist-map-name* [**copy-attributes**] **Example:** `Router(config-router)# bgp inject-map ORIGINATE exist-map LEARNED_PATH`	Specifies the inject map and the exist map for conditional route injection. • Use the **copy-attributes** keyword to specify that the injected route inherit the attributes of the aggregate route.
Step 5	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 6	**route-map** *map-tag* [**permit**| **deny**][*sequence-number*] **Example:** `Router(config)# route-map LEARNED_PATH permit 10`	Configures a route map and enters route map configuration mode.
Step 7	**match ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [*access-list-number*... | *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]} **Example:** `Router(config-route-map)# match ip address prefix-list SOURCE`	Specifies the aggregate route to which a more specific route will be injected. • In this example, the prefix list named SOURCE is used to redistribute the source of the route.
Step 8	**match ip route-source** {*access-list-number* | *access-list-name*} [*access-list-number*...| *access-list-name*...] **Example:** `Router(config-route-map)# match ip route-source prefix-list ROUTE_SOURCE`	Specifies the match conditions for redistributing the source of the route. • In this example, the prefix list named ROUTE_SOURCE is used to redistribute the source of the route. **Note** The route source is the neighbor address that is configured with the **neighbor remote-as** command. The tracked prefix must come from this neighbor in order for conditional route injection to occur.
Step 9	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 10	**route-map** *map-tag* [**permit**| **deny**][*sequence-number*] **Example:** `Router(config)# route-map ORIGINATE permit 10`	Configures a route map and enters route map configuration mode.
Step 11	**set ip address** {*access-list-number* [*access-list-number*... | *access-list-name*...] | *access-list-name* [access-list-number... | access-list-name] | **prefix-list** *prefix-list-name* [*prefix-list-name*...]} **Example:** `Router(config-route-map)# set ip address prefix-list ORIGINATED_ROUTES`	Specifies the routes to be injected. • In this example, the prefix list named originated_routes is used to redistribute the source of the route.
Step 12	**set community** {*community-number* [**additive**] [*well-known-community*] | **none**} **Example:** `Router(config-route-map)# set community 14616:555 additive`	Sets the BGP community attribute of the injected route.
Step 13	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 14	**ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list SOURCE permit 10.1.1.0/24`	Configures a prefix list. • In this example, the prefix list named SOURCE is configured to permit routes from network 10.1.1.0/24.
Step 15	Repeat Step 14 for every prefix list to be created.	--
Step 16	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 17	**show ip bgp injected-paths** **Example:** `Router# show ip bgp injected-paths`	(Optional) Displays information about injected paths.

Examples

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

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

• Troubleshooting Tips, page 86

Troubleshooting Tips

BGP conditional route injection is based on the injection of a more specific prefix into the BGP routing table when a less specific prefix is present. If conditional route injection is not working properly, verify the following:

- If conditional route injection is configured but does not occur, verify the existence of the aggregate prefix in the BGP routing table. The existence (or not) of the tracked prefix in the BGP routing table can be verified with the **show ip bgp**command.
- If the aggregate prefix exists but conditional route injection does not occur, verify that the aggregate prefix is being received from the correct neighbor and the prefix list identifying that neighbor is a /32 match.
- Verify the injection (or not) of the more specific prefix using the **show ip bgp injected-paths**command.
- Verify that the prefix that is being injected is not outside of the scope of the aggregate prefix.
- Ensure that the inject route map is configured with the **set ip address** command and not the **match ip address** command.

Originating BGP Routes Using Backdoor Routes

Use this task to indicate to border routers which networks are reachable using a backdoor route. A backdoor network is treated the same as a local network except that it is not advertised. For more information see the BGP Backdoor Routes, page 34.

This task assumes that the IGP--EIGRP in this example--is already configured for the BGP peers. The configuration is done at Router B in the figure above (in the "BGP Backdoor Routes" section) and the BGP peer is Router D.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **network** *ip-address* **backdoor**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 172.22.1.2 remote-as 45000`	Adds the IP address of the neighbor in the specified autonomous system to the multiprotocol BGP neighbor table of the local router. • In this example, the peer is an internal peer as the autonomous system number specified for the peer is the same number specified in Step 3.
Step 5	**network** *ip-address* **backdoor** **Example:** `Router(config-router)# network 172.21.1.0 backdoor`	Indicates a network that is reachable through a backdoor route.

	Command or Action	Purpose
Step 6	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Configuring a BGP Peer Group

This task explains how to configure a BGP peer group. Often, in a BGP speaker, many neighbors are configured with the same update policies (that is, the same outbound route maps, distribute lists, filter lists, update source, and so on). Neighbors with the same update policies can be grouped into peer groups to simplify configuration and, more importantly, to make updating more efficient. When you have many peers, this approach is highly recommended.

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

- Creating the peer group
- Assigning options to the peer group
- Making neighbors members of the peer group

You can disable a BGP peer or peer group without removing all the configuration information using the **neighbor shutdown** router configuration command.

Note By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *peer-group-name* **peer-group**
5. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
6. **neighbor** *ip-address* **peer-group** *peer-group-name*
7. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
8. **neighbor** *peer-group-name* **activate**
9. **neighbor** *ip-address* **peer-group** *peer-group-name*
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** *peer-group-name* **peer-group** **Example:** `Router(config-router)# neighbor fingroup peer-group`	Creates a BGP peer group.
Step 5	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address of the neighbor in the specified autonomous system to the multiprotocol BGP neighbor table of the local router.
Step 6	**neighbor** *ip-address* **peer-group** *peer-group-name* **Example:** `Router(config-router)# neighbor 192.168.1.1 peer-group fingroup`	Assigns the IP address of a BGP neighbor to a peer group.

	Command or Action	Purpose
Step 7	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 multicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. This is the default. • The **multicast** keyword specifies that IPv4 multicast address prefixes will be exchanged. • The **vrf**keyword and *vrf-name* argument specify that IPv4 VRF instance information will be exchanged.
Step 8	**neighbor** *peer-group-name* **activate** **Example:** `Router(config-router-af)# neighbor fingroup activate`	Enables the neighbor to exchange prefixes for the IPv4 address family with the local router. **Note** By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only unicast address prefixes. To allow BGP to exchange other address prefix types, such as multicast that is configured in this example, neighbors must also be activated using the **neighbor activate** command.
Step 9	**neighbor** *ip-address* **peer-group** *peer-group-name* **Example:** `Router(config-router-af)# neighbor 192.168.1.1 peer-group fingroup`	Assigns the IP address of a BGP neighbor to a peer group.
Step 10	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

Configuring Peer Session Templates

The following tasks create and configure a peer session template:

- Configuring a Basic Peer Session Template, page 90
- Configuring Peer Session Template Inheritance with the inherit peer-session Command, page 93
- Configuring Peer Session Template Inheritance with the neighbor inherit peer-session Command, page 95

Configuring a Basic Peer Session Template

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

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

Note The following restrictions apply to the peer session templates:

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

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-session** *session-template-name*
5. **remote-as** *autonomous-system-number*
6. **timers** *keepalive-interval hold-time*
7. **end**
8. **show ip bgp template peer-session** [*session-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-session** *session-template-name* **Example:** `Router(config-router)# template peer-session INTERNAL-BGP`	Enters session-template configuration mode and creates a peer session template.
Step 5	**remote-as** *autonomous-system-number* **Example:** `Router(config-router-stmp)# remote-as 202`	(Optional) Configures peering with a remote neighbor in the specified autonomous system. **Note** Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.
Step 6	**timers** *keepalive-interval hold-time* **Example:** `Router(config-router-stmp)# timers 30 300`	(Optional) Configures BGP keepalive and hold timers. • The hold time must be at least twice the keepalive time. **Note** Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.
Step 7	**end** **Example:** `Router(config-router)# end`	Exits session-template configuration mode and returns to privileged EXEC mode.
Step 8	**show ip bgp template peer-session** [*session-template-name*] **Example:** `Router# show ip bgp template peer-session`	Displays locally configured peer session templates. • The output can be filtered to display a single peer policy template with the *session-template-name* argument. This command also supports all standard output modifiers.

- What to Do Next, page 92

What to Do Next

After the peer session template is created, the configuration of the peer session template can be inherited or applied by another peer session template with the **inherit peer-session** or **neighbor inherit peer-session** command.

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

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

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-session** *session-template-name*
5. **description** *text-string*
6. **update-source** *interface-type interface-number*
7. **inherit peer-session** *session-template-name*
8. **end**
9. **show ip bgp template peer-session** [*session-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**template peer-session** *session-template-name* **Example:** `Router(config-router)# template peer-session CORE1`	Enter session-template configuration mode and creates a peer session template.
Step 5	**description** *text-string* **Example:** `Router(config-router-stmp)# description CORE-123`	(Optional) Configures a description. • The text string can be up to 80 characters. **Note** Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.
Step 6	**update-source** *interface-type interface-number* **Example:** `Router(config-router-stmp)# update-source loopback 1`	(Optional) Configures a router to select a specific source or interface to receive routing table updates. • The example uses a loopback interface. The advantage to this configuration is that the loopback interface is not as susceptible to the effects of a flapping interface. **Note** Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.
Step 7	**inherit peer-session** *session-template-name* **Example:** `Router(config-router-stmp)# inherit peer-session INTERNAL-BGP`	Configures this peer session template to inherit the configuration of another peer session template. • The example configures this peer session template to inherit the configuration from INTERNAL-BGP. This template can be applied to a neighbor, and the configuration INTERNAL-BGP will be applied indirectly. No additional peer session templates can be directly applied. However, the directly inherited template can contain up to seven indirectly inherited peer session templates.
Step 8	**end** **Example:** `Router(config-router)# end`	Exits session-template configuration mode and enters privileged EXEC mode.
Step 9	**show ip bgp template peer-session** [*session-template-name*] **Example:** `Router# show ip bgp template peer-session`	Displays locally configured peer session templates. • The output can be filtered to display a single peer policy template with the optional *session-template-name* argument. This command also supports all standard output modifiers.

• What to Do Next, page 95

What to Do Next

After the peer session template is created, the configuration of the peer session template can be inherited or applied by another peer session template with the **inherit peer-session** or **neighbor inherit peer-session** command.

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

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **neighbor** *ip-address* **inherit peer-session** *session-template-name*
6. **end**
7. **show ip bgp template peer-session** [*session-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 172.16.0.1 remote-as 202`	Configures a peering session with the specified neighbor. • The explicit **remote-as** statement is required for the neighbor inherit statement in Step 5 to work. If a peering is not configured, the specified neighbor in Step 5 will not accept the session template.
Step 5	**neighbor** *ip-address* **inherit peer-session** *session-template-name* **Example:** `Router(config-router)# neighbor 172.16.0.1 inherit peer-session CORE1`	Sends a peer session template to a neighbor so that the neighbor can inherit the configuration. • The example configures a router to send the peer session template named CORE1 to the 172.16.0.1 neighbor to inherit. This template can be applied to a neighbor, and if another peer session template is indirectly inherited in CORE1, the indirectly inherited configuration will also be applied. No additional peer session templates can be directly applied. However, the directly inherited template can also inherit up to seven additional indirectly inherited peer session templates.
Step 6	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.
Step 7	**show ip bgp template peer-session** [*session-template-name*] **Example:** `Router# show ip bgp template peer-session`	Displays locally configured peer session templates. • The output can be filtered to display a single peer policy template with the optional *session-template-name* argument. This command also supports all standard output modifiers.

• What to Do Next, page 96

What to Do Next

To create a peer policy template, go to the Configuring Peer Policy Templates, page 96.

Configuring Peer Policy Templates

The following tasks create and configure a peer policy template:

• Configuring Basic Peer Policy Templates, page 97
• Configuring Peer Policy Template Inheritance with the inherit peer-policy Command, page 99
• Configuring Peer Policy Template Inheritance with the neighbor inherit peer-policy Command, page 101

Configuring Basic Peer Policy Templates

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

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

Note The following restrictions apply to the peer policy templates:

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

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-policy** *policy-template-name*
5. **maximum-prefix** *prefix-limit* [*threshold*] [**restart** *restart-interval* | **warning-only**]
6. **weight** *weight-value*
7. **prefix-list** *prefix-list-name* {**in** | **out**}
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-policy** *policy-template-name* **Example:** `Router(config-router)# template peer-policy GLOBAL`	Enters policy-template configuration mode and creates a peer policy template.
Step 5	**maximum-prefix** *prefix-limit* [*threshold*] [**restart** *restart-interval* | **warning-only**] **Example:** `Router(config-router-ptmp)# maximum-prefix 10000`	(Optional) Configures the maximum number of prefixes that a neighbor will accept from this peer. **Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the Peer Policy Templates, page 38.
Step 6	**weight** *weight-value* **Example:** `Router(config-router-ptmp)# weight 300`	(Optional) Sets the default weight for routes that are sent from this neighbor. **Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the Peer Policy Templates, page 38.
Step 7	**prefix-list** *prefix-list-name* {**in** | **out**} **Example:** `Router(config-router-ptmp)# prefix-list NO-MARKETING in`	(Optional) Filters prefixes that are received by the router or sent from the router. • The prefix list in the example filters inbound internal addresses. **Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the Peer Policy Templates, page 38.
Step 8	**end** **Example:** `Router(config-router-ptmp)# end`	Exits policy-template configuration mode and returns to privileged EXEC mode.

- What to Do Next, page 98

What to Do Next

After the peer policy template is created, the configuration of the peer policy template can be inherited or applied by another peer policy template. For more details about peer policy inheritance see the

"Configuring Peer Policy Template Inheritance with the inherit peer-policy Command" section or the "Configuring Peer Policy Template Inheritance with the neighbor inherit peer-policy Command" section.

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

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

When BGP neighbors use inherited peer templates, it can be difficult to determine which policies are associated with a specific template. In Cisco IOS Release 12.0(25)S, 12.4(11)T, 12.2(33)SRB, 12.2(33)SB, and later releases, the **detail** keyword was added to the **show ip bgp template peer-policy** command to display the detailed configuration of local and inherited policies associated with a specific template.

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-policy** *policy-template-name*
5. **route-map** *map-name* {**in**| **out**}
6. **inherit peer-policy** *policy-template-name sequence-number*
7. **end**
8. **show ip bgp template peer-policy** [*policy-template-name*[**detail**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-policy** *policy-template-name* **Example:** `Router(config-router)# template peer-policy NETWORK1`	Enter policy-template configuration mode and creates a peer policy template.
Step 5	**route-map** *map-name* {**in**| **out**} **Example:** `Router(config-router-ptmp)# route-map ROUTE in`	(Optional) Applies the specified route map to inbound or outbound routes. **Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the Peer Policy Templates, page 38.
Step 6	**inherit peer-policy** *policy-template-name sequence-number* **Example:** `Router(config-router-ptmp)# inherit peer-policy GLOBAL 10`	Configures the peer policy template to inherit the configuration of another peer policy template. • The *sequence-number* argument sets the order in which the peer policy template is evaluated. Like a route map sequence number, the lowest sequence number is evaluated first. • The example configures this peer policy template to inherit the configuration from GLOBAL. If the template created in these steps is applied to a neighbor, the configuration GLOBAL will also be inherited and applied indirectly. Up to six additional peer policy templates can be indirectly inherited from GLOBAL for a total of eight directly applied and indirectly inherited peer policy templates. • This template in the example will be evaluated first if no other templates are configured with a lower sequence number.
Step 7	**end** **Example:** `Router(config-router-ptmp)# end`	Exits policy-template configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 8	**show ip bgp template peer-policy** [*policy-template-name*[**detail**]] **Example:** `Router# show ip bgp template peer-policy NETWORK1 detail`	Displays locally configured peer policy templates. • The output can be filtered to display a single peer policy template with the *policy-template-name* argument. This command also supports all standard output modifiers. • Use the **detail** keyword to display detailed policy information. **Note** The **detail** keyword is supported only in Cisco IOS Release 12.0(25)S, 12.4(11)T, 12.2(33)SRB, 12.2(33)SB, and later releases.

Examples

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

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

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

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

When BGP neighbors use multiple levels of peer templates, it can be difficult to determine which policies are applied to the neighbor. In Cisco IOS Release 12.0(25)S, 12.4(11)T, 12.2(33)SRB, 12.2(33)SB, and later releases, the **policy** and **detail** keywords were added to the **show ip bgp neighbors** command to display the inherited policies and policies configured directly on the specified neighbor.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**multicast** | **unicast** | **vrf** *vrf-name*]
6. **neighbor** *ip-address* **inherit peer-policy** *policy-template-name*
7. **end**
8. **show ip bgp neighbors** [*ip-address*[**policy** [**detail**]]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Configures a peering session with the specified neighbor. • The explicit **remote-as** statement is required for the **neighbor inherit** statement in Step 6 to work. If a peering is not configured, the specified neighbor in Step 6 will not accept the session template.
Step 5	**address-family ipv4** [**multicast** | **unicast** | **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure a neighbor to accept address family-specific command configurations.

	Command or Action	Purpose
Step 6	**neighbor** *ip-address* **inherit peer-policy** *policy-template-name* **Example:** `Router(config-router-af)# neighbor 192.168.1.2 inherit peer-policy GLOBAL`	Sends a peer policy template to a neighbor so that the neighbor can inherit the configuration. • The example configures a router to send the peer policy template named GLOBAL to the 192.168.1.2 neighbor to inherit. This template can be applied to a neighbor, and if another peer policy template is indirectly inherited from GLOBAL, the indirectly inherited configuration will also be applied. Up to seven additional peer policy templates can be indirectly inherited from GLOBAL.
Step 7	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 8	**show ip bgp neighbors** [*ip-address*[**policy** [**detail**]]] **Example:** `Router# show ip bgp neighbors 192.168.1.2 policy`	Displays locally configured peer policy templates. • The output can be filtered to display a single peer policy template with the *policy-template-name* argument. This command also supports all standard output modifiers. • Use the **policy** keyword to display the policies applied to this neighbor per address family. • Use the **detail** keyword to display detailed policy information. • The **policy** and **detail** keywords are supported only in Cisco IOS Release 12.0(25)S, 12.4(11)T, 12.2(33)SRB, 12.2(33)SB, and later releases. **Note** Only the syntax required for this task is shown. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

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

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

Monitoring and Maintaining BGP Dynamic Update Groups

Use this task to clear and display information about the processing of dynamic BGP update groups. The performance of BGP update message generation is improved with the use of BGP update groups. With the configuration of the BGP peer templates and the support of the dynamic BGP update groups, the network operator no longer needs to configure peer groups in BGP and can benefit from improved configuration flexibility and system performance. For more information about using BGP peer templates, see the Configuring Peer Session Templates, page 90 and the Configuring Peer Policy Templates, page 96.

SUMMARY STEPS

1. **enable**
2. **clear ip bgp update-group** [*index-group*| *ip-address*]
3. **show ip bgp replication** [*index-group*| *ip-address*]
4. **show ip bgp update-group** [*index-group* | *ip-address*] [**summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear ip bgp update-group** [*index-group*| *ip-address*] **Example:** `Router#` **clear ip bgp update-group 192.168.2.2**	Clears BGP update group membership and recalculate BGP update groups: • In the example provided, the membership of neighbor 192.168.2.2 is cleared from an update group.
Step 3	**show ip bgp replication** [*index-group*| *ip-address*] **Example:** `Router#` **show ip bgp replication**	Displays update replication statistics for BGP update groups.
Step 4	**show ip bgp update-group** [*index-group* | *ip-address*] [**summary**] **Example:** `Router#` **show ip bgp update-group**	Displays information about BGP update groups.

• Troubleshooting Tips, page 104

Troubleshooting Tips

Use the **debug ip bgp groups** command to display information about the processing of BGP update groups. Information can be displayed for all update groups, an individual update group, or a specific BGP neighbor. The output of this command can be very verbose. This command should not be deployed in a production network unless your are troubleshooting a problem.

Configuration Examples for a Basic BGP Network

Example Configuring a BGP Process and Customizing Peers

The following example shows the configuration for Router B in the figure above (in the "Customizing a BGP Peer" section) with a BGP process configured with two neighbor peers (at Router A and at Router E) in separate autonomous systems. IPv4 unicast routes are exchanged with both peers and IPv4 multicast routes are exchanged with the BGP peer at Router E.

Router B

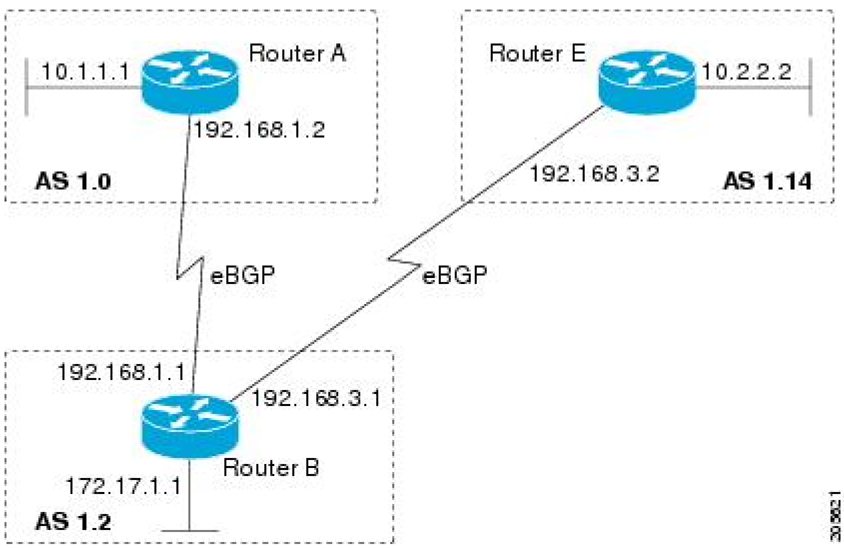
```
router bgp 45000
 bgp router-id 172.17.1.99
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.3.2 remote-as 50000
 neighbor 192.168.3.2 description finance
 !
 address-family ipv4
  neighbor 192.168.1.2 activate
  neighbor 192.168.3.2 activate
  no auto-summary
  no synchronization
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
 !
 address-family ipv4 multicast
  neighbor 192.168.3.2 activate
  neighbor 192.168.3.2 advertisement-interval 25
  no auto-summary
  no synchronization
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
```

Examples Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers

Asplain Default Format in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)SXI1, and Later Releases

The following example is available in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases and shows the configuration for Router A, Router B, and Router E in the figure below with a BGP process configured between three neighbor peers (at Router A, at Router B, and at Router E) in separate 4-byte autonomous systems configured using asplain notation. IPv4 unicast routes are exchanged with all peers.

Figure 10 BGP Peers Using 4-Byte Autonomous System Numbers in Asplain Format



Router A

```
router bgp 65536
 bgp router-id 10.1.1.99
 no bgp default ipv4-unicast
 bgp fast-external-fallover
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.1.1 remote-as 65538
 !
 address-family ipv4
  neighbor 192.168.1.1 activate
  no auto-summary
  no synchronization
  network 10.1.1.0 mask 255.255.255.0
  exit-address-family
```

Router B

```
router bgp 65538
 bgp router-id 172.17.1.99
 no bgp default ipv4-unicast
```

```
 bgp fast-external-fallover
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.1.2 remote-as 65536
 neighbor 192.168.3.2 remote-as 65550
 neighbor 192.168.3.2 description finance
 !
 address-family ipv4
  neighbor 192.168.1.2 activate
  neighbor 192.168.3.2 activate
  no auto-summary
  no synchronization
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
```

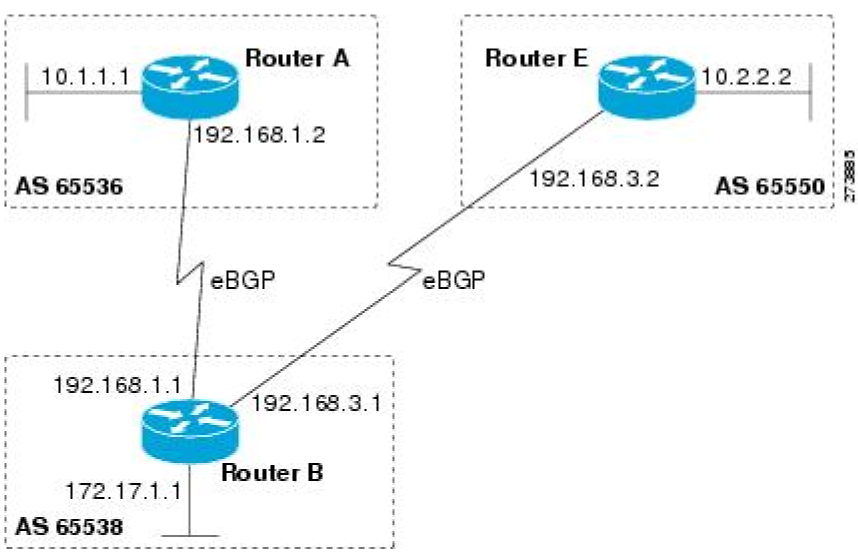
Router E

```
router bgp 65550
 bgp router-id 10.2.2.99
 no bgp default ipv4-unicast
 bgp fast-external-fallover
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.3.1 remote-as 65538
 !
 address-family ipv4
  neighbor 192.168.3.1 activate
  no auto-summary
  no synchronization
  network 10.2.2.0 mask 255.255.255.0
  exit-address-family
```

Asdot Default Format in Cisco IOS Release 12.0(32)S12, and 12.4(24)T

The following example is available in Cisco IOS Release 12.0(32)S12, and 12.4(24)T and shows how to create the configuration for Router A, Router B, and Router E in the figure below with a BGP process configured between three neighbor peers (at Router A, at Router B, and at Router E) in separate 4-byte autonomous systems configured using the default asdot format. IPv4 unicast routes are exchanged with all peers.

Figure 11 BGP Peers Using 4-Byte Autonomous System Numbers in Asdot Format



Router A

```
router bgp 1.0
 bgp router-id 10.1.1.99
 no bgp default ipv4-unicast
 bgp fast-external-fallover
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.1.1 remote-as 1.2
 !
 address-family ipv4
  neighbor 192.168.1.1 activate
  no auto-summary
  no synchronization
  network 10.1.1.0 mask 255.255.255.0
  exit-address-family
```

Router B

```
router bgp 1.2
 bgp router-id 172.17.1.99
 no bgp default ipv4-unicast
 bgp fast-external-fallover
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.1.2 remote-as 1.0
 neighbor 192.168.3.2 remote-as 1.14
 neighbor 192.168.3.2 description finance
 !
 address-family ipv4
  neighbor 192.168.1.2 activate
  neighbor 192.168.3.2 activate
  no auto-summary
  no synchronization
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
```

Router E

```
router bgp 1.14
 bgp router-id 10.2.2.99
 no bgp default ipv4-unicast
 bgp fast-external-fallover
 bgp log-neighbor-changes
 timers bgp 70 120
 neighbor 192.168.3.1 remote-as 1.2
 !
 address-family ipv4
  neighbor 192.168.3.1 activate
  no auto-summary
  no synchronization
  network 10.2.2.0 mask 255.255.255.0
  exit-address-family
```

Examples Configuring a VRF and Setting an Extended Community Using a BGP 4-Byte Autonomous System Number

Asplain Default Format in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)SXI1, and Later Releases

The following example is available in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases and shows how to create a VRF with a route-target that uses a 4-byte autonomous system number, 65537, and how to set the route target to extended community value 65537:100 for routes that are permitted by the route map.

```
ip vrf vpn_red
 rd 64500:100
 route-target both 65537:100
 exit
route-map red_map permit 10
 set extcommunity rt 65537:100
 end
```

After the configuration is completed, use the **show route-map** command to verify that the extended community is set to the route target that contains the 4-byte autonomous system number of 65537.

```
RouterB# show route-map red_map
route-map red_map, permit, sequence 10
  Match clauses:
  Set clauses:
    extended community RT:65537:100
  Policy routing matches: 0 packets, 0 bytes
```

Asdot Default Format in Cisco IOS Release 12.0(32)S12, and 12.4(24)T

The following example is available in Cisco IOS Release 12.0(32)S12, and 12.4(24)T and shows how to create a VRF with a route-target that uses a 4-byte autonomous system number, 1.1, and how to set the route target to extended community value 1.1:100 for routes that are permitted by the route map.

Note In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SXI1, and later releases, this example works if you have configured asdot as the default display format using the **bgp asnotation dot** command.

```
ip vrf vpn_red
 rd 64500:100
 route-target both 1.1:100
 exit
route-map red_map permit 10
 set extcommunity rt 1.1:100
 end
```

After the configuration is completed, use the **show route-map** command to verify that the extended community is set to the route target that contains the 4-byte autonomous system number of 1.1.

```
RouterB# show route-map red_map
route-map red_map, permit, sequence 10
  Match clauses:
  Set clauses:
    extended community RT:1.1:100
  Policy routing matches: 0 packets, 0 bytes
```

Example NLRI to AFI Configuration

The following example upgrades an existing router configuration file in the NLRI format to the AFI format and set the router CLI to use only commands in the AFI format:

```
router bgp 60000
 bgp upgrade-cli
```

The **show running-config** command can be used in privileged EXEC mode to verify that an existing router configuration file has been upgraded from the NLRI format to the AFI format. The following sections provide sample output from a router configuration file in the NLRI format, and the same router configuration file after it has been upgraded to the AFI format with the **bgp upgrade-cli** command in router configuration mode.

Note After a router has been upgraded from the AFI format to the NLRI format with the **bgp upgrade-cli**command, NLRI commands will no longer be accessible or configurable.

Router Configuration File in NLRI Format Before Upgrading

The following sample output is from the **show running-config** command in privileged EXEC mode. The sample output shows a router configuration file, in the NLRI format, prior to upgrading to the AFI format with the **bgp upgrade-cli** command. The sample output is filtered to show only the affected portion of the router configuration.

```
Router# show running-config | begin bgp
router bgp 101
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.1.1.1 remote-as 505 nlri unicast multicast
 no auto-summary
!
ip default-gateway 10.4.9.1
ip classless
!
!
route-map REDISTRIBUTE-MULTICAST permit 10
 match ip address prefix-list MULTICAST-PREFIXES
 set nlri multicast
!
route-map MULTICAST-PREFIXES permit 10
!
route-map REDISTRIBUTE-UNICAST permit 20
 match ip address prefix-list UNICAST-PREFIXES
 set nlri unicast
!
!
!
line con 0
line aux 0
line vty 0 4
 password PASSWORD
 login
!
end
```

Router Configuration File in AFI Format After Upgrading

The following sample output shows the router configuration file after it has been upgraded to the AFI format. The sample output is filtered to show only the affected portion of the router configuration file.

```
Router# show running-config | begin bgp
router bgp 101
 bgp log-neighbor-changes
 neighbor 10.1.1.1 remote-as 505
 no auto-summary
 !
 address-family ipv4 multicast
  neighbor 10.1.1.1 activate
  no auto-summary
  no synchronization
  exit-address-family
 !
 address-family ipv4
  neighbor 10.1.1.1 activate
```

```
  no auto-summary
  no synchronization
  exit-address-family
!
ip default-gateway 10.4.9.1
ip classless
!
!
route-map REDISTRIBUTE-MULTICAST_mcast permit 10
 match ip address prefix-list MULTICAST-PREFIXES
!
route-map REDISTRIBUTE-MULTICAST permit 10
 match ip address prefix-list MULTICAST-PREFIXES
!
route-map MULTICAST-PREFIXES permit 10
!
route-map REDISTRIBUTE-UNICAST permit 20
 match ip address prefix-list UNICAST-PREFIXES
!
!
!
line con 0
line aux 0
line vty 0 4
 password PASSWORD
 login
!
end
```

Examples Removing BGP Configuration Commands Using a Redistribution Example

The following examples show both the CLI configuration to enable the redistribution of BGP routes into EIGRP using a route map, and the CLI configuration to remove the redistribution and route map. Some BGP configuration commands can affect other CLI commands and this example demonstrates how the removal of one command affects another command.

In the first configuration example, a route map is configured to match and set autonomous system numbers. BGP neighbors in three different autonomous systems are configured and activated. An EIGRP routing process is started, and the redistribution of BGP routes into EIGRP using the route map is configured.

CLI to Enable BGP Route Redistribution Into EIGRP

```
route-map bgp-to-eigrp permit 10
 match tag 50000
 set tag 65000
 exit
router bgp 45000
 bgp log-neighbor-changes
 address-family ipv4
  neighbor 172.16.1.2 remote-as 45000
  neighbor 172.21.1.2 remote-as 45000
  neighbor 192.168.1.2 remote-as 40000
  neighbor 192.168.3.2 remote-as 50000
  neighbor 172.16.1.2 activate
  neighbor 172.21.1.2 activate
  neighbor 192.168.1.2 activate
  neighbor 192.168.3.2 activate
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
 exit
router eigrp 100
 redistribute bgp 45000 metric 10000 100 255 1 1500 route-map bgp-to-eigrp
 no auto-summary
 exit
```

In the second configuration example, both the **route-map** command and the **redistribute** command are disabled. If only the route-map command is removed, it does not automatically disable the redistribution.

The redistribution will now occur without any matching or filtering. To remove the redistribution configuration, the **redistribute** command must also be disabled.

CLI to Remove BGP Route Redistribution Into EIGRP

```
configure terminal
 no route-map bgp-to-eigrp
 router eigrp 100
  no redistribute bgp 45000
  end
```

Examples BGP Soft Reset

The following examples show two ways to reset the connection for BGP peer 192.168.1.1.

Dynamic Inbound Soft Reset Example

The following example shows the **clear ip bgp 192.168.1.1 soft in** EXEC command used to initiate a dynamic soft reconfiguration in the BGP peer 192.168.1.1. This command requires that the peer support the route refresh capability.

```
clear ip bgp 192.168.1.1 soft in
```

Inbound Soft Reset Using Stored Information Example

The following example shows how to enable inbound soft reconfiguration for the neighbor 192.168.1.1. All the updates received from this neighbor will be stored unmodified, regardless of the inbound policy. When inbound soft reconfiguration is performed later, the stored information will be used to generate a new set of inbound updates.

```
router bgp 100
 neighbor 192.168.1.1 remote-as 200
 neighbor 192.168.1.1 soft-reconfiguration inbound
```

The following example clears the session with the neighbor 192.168.1.1:

```
clear ip bgp 192.168.1.1 soft in
```

Example Resetting BGP Peers Using 4-Byte Autonomous System Numbers

The following examples show how to clear BGP peers belonging to an autonomous system that uses 4-byte autonomous system numbers. This example requires Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, or a later release to be running on the router. The initial state of the BGP routing table is shown using the **show ip bgp** command, and peers in 4-byte autonomous systems 65536 and 65550 are displayed.

```
RouterB# show ip bgp

BGP table version is 4, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 65536  i
*> 10.2.2.0/24      192.168.3.2              0             0 65550  i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
```

The **clear ip bgp 65550** command is entered to remove all BGP peers in the 4-byte autonomous system 65550. The ADJCHANGE message shows that the BGP peer at 192.168.3.2 is being reset.

```
RouterB# clear ip bgp 65550
RouterB#
*Nov 30 23:25:27.043: %BGP-5-ADJCHANGE: neighbor 192.168.3.2 Down User reset
```

The **show ip bgp** command is entered again, and only the peer in 4-byte autonomous systems 65536 is now displayed.

```
RouterB# show ip bgp
BGP table version is 5, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 65536  i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
```

Almost immediately the next ADJCHANGE message shows that the BGP peer at 192.168.3.2 (in the 4-byte autonomous system 65550) is now back up.

```
RouterB#
*Nov 30 23:25:55.995: %BGP-5-ADJCHANGE: neighbor 192.168.3.2 Up
```

Example Resetting and Displaying Basic BGP Information

The following example shows how to reset and display basic BGP information.

The **clear ip bgp *** command clears and resets all the BGP neighbor sessions. In Cisco IOS Release 12.2(25)S and later releases, the syntax is **clear ip bgp all**. Specific neighbors or all peers in an autonomous system can be cleared by using the *neighbor-address* and *autonomous-system-number* arguments. If no argument is specified, this command will clear and reset all BGP neighbor sessions.

Note The **clear ip bgp *** command also clears all the internal BGP structures which makes it useful as a troubleshooting tool.

```
Router# clear ip bgp *
```

The **show ip bgp** command is used to display all the entries in the BGP routing table. The following example displays BGP routing table information for the 10.1.1.0 network:

```
Router# show ip bgp 10.1.1.0 255.255.255.0
BGP routing table entry for 10.1.1.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to update-groups:
     1
  40000
    192.168.1.2 from 192.168.1.2 (10.1.1.99)
      Origin IGP, metric 0, localpref 100, valid, external, best
```

The **show ip bgp neighbors**command is used to display information about the TCP and BGP connections to neighbors. The following example displays the routes that were advertised from Router B in the figure above (in the "Configuring a BGP Peer for the IPv4 VRF Address Family" section) to its BGP neighbor 192.168.3.2 on Router E:

```
Router# show ip bgp neighbors 192.168.3.2 advertised-routes
BGP table version is 3, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
```

```
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 40000 i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
Total number of prefixes 2
```

The **show ip bgp paths**command is used to display all the BGP paths in the database. The following example displays BGP path information for Router B in the figure above (in the "Customizing a BGP Peer" section):

```
Router# show ip bgp paths
Address    Hash Refcount Metric Path
0x2FB5DB0     0        5      0 i
0x2FB5C90     1        4      0 i
0x2FB5C00  1361        2      0 50000 i
0x2FB5D20  2625        2      0 40000 i
```

The **show ip bgp summary**command is used to display the status of all BGP connections. The following example displays BGP routing table information for Router B in the figure above (in the "Customizing a BGP Peer" section:

```
Router# show ip bgp summary
BGP router identifier 172.17.1.99, local AS number 45000
BGP table version is 3, main routing table version 3
2 network entries using 234 bytes of memory
2 path entries using 104 bytes of memory
4/2 BGP path/bestpath attribute entries using 496 bytes of memory
2 BGP AS-PATH entries using 48 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 882 total bytes of memory
BGP activity 14/10 prefixes, 16/12 paths, scan interval 60 secs
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.1.2     4 40000     667     672        3    0    0 00:03:49        1
192.168.3.2     4 50000     468     467        0    0    0 00:03:49 (NoNeg)
```

Examples Aggregating Prefixes Using BGP

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

In the following example, the **redistribute static**router configuration command is used to redistribute aggregate route 10.0.0.0:

```
ip route 10.0.0.0 255.0.0.0 null 0
!
router bgp 100
 redistribute static
```

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

```
router bgp 100
 aggregate-address 10.0.0.0 255.0.0.0
```

The following example shows how to create an aggregate entry using the same rules as in the previous example, but the path advertised for this route will be an AS-SET consisting of all elements contained in all paths that are being summarized:

```
router bgp 100
 aggregate-address 10.0.0.0 255.0.0.0 as-set
```

The following example shows how to create the aggregate route for 10.0.0.0 and also suppress advertisements of more specific routes to all neighbors:

```
router bgp 100
 aggregate-address 10.0.0.0 255.0.0.0 summary-only
```

The following example, starting in global configuration mode, configures BGP to not advertise inactive routes:

```
Router(config)# router bgp 50000
Router(config-router)# address-family ipv4 unicast
Router(config-router-af)# bgp suppress-inactive

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

The following example configures a maximum route limit in the VRF named red and configures BGP to not advertise inactive routes through the VRF named RED:

```
Router(config)# ip vrf RED

Router(config-vrf)# rd 50000:10
Router(config-vrf)# maximum routes 1000 10

Router(config-vrf)# exit
Router(config)# router bgp 50000
Router(config-router)# address-family ipv4 vrf RED
Router(config-router-af)# bgp suppress-inactive

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

Example Configuring a BGP Peer Group

The following example shows how to use an address family to configure a peer group so that all members of the peer group are both unicast- and multicast-capable:

```
router bgp 45000
neighbor 192.168.1.2 remote-as 40000
neighbor 192.168.3.2 remote-as 50000
address-family ipv4 unicast
 neighbor mygroup peer-group
 neighbor 192.168.1.2 peer-group mygroup
 neighbor 192.168.3.2 peer-group mygroup
router bgp 45000
neighbor 192.168.1.2 remote-as 40000
neighbor 192.168.3.2 remote-as 50000
address-family ipv4 multicast
 neighbor mygroup peer-group
 neighbor 192.168.1.2 peer-group mygroup
 neighbor 192.168.3.2 peer-group mygroup
 neighbor 192.168.1.2 activate
 neighbor 192.168.3.2 activate
```

Example Configuring Peer Session Templates

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

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

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

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

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

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

Example Configuring Peer Policy Templates

The following example creates a peer policy template named GLOBAL in policy-template configuration mode:

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

The following example creates a peer policy template named PRIMARY-IN in policy-template configuration mode:

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

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

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

The following example configures the 192.168.2.2 neighbor in address family mode to inherit the peer policy template name CUSTOMER-A. The 192.168.2.2 neighbor will also indirectly inherit the peer policy templates named PRIMARY-IN and GLOBAL.

```
router bgp 45000
 neighbor 192.168.2.2 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.168.2.2 inherit peer-policy CUSTOMER-A
  end
```

Examples Monitoring and Maintaining BGP Dynamic Update Peer-Groups

No configuration is required to enable the BGP dynamic update of peer groups and the algorithm runs automatically. The following examples show how BGP update group information can be cleared or displayed.

clear ip bgp update-group Example

The following example clears the membership of neighbor 10.0.0.1 from an update group:

```
Router#
 clear ip bgp update-group 10.0.0.1
```

debug ip bgp groups Example

The following example output from the **debug ip bgp groups** command shows the recalculation of update groups after the **clear ip bgp groups** command was issued:

```
Router# debug ip bgp groups
5w4d: %BGP-5-ADJCHANGE: neighbor 10.4.9.5 Down User reset
5w4d: BGP-DYN(0): Comparing neighbor 10.4.9.5 flags 0x0 cap 0x0 and updgrp 2 fl0
5w4d: BGP-DYN(0): Update-group 2 flags 0x0 cap 0x0 policies same as 10.4.9.5 fl0
5w4d: %BGP-5-ADJCHANGE: neighbor 10.4.9.8 Down User reset
5w4d: BGP-DYN(0): Comparing neighbor 10.4.9.8 flags 0x0 cap 0x0 and updgrp 2 fl0
5w4d: BGP-DYN(0): Update-group 2 flags 0x0 cap 0x0 policies same as 10.4.9.8 fl0
5w4d: %BGP-5-ADJCHANGE: neighbor 10.4.9.21 Down User reset
5w4d: BGP-DYN(0): Comparing neighbor 10.4.9.21 flags 0x0 cap 0x0 and updgrp 1 f0
5w4d: BGP-DYN(0): Update-group 1 flags 0x0 cap 0x0 policies same as 10.4.9.21 f0
5w4d: %BGP-5-ADJCHANGE: neighbor 10.4.9.5 Up
5w4d: %BGP-5-ADJCHANGE: neighbor 10.4.9.21 Up
5w4d: %BGP-5-ADJCHANGE: neighbor 10.4.9.8 Up
```

show ip bgp replication Example

The following sample output from the **show ip bgp replication**command shows update group replication information for all for neighbors:

```
Router# show ip bgp replication
BGP Total Messages Formatted/Enqueued : 0/0
     Index     Type  Members          Leader   MsgFmt  MsgRepl  Csize  Qsize
         1 internal        1       10.4.9.21        0        0      0      0
         2 internal        2        10.4.9.5        0        0      0      0
```

show ip bgp update-group Example

The following sample output from the **show ip bgp update-group** command shows update group information for all neighbors:

```
Router# show ip bgp update-group
BGP version 4 update-group 1, internal, Address Family: IPv4 Unicast
  BGP Update version : 0, messages 0/0
```

```
  Route map for outgoing advertisements is COST1
  Update messages formatted 0, replicated 0
  Number of NLRIs in the update sent: max 0, min 0
  Minimum time between advertisement runs is 5 seconds
  Has 1 member:
  10.4.9.21
BGP version 4 update-group 2, internal, Address Family: IPv4 Unicast
  BGP Update version : 0, messages 0/0
  Update messages formatted 0, replicated 0
  Number of NLRIs in the update sent: max 0, min 0
  Minimum time between advertisement runs is 5 seconds
  Has 2 members:
  10.4.9.5 10.4.9.8
```

Where to Go Next

- If you want to connect to an external service provider, see the "Connecting to a Service Provider Using External BGP" module.
- To configure BGP neighbor session options, proceed to the "Configuring BGP Neighbor Session Options" module.
- If you want to configure some iBGP features, see the "Configuring Internal BGP Features" module.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
BGP commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
IPv6 commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS IPv6 Command Reference*
Overview of Cisco BGP conceptual information with links to all the individual BGP modules	"Cisco BGP Overview" module
Multiprotocol Label Switching (MPLS) and BGP configuration example using the IPv4 VRF address family	"Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels" module
Basic MPLS VPN and BGP configuration example	"Configuring MPLS Layer 3 VPNs" module

Standards

Standard	Title
MDT SAFI	MDT SAFI

MIBs

MIB	MIBs Link
CISCO-BGP4-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1772	*Application of the Border Gateway Protocol in the Internet*
RFC 1773	*Experience with the BGP Protocol*
RFC 1774	*BGP-4 Protocol Analysis*
RFC 1930	*Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)*
RFC 2519	*A Framework for Inter-Domain Route Aggregation*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 2918	*Route Refresh Capability for BGP-4*
RFC 3392	*Capabilities Advertisement with BGP-4*
RFC 4271	A Border Gateway Protocol 4 (BGP-4)
RFC 4893	BGP Support for Four-octet AS Number Space
RFC 5396	Textual Representation of Autonomous system (AS) Numbers
RFC 5398	Autonomous System (AS) Number Reservation for Documentation Use

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring a Basic BGP Network

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 11 Feature Information for Configuring a Basic BGP Network

Feature Name	Releases	Feature Configuration Information
BGP Version 4	Cisco IOS XE 3.1.0SG	BGP is an interdomain routing protocol designed to provide loop-free routing between separate routing domains that contain independent routing policies (autonomous systems). The Cisco IOS software implementation of BGP version 4 includes multiprotocol extensions to allow BGP to carry routing information for IP multicast routes and multiple Layer 3 protocol address families including IP Version 4 (IPv4), IP Version 6 (IPv6), Virtual Private Networks version 4 (VPNv4), and Connectionless Network Services (CLNS).

Feature Name	Releases	Feature Configuration Information
BGP Conditional Route Injection	12.0(22)S 12.2(4)T 12.2(14)S 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Conditional Route Injection feature allows you to inject more specific prefixes into a BGP routing table over less specific prefixes that were selected through normal route aggregation. These more specific prefixes can be used to provide a finer granularity of traffic engineering or administrative control than is possible with aggregated routes.
BGP Configuration Using Peer Templates	12.0(24)S 12.2(18)S 12.2(27)SBC 12.3(4)T 15.0(1)S	The BGP Configuration Using Peer Templates feature introduces a new mechanism that groups distinct neighbor configurations for BGP neighbors that share policies. This type of policy configuration has been traditionally configured with BGP peer groups. However, peer groups have certain limitations because peer group configuration is bound to update grouping and specific session characteristics. Configuration templates provide an alternative to peer group configuration and overcome some of the limitations of peer groups.

Feature Name	Releases	Feature Configuration Information
BGP Dynamic Update Peer Groups	12.0(24)S 12.2(18)S 12.2(27)SBC 12.3(4)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Dynamic Update Peer Groups feature introduces a new algorithm that dynamically calculates and optimizes update groups of neighbors that share the same outbound policies and can share the same update messages. In previous versions of Cisco IOS software, BGP update messages were grouped based on peer-group configurations. This method of grouping updates limited outbound policies and specific-session configurations. The BGP Dynamic Update Peer Group feature separates update group replication from peer group configuration, which improves convergence time and flexibility of neighbor configuration.
BGP Hybrid CLI	12.0(22)S 12.2(15)T 15.0(1)S	The BGP Hybrid CLI feature simplifies the migration of BGP networks and existing configurations from the NLRI format to the AFI format. This new functionality allows the network operator to configure commands in the AFI format and save these command configurations to existing NLRI formatted configurations. The feature provides the network operator with the capability to take advantage of new features and provides support for migration from the NLRI format to the AFI format.

Feature Name	Releases	Feature Configuration Information
BGP Neighbor Policy	12.2(33)SB 12.2(33)SRB 12.4(11)T Cisco IOS XE 3.1.0SG	The BGP Neighbor Policy feature introduces new keywords to two existing commands to display information about local and inherited policies. When BGP neighbors use multiple levels of peer templates, it can be difficult to determine which policies are applied to the neighbor. Inherited policies are policies that the neighbor inherits from a peer-group or a peer-policy template. The following commands were modified by this feature: **show ip bgp neighbors**, **show ip bgp template peer-policy**.

Feature Name	Releases	Feature Configuration Information
BGP Support for 4-Byte ASN	12.0(32)S12 12.0(32)SY8 12.0(33)S3 12.2(33)SRE 12.2(33)XNE 12.2(33)SXI1, 12.4(24)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Support for 4-Byte ASN feature introduced support for 4-byte autonomous system numbers. Because of increased demand for autonomous system numbers, in January 2009 the IANA will start to allocate 4-byte autonomous system numbers in the range from 65536 to 4294967295. In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default regular expression match and output display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain format and the asdot format as described in RFC 5396. To change the default regular expression match and output display of 4-byte autonomous system numbers to asdot format, use the **bgp asnotation dot** command. In Cisco IOS Release 12.0(32)S12, and 12.4(24)T, the Cisco implementation of 4-byte autonomous system numbers uses asdot as the only configuration format, regular expression match, and output display, with no asplain support. The following commands were introduced or modified by this feature: **bgp asnotation dot**, **bgp confederation identifier**, **bgp confederation peers**, all **clear ip bgp**commands that configure an autonomous system number, **ip as-path access-list**, **ip extcommunity-list**, **match source-protocol**, **neighbor local-**

Feature Name	Releases	Feature Configuration Information
		as, **neighbor remote-as**, **neighbor soo**, **redistribute (**IP**)**, **router bgp**, **route-target**, **set as-path**, **set extcommunity**, **set origin**, **soo**, all **show ip bgp** commands that display an autonomous system number, and **show ip extcommunity-list**.
Suppress BGP Advertisement for Inactive Routes	12.2(25)S 12.2(33)SXH 15.0(1)M 15.0(1)S	The Suppress BGP Advertisements for Inactive Routes feature allows you to configure the suppression of advertisements for routes that are not installed in the Routing Information Base (RIB). Configuring this feature allows Border Gateway Protocol (BGP) updates to be more consistent with data used for traffic forwarding.

Connecting to a Service Provider Using External BGP

This module describes configuration tasks that will enable your Border Gateway Protocol (BGP) network to access peer devices in external networks such as those from Internet service providers (ISPs). BGP is an interdomain routing protocol that is designed to provide loop-free routing between organizations. External BGP (eBGP) peering sessions are configured to allow peers from different autonomous systems to exchange routing updates. Tasks to help manage the traffic that is flowing inbound and outbound are described, as are tasks to configure BGP policies to filter the traffic. Multihoming techniques that provide redundancy for connections to a service provider are also described.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Connecting to a Service Provider Using External BGP

- Before connecting to a service provider you need to understand how to configure the basic BGP process and peers. See the "Cisco BGP Overview" and "Configuring a Basic BGP Network" modules for more details.

- The tasks and concepts in this chapter will help you configure BGP features that would be useful if you are connecting your network to a service provider. For each connection to the Internet, you must have an assigned autonomous system number from the Internet Assigned Numbers Authority (IANA).

Restrictions for Connecting to a Service Provider Using External BGP

- A router that runs Cisco IOS software can be configured to run only one BGP routing process and to be a member of only one BGP autonomous system. However, a BGP routing process and autonomous system can support multiple address family configurations.
- Policy lists are not supported in versions of Cisco IOS software prior to Cisco IOS Release 12.0(22)S and 12.2(15)T. Reloading a router that is running an older version of Cisco IOS software may cause some routing policy configurations to be lost.

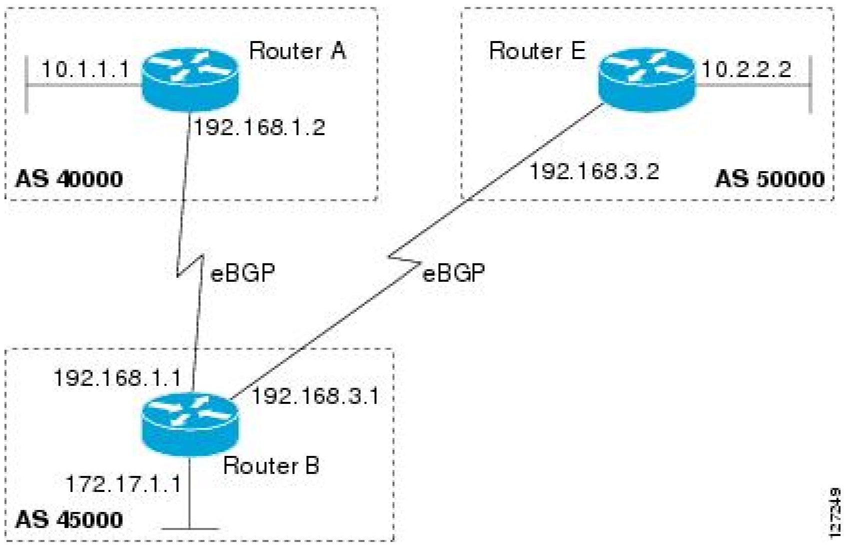
Information About Connecting to a Service Provider Using External BGP

External BGP Peering

BGP is an interdomain routing protocol designed to provide loop-free routing links between organizations. BGP is designed to run over a reliable transport protocol and it uses TCP (port 179) as the transport protocol. The destination TCP port is assigned 179, and the local port is assigned a random port number. Cisco IOS software supports BGP version 4, which has been used by ISPs to help build the Internet. RFC 1771 introduced and discussed a number of new BGP features to allow the protocol to scale for Internet use.

External BGP peering sessions are configured to allow BGP peers from different autonomous systems to exchange routing updates. By design, a BGP routing process expects eBGP peers to be directly connected, for example, over a WAN connection. However, there are many real-world scenarios where this rule would prevent routing from occurring. Peering sessions for multihop neighbors are configured with the **neighbor ebgp-multihop** command. The figure below shows simple eBGP peering between three routers. Router B peers with Router A and Router E. In the figure below, the **neighbor ebgp-multihop** command could be used to establish peering between Router A and Router E although this is a very simple network design. BGP forwards information about the next hop in the network using the NEXT_HOP attribute, which is set to the IP address of the interface that advertises a route in an eBGP peering session by default. The source interface can be a physical interface or a loopback interface.

Figure 12 BGP Peers in Different Autonomous Systems



Loopback interfaces are preferred for establishing eBGP peering sessions because loopback interfaces are less susceptible to interface flapping. Interfaces on networking devices can fail, and they can also be taken out of service for maintenance. When an interface is administratively brought up or down, due to failure or maintenance, it is referred to as a flap. Loopback interfaces provide a stable source interface to ensure that the IP address assigned to the interface is always reachable as long as the IP routing protocols continue to advertise the subnet assigned to the loopback interface. Loopback interfaces allow you to conserve address space by configuring a single address with /32 bit mask. Before a loopback interface is configured for an eBGP peering session, you must configure the **neighbor update-source** command and specify the loopback interface. With this configuration, the loopback interface becomes the source interface and its IP address is advertised as the next hop for routes that are advertised through this loopback. If loopback interfaces are used to connect single-hop eBGP peers, you must configure the **neighbor disable-connected-check** command before you can establish the eBGP peering session.

Connecting to external networks enables traffic from your network to be forwarded to other networks and across the Internet. Traffic will also be flowing into, and possibly through, your network. BGP contains various techniques to influence how the traffic flows into and out of your network, and to create BGP policies that filter the traffic, inbound and outbound. To influence the traffic flow, BGP uses certain BGP attributes that can be included in update messages or used by the BGP routing algorithm. BGP policies to filter traffic also use some of the BGP attributes with route maps, access lists including AS-path access lists, filter lists, policy lists, and distribute lists. Managing your external connections may involve multihoming techniques where there is more than one connection to an ISP or connections to more than one ISP for backup or performance purposes. Tagging BGP routes with different community attributes across autonomous system or physical boundaries can prevent the need to configure long lists of individual permit or deny statements.

BGP Autonomous System Number Formats

Prior to January 2009, BGP autonomous system numbers that were allocated to companies were 2-octet numbers in the range from 1 to 65535 as described in RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*. Due to increased demand for autonomous system numbers, the Internet Assigned Number Authority (IANA) will start in January 2009 to allocate four-octet autonomous system numbers in the range from 65536 to 4294967295. RFC 5396, *Textual Representation of Autonomous System (AS) Numbers* , documents three methods of representing autonomous system numbers. Cisco has implemented the following two methods:

- Asplain--Decimal value notation where both 2-byte and 4-byte autonomous system numbers are represented by their decimal value. For example, 65526 is a 2-byte autonomous system number and 234567 is a 4-byte autonomous system number.
- Asdot--Autonomous system dot notation where 2-byte autonomous system numbers are represented by their decimal value and 4-byte autonomous system numbers are represented by a dot notation. For example, 65526 is a 2-byte autonomous system number and 1.169031 is a 4-byte autonomous system number (this is dot notation for the 234567 decimal number).

For details about the third method of representing autonomous system numbers, see RFC 5396.

Asdot Only Autonomous System Number Formatting

In Cisco IOS Release 12.0(32)S12, 12.4(24)T, and later releases, the 4-octet (4-byte) autonomous system numbers are entered and displayed only in asdot notation, for example, 1.10 or 45000.64000. When using regular expressions to match 4-byte autonomous system numbers the asdot format includes a period which is a special character in regular expressions. A backslash must be entered before the period for example, 1\. 14, to ensure the regular expression match does not fail. The table below shows the format in which 2-byte and 4-byte autonomous system numbers are configured, matched in regular expressions, and displayed in **show** command output in Cisco IOS images where only asdot formatting is available.

Table 12 Asdot Only 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Asplain as Default Autonomous System Number Formatting

In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain and asdot format. In addition, the default format for matching 4-byte autonomous system numbers in regular expressions is asplain, so you must ensure that any regular expressions to match 4-byte autonomous system numbers are written in the asplain format. If you want to change the default **show** command output to display 4-byte autonomous system numbers in the asdot format, use the **bgp asnotation dot** command under router configuration mode. When the asdot format is enabled as the default, any regular expressions to match 4-byte autonomous system numbers must be written using the asdot format, or the regular expression match will fail. The tables below show that although you can configure 4-byte autonomous system numbers in either asplain or asdot format, only one format is used to display **show** command output and control 4-byte autonomous system number matching for regular expressions, and the default is asplain format. To display 4-byte autonomous system numbers in **show** command output and to control matching for regular expressions in the asdot format, you must configure the **bgp asnotation dot** command. After enabling the **bgp asnotation dot** command, a hard reset must be initiated for all BGP sessions by entering the **clear ip bgp *** command.

Note If you are upgrading to an image that supports 4-byte autonomous system numbers, you can still use 2-byte autonomous system numbers. The **show** command output and regular expression match are not changed and remain in asplain (decimal value) format for 2-byte autonomous system numbers regardless of the format configured for 4-byte autonomous system numbers.

Table 13 Default Asplain 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 65536 to 4294967295
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 65536 to 4294967295

Table 14 Asdot 4-Byte Autonomous System Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Reserved and Private Autonomous System Numbers

In Cisco IOS Release 12.0(32)S12, 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, 12.4(24)T, and later releases, the Cisco implementation of BGP supports RFC 4893. RFC 4893 was developed to allow BGP to support a gradual transition from 2-byte autonomous system numbers to 4-byte autonomous system numbers. A new reserved (private) autonomous system number, 23456, was created by RFC 4893 and this number cannot be configured as an autonomous system number in the Cisco IOS CLI.

RFC 5398, *Autonomous System (AS) Number Reservation for Documentation Use*, describes new reserved autonomous system numbers for documentation purposes. Use of the reserved numbers allow configuration examples to be accurately documented and avoids conflict with production networks if these configurations are literally copied. The reserved numbers are documented in the IANA autonomous system number registry. Reserved 2-byte autonomous system numbers are in the contiguous block, 64496 to 64511 and reserved 4-byte autonomous system numbers are from 65536 to 65551 inclusive.

Private 2-byte autonomous system numbers are still valid in the range from 64512 to 65534 with 65535 being reserved for special use. Private autonomous system numbers can be used for internal routing domains but must be translated for traffic that is routed out to the Internet. BGP should not be configured to advertise private autonomous system numbers to external networks. Cisco IOS software does not remove private autonomous system numbers from routing updates by default. We recommend that ISPs filter private autonomous system numbers.

Note Autonomous system number assignment for public and private networks is governed by the IANA. For information about autonomous-system numbers, including reserved number assignment, or to apply to register an autonomous system number, see the following URL: http://www.iana.org/.

BGP Attributes

BGP selects a single path, by default, as the best path to a destination host or network. The best-path selection algorithm analyzes path attributes to determine which route is installed as the best path in the BGP routing table. Each path carries various attributes that are used in BGP best-path analysis. Cisco IOS software provides the ability to influence BGP path selection by altering these attributes via the command-line interface (CLI). BGP path selection can also be influenced through standard BGP policy configuration.

BGP uses the best-path selection algorithm to find a set of equally good routes. These routes are the potential multipaths. In Cisco IOS Release 12.2(33)SRD and later releases, when there are more equally good multipaths available than the maximum permitted number, then the oldest paths are selected as multipaths.

BGP can include path attribute information in update messages. BGP attributes describe the characteristic of the route, and the software uses these attributes to help make decisions about which routes to advertise. Some of this attribute information can be configured at a BGP-speaking networking device. There are some mandatory attributes that are always included in the update message and some discretionary attributes. The following BGP attributes can be configured:

- AS-path
- Community
- Local_Pref
- Multi_Exit_Discriminator (MED)
- Next_Hop
- Origin

AS-path

This attribute contains a list or set of the autonomous system numbers through which routing information has passed. The BGP speaker adds its own autonomous system number to the list when it forwards the update message to external peers.

Community

BGP communities are used to group networking devices that share common properties, regardless of network, autonomous system, or any physical boundaries. In large networks applying a common routing policy through prefix lists or access lists requires individual peer statements on each networking device. Using the BGP community attribute BGP neighbors, with common routing policies, can implement inbound or outbound route filters based on the community tag rather than consult large lists of individual permit or deny statements.

Local_Pref

Within an autonomous system, the Local_Pref attribute is included in all update messages between BGP peers. If there are several paths to the same destination, the local preference attribute with the highest value indicates the preferred outbound path from the local autonomous system. The highest ranking route is advertised to internal peers. The Local_Pref value is not forwarded to external peers.

Multi_Exit_Discriminator

The MED attribute indicates (to an external peer) a preferred path into an autonomous system. If there are multiple entry points into an autonomous system, the MED can be used to influence another autonomous system to choose one particular entry point. A metric is assigned where a lower MED metric is preferred by the software over a higher MED metric. The MED metric is exchanged between autonomous systems, but after a MED is forwarded into an autonomous system, the MED metric is reset to the default value of 0. When an update is sent to an internal BGP (iBGP) peer, the MED is passed along without any change, allowing all the peers in the same autonomous system to make a consistent path selection.

By default, a router will compare the MED attribute for paths only from BGP peers that reside in the same autonomous system. The **bgp always-compare-med** command can be configured to allow the router to compare metrics from peers in different autonomous systems.

Note The Internet Engineering Task Force (IETF) decision regarding BGP MED assigns a value of infinity to the missing MED, making the route that lacks the MED variable the least preferred. The default behavior of BGP routers that run Cisco IOS software is to treat routes without the MED attribute as having a MED of 0, making the route that lacks the MED variable the most preferred. To configure the router to conform to the IETF standard, use the **bgp bestpath med missing-as-worst**router configuration command.

Next_Hop

The Next_Hop attribute identifies the next-hop IP address to be used as the BGP next hop to the destination. The router makes a recursive lookup to find the BGP next hop in the routing table. In external BGP (eBGP), the next hop is the IP address of the peer that sent the update. Internal BGP (iBGP) sets the next-hop address to the IP address of the peer that advertised the prefix for routes that originate internally. When any routes to iBGP that are learned from eBGP are advertised, the Next_Hop attribute is unchanged.

A BGP next-hop IP address must be reachable in order for the router to use a BGP route. Reachability information is usually provided by the IGP, and changes in the IGP can influence the forwarding of the next-hop address over a network backbone.

Origin

This attribute indicates how the route was included in a BGP routing table. In Cisco IOS software, a route defined using the BGP **network** command is given an origin code of Interior Gateway Protocol (IGP). Routes distributed from an Exterior Gateway Protocol (EGP) are coded with an origin of EGP, and routes redistributed from other protocols are defined as Incomplete. BGP decision policy for origin prefers IGP over EGP, and then EGP over Incomplete.

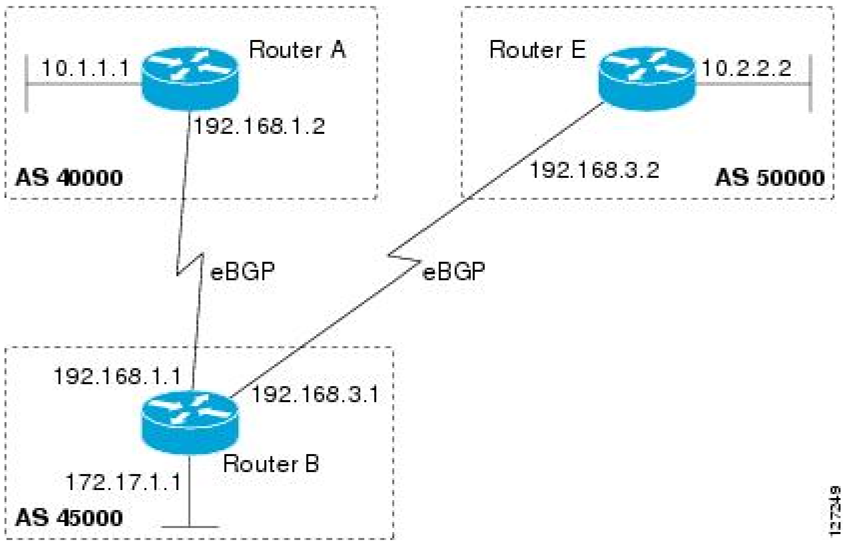
Multihoming

Multihoming is defined as connecting an autonomous system with more than one service provider. If you have any reliability issues with one service provider, then you have a backup connection. Performance issues can also be addressed by multihoming because better paths to the destination network can be utilized.

Unless you are a service provider, you must plan your routing configuration carefully to avoid Internet traffic traveling through your autonomous system and consuming all your bandwidth. The figure below shows that autonomous system 45000 is multihomed to autonomous system 40000 and autonomous system

50000. Assuming autonomous system 45000 is not a service provider, then several techniques such as load balancing or some form of routing policy must be configured to allow traffic from autonomous system 45000 to reach either autonomous system 40000 or autonomous system 50000 but not allow much, if any, transit traffic.

Figure 13 Multihoming Topology



MED Attribute

Configuring the MED attribute is another method that BGP can use to influence the choice of paths into another autonomous system. The MED attribute indicates (to an external peer) a preferred path into an autonomous system. If there are multiple entry points into an autonomous system, the MED can be used to influence another autonomous system to choose one particular entry point. A metric is assigned using route maps where a lower MED metric is preferred by the software over a higher MED metric.

Transit Versus Nontransit Traffic

Most of the traffic within an autonomous system contains a source or destination IP address residing within the autonomous system, and this traffic is referred to as nontransit (or local) traffic. Other traffic is defined as transit traffic. As traffic across the Internet increases, controlling transit traffic becomes more important.

A service provider is considered to be a transit autonomous system and must provide connectivity to all other transit providers. In reality, few service providers actually have enough bandwidth to allow all transit traffic, and most service providers have to purchase such connectivity from Tier 1 service providers.

An autonomous system that does not usually allow transit traffic is called a stub autonomous system and will link to the Internet through one service provider.

BGP Policy Configuration

BGP policy configuration is used to control prefix processing by the BGP routing process and to filter routes from inbound and outbound advertisements. Prefix processing can be controlled by adjusting BGP timers, altering how BGP handles path attributes, limiting the number of prefixes that the routing process will accept, and configuring BGP prefix dampening. Prefixes in inbound and outbound advertisements are filtered using route maps, filter lists, IP prefix lists, autonomous-system-path access lists, IP policy lists, and distribute lists. The table below shows the processing order of BGP policy filters.

Table 15 BGP Policy Processing Order

Inbound	Outbound
Route map	Distribute list
Filter list, AS-path access list, or IP policy	IP prefix list
IP prefix list	Filter list, AS-path access list, or IP policy
Distribute list	Route map

Note In Cisco IOS Releases 12.0(22)S, 12.2(15)T, 12.2(18)S, and later releases, the maximum number of autonomous system access lists that can be configured with the **ip as-path access-list** command is increased from 199 to 500.

Whenever there is a change in the routing policy due to a configuration change, BGP peering sessions must be reset using the **clear ip bgp** command. Cisco IOS software supports the following three mechanisms to reset BGP peering sessions:

- Hard reset--A hard reset tears down the specified peering sessions, including the TCP connection, and deletes routes coming from the specified peer.
- Soft reset--A soft reset uses stored prefix information to reconfigure and activate BGP routing tables without tearing down existing peering sessions. Soft reset uses stored update information, at the cost of additional memory for storing the updates, to allow you to apply a new BGP policy without disrupting the network. Soft reset can be configured for inbound or outbound sessions.
- Dynamic inbound soft reset--The route refresh capability, as defined in RFC 2918, allows the local router to reset inbound routing tables dynamically by exchanging route refresh requests to supporting peers. The route refresh capability does not store update information locally for nondisruptive policy changes. It instead relies on dynamic exchange with supporting peers. Route refresh must first be advertised through BGP capability negotiation between peers. All BGP routers must support the route refresh capability.

To determine if a BGP router supports this capability, use the **show ip bgp neighbors**command. The following message is displayed in the output when the router supports the route refresh capability:

```
Received route refresh capability from peer.
```

BGP Prefix-Based Outbound Route Filtering

BGP prefix-based outbound route filtering uses the BGP ORF send and receive capabilities to minimize the number of BGP updates that are sent between BGP peers. Configuring BGP ORF can help reduce the amount of system resources required for generating and processing routing updates by filtering out unwanted routing updates at the source. For example, BGP ORF can be used to reduce the amount of processing required on a router that is not accepting full routes from a service provider network.

The BGP prefix-based outbound route filtering is enabled through the advertisement of ORF capabilities to peer routers. The advertisement of the ORF capability indicates that a BGP peer will accept a prefix list from a neighbor and apply the prefix list to locally configured ORFs (if any exist). When this capability is enabled, the BGP speaker can install the inbound prefix list filter to the remote peer as an outbound filter, which reduces unwanted routing updates.

The BGP prefix-based outbound route filtering can be configured with send or receive ORF capabilities. The local peer advertises the ORF capability in send mode. The remote peer receives the ORF capability in receive mode and applies the filter as an outbound policy. The local and remote peers exchange updates to maintain the ORF on each router. Updates are exchanged between peer routers by address family depending on the ORF prefix list capability that is advertised. The remote peer starts sending updates to the local peer after a route refresh has been requested with the **clear ip bgp in prefix-filter**command or after an ORF prefix list with immediate status is processed. The BGP peer will continue to apply the inbound prefix list to received updates after the local peer pushes the inbound prefix list to the remote peer.

BGP Communities

BGP communities are used to group routes (also referred to as color routes) that share common properties, regardless of network, autonomous system, or any physical boundaries. In large networks applying a common routing policy through prefix-lists or access-lists requires individual peer statements on each networking device. Using the BGP community attribute BGP speakers, with common routing policies, can implement inbound or outbound route filters based on the community tag rather than consult large lists of individual permit or deny statements.

Standard community lists are used to configure well-known communities and specific community numbers. Expanded community lists are used to filter communities using a regular expression. Regular expressions are used to configure patterns to match community attributes.

The community attribute is optional, which means that it will not be passed on by networking devices that do not understand communities. Networking devices that understand communities must be configured to handle the communities or they will be discarded.

There are four predefined communities:

- no-export--Do not advertise to external BGP peers.
- no-advertise--Do not advertise this route to any peer.
- internet--Advertise this route to the Internet community; all BGP-speaking networking devices belong to it.
- local-as--Do not send outside the local autonomous system.

In Cisco IOS Release 12.2(8)T, BGP named community lists were introduced. BGP named community lists allow meaningful names to be assigned to community lists with no limit on the number of community lists that can be configured. A named community list can be configured with regular expressions and with numbered community lists. All the rules of numbered communities apply to named community lists except that there is no limitation on the number of named community lists that can be configured.

Note Both standard and expanded community lists have a limitation of 100 community groups that can be configured within each type of list. A named community list does not have this limitation.

Extended Communities

Extended community attributes are used to configure, filter, and identify routes for virtual routing and forwarding (VRF) instances and Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs). All of the standard rules of access lists apply to the configuration of extended community lists. Regular expressions are supported by the expanded range of extended community list numbers. All regular expression configuration options are supported. The route target (RT) and site of origin (SoO) extended community attributes are supported by the standard range of extended community lists.

Route Target Extended Community Attribute

The RT extended community attribute is configured with the **rt** keyword of the **ip extcommunity-list**command. This attribute is used to identify a set of sites and VRFs that may receive routes that are tagged with the configured route target. Configuring the route target extended community attribute with a route allows that route to be placed in the per-site forwarding tables that are used for routing traffic that is received from corresponding sites.

Site of Origin Extended Community Attribute

The SoO extended community attribute is configured with the **soo** keyword of the **ip extcommunity-list**command. This attribute uniquely identifies the site from which the provider edge (PE) router learned the route. All routes learned from a particular site must be assigned the same SoO extended community attribute, regardless if a site is connected to a single PE router or multiple PE routers. Configuring this attribute prevents routing loops from occurring when a site is multihomed. The SoO extended community attribute is configured on the interface and is propagated into BGP through redistribution. The SoO extended community attribute can be applied to routes that are learned from VRFs. The SoO extended community attribute should not be configured for stub sites or sites that are not multihomed.

IP Extended Community-List Configuration Mode

Named and numbered extended community lists can be configured in IP extended community-list configuration mode. The IP extended community-list configuration mode supports all of the functions that are available in global configuration mode. In addition, the following operations can be performed:

- Configure sequence numbers for extended community list entries.
- Resequence existing sequence numbers for extended community list entries.
- Configure an extended community list to use default values.

Default Sequence Numbering

Extended community list entries start with the number 10 and increment by 10 for each subsequent entry when no sequence number is specified, when default behavior is configured, and when an extended community list is resequenced without specifying the first entry number or the increment range for subsequent entries.

Resequencing Extended Community Lists

Extended community-list entries are sequenced and resequenced on a per-extended community list basis. The **resequence** command can be used without any arguments to set all entries in a list to default sequence numbering. The **resequence** command also allows the sequence number of the first entry and increment range to be set for each subsequent entry. The range of configurable sequence numbers is from 1 to 2147483647.

Extended Community Lists

Extended community attributes are used to configure, filter, and identify routes for VRF instances and MPLS VPNs. The **ip extcommunity-list**command is used to configure named or numbered extended community lists. All of the standard rules of access lists apply to the configuration of extended community lists. Regular expressions are supported by the expanded range of extended community list numbers.

Administrative Distance

Administrative distance is a measure of the preference of different routing protocols. BGP has a **distance bgp** command that allows you to set different administrative distances for three route types: external, internal, and local. BGP, like other protocols, prefers the route with the lowest administrative distance.

BGP Route Map Policy Lists

BGP route map policy lists allow a network operator to group route map match clauses into named lists called policy lists. A policy list functions like a macro. When a policy list is referenced in a route map, all of the match clauses are evaluated and processed as if they had been configured directly in the route map. This enhancement simplifies the configuration of BGP routing policy in medium-size and large networks because a network operator can preconfigure policy lists with groups of match clauses and then reference these policy lists within different route maps. The network operator no longer needs to manually reconfigure each recurring group of match clauses that occur in multiple route map entries.

A policy lists functions like a macro when it is configured in a route map and has the following capabilities and characteristics:

- When a policy list is referenced within a route map, all the match statements within the policy list are evaluated and processed.
- Two or more policy lists can be configured with a route map. Policy lists can be configured within a route map to be evaluated with AND or OR semantics.
- Policy lists can coexist with any other preexisting match and set statements that are configured within the same route map but outside of the policy lists.
- When multiple policy lists perform matching within a route map entry, all policy lists match on the incoming attribute only.

Policy lists support only match clauses and do not support set clauses. Policy lists can be configured for all applications of route maps, including redistribution, and can also coexist, within the same route map entry, with match and set clauses that are configured separately from the policy lists.

Note Policy lists are supported only by BGP and are not supported by other IP routing protocols.

BGP Route Map with a Continue Clause

In Cisco IOS Release 12.3(2)T, 12.0(24)S, 12.2(33)SRB, and later releases, the continue clause was introduced into BGP route map configuration. The continue clause allows for more programmable policy configuration and route filtering and introduced the capability to execute additional entries in a route map after an entry is executed with successful match and set clauses. Continue clauses allow the network operator to configure and organize more modular policy definitions so that specific policy configurations need not be repeated within the same route map. Before the continue clause was introduced, route map configuration was linear and did not allow any control over the flow of a route map.

In Cisco IOS Release 12.0(31)S, 12.2(33)SB, 12.2(33)SRB, 12.2(33)SXI, 12.4(4)T, and later releases, support for continue clauses for outbound route maps was introduced.

- Route Map Operation Without Continue Clauses, page 139
- Route Map Operation with Continue Clauses, page 139
- Match Operations with Continue Clauses, page 139

Route Map Operation Without Continue Clauses

A route map evaluates match clauses until a successful match occurs. After the match occurs, the route map stops evaluating match clauses and starts executing set clauses, in the order in which they were configured. If a successful match does not occur, the route map "falls through" and evaluates the next sequence number of the route map until all configured route map entries have been evaluated or a successful match occurs. Each route map sequence is tagged with a sequence number to identify the entry. Route map entries are evaluated in order starting with the lowest sequence number and ending with the highest sequence number. If the route map contains only set clauses, the set clauses will be executed automatically, and the route map will not evaluate any other route map entries.

Route Map Operation with Continue Clauses

When a continue clause is configured, the route map will continue to evaluate and execute match clauses in the specified route map entry after a successful match occurs. The continue clause can be configured to go to (or jump to) a specific route map entry by specifying the sequence number, or if a sequence number is not specified, the continue clause will go to the next sequence number. This behavior is called an "implied continue." If a match clause exists, the continue clause is executed only if a match occurs. If no successful matches occur, the continue clause is ignored.

Match Operations with Continue Clauses

If a match clause does not exist in the route map entry but a continue clause does, the continue clause will be automatically executed and go to the specified route map entry. If a match clause exists in a route map entry, the continue clause is executed only when a successful match occurs. When a successful match occurs and a continue clause exists, the route map executes the set clauses and then goes to the specified route map entry. If the next route map entry contains a continue clause, the route map will execute the continue clause if a successful match occurs. If a continue clause does not exist in the next route map entry, the route map will be evaluated normally. If a continue clause exists in the next route map entry but a match does not occur, the route map will not continue and will "fall through" to the next sequence number if one exists.

Set Operations with Continue Clauses

Set clauses are saved during the match clause evaluation process and executed after the route-map evaluation is completed. The set clauses are evaluated and executed in the order in which they were configured. Set clauses are executed only after a successful match occurs, unless the route map does not contain a match clause. The continue statement proceeds to the specified route map entry only after configured set actions are performed. If a set action occurs in the first route map and then the same set action occurs again, with a different value, in a subsequent route map entry, the last set action may override any previous set actions that were configured with the same **set** command unless the **set** command permits more than one value. For example, the **set as-path prepend** command permits more than one autonomous system number to be configured.

Note A continue clause can be executed, without a successful match, if a route map entry does not contain a match clause.

Note Route maps have a linear behavior and not a nested behavior. Once a route is matched in a route map permit entry with a continue command clause, it will not be processed by the implicit deny at the end of the route-map. For an example, see "Filtering Traffic Using Continue Clauses in a BGP Route Map Examples".

Restrictions

- Continue clauses for outbound route maps are supported only in Cisco IOS Release 12.0(31)S, 12.2(33)SB, 12.2(33)SRB, 12.2(33)SXI, 12.4(4)T, and later releases.
- Continue clauses can go only to a higher route map entry (a route map entry with a higher sequence number) and cannot go to a lower route map entry.

How to Connect to a Service Provider Using External BGP

Influencing Inbound Path Selection

BGP can be used to influence the choice of paths in another autonomous system. There may be several reasons for wanting BGP to choose a path that is not the obvious best route, for example, to avoid some types of transit traffic passing through an autonomous system or perhaps to avoid a very slow or congested link. BGP can influence inbound path selection using one of the following BGP attributes:

- AS-path
- MED

Perform one of the following tasks to influence inbound path selection:

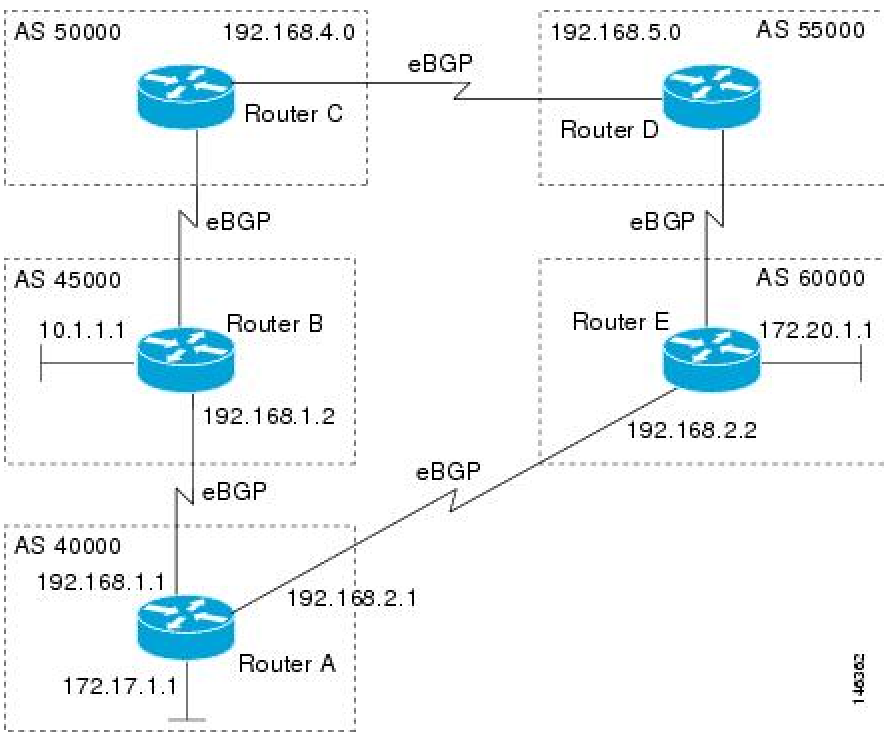
Influencing Inbound Path Selection by Modifying the AS-path Attribute

Perform this task to influence the inbound path selection for traffic destined for the 172.17.1.0 network by modifying the AS-path attribute. The configuration is performed at Router A in the figure below. For a configuration example of this task using 4-byte autonomous system numbers in asplain format, see Influencing Inbound Path Selection by Modifying the AS-path Attribute Using 4-Byte AS Numbers Example, page 199.

One of the methods that BGP can use to influence the choice of paths in another autonomous system is to modify the AS-path attribute. For example, in the figure below, Router A advertises its own network, 172.17.1.0, to its BGP peers in autonomous system 45000 and autonomous system 60000. When the routing information is propagated to autonomous system 50000, the routers in autonomous system 50000 have network reachability information about network 172.17.1.0 from two different routes. The first route is from autonomous system 45000 with an AS-path consisting of 45000, 40000, the second route is through autonomous system 55000 with an AS-path of 55000, 60000, 40000. If all other BGP attribute values are the same, Router C in autonomous system 50000 would choose the route through autonomous system 45000 for traffic destined for network 172.17.1.0 because it is the shortest route in terms of autonomous systems traversed.

Autonomous system 40000 now receives all traffic from autonomous system 50000 for the 172.17.1.0 network through autonomous system 45000. If, however, the link between autonomous system 45000 and autonomous system 40000 is a really slow and congested link, the **set as-path prepend**command can be used at Router A to influence inbound path selection for the 172.17.1.0 network by making the route through autonomous system 45000 appear to be longer than the path through autonomous system 60000. The configuration is done at Router A in the figure below by applying a route map to the outbound BGP updates to Router B. Using the **set as-path prepend**command, all the outbound BGP updates from Router A to Router B will have their AS-path attribute modified to add the local autonomous system number 40000 twice. After the configuration, autonomous system 50000 receives updates about the 172.17.1.0 network through autonomous system 45000. The new AS-path is 45000, 40000, 40000, and 40000, which is now longer than the AS-path from autonomous system 55000 (unchanged at a value of 55000, 60000, 40000). Networking devices in autonomous system 50000 will now prefer the route through autonomous system 55000 to forward packets with a destination address in the 172.17.1.0 network.

Figure 14 Network Topology for Modifying the AS-path Attribute



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
7. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
8. **neighbor** {*ip-address*| *peer-group-name*} **activate**
9. **exit-address-family**
10. **exit**
11. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
12. **set as-path** {**tag** | **prepend** *as-path-string*}
13. **end**
14. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • In this example, the BGP peer on Router B at 192.168.1.2 is added to the IPv4 multiprotocol BGP neighbor table and will receive BGP updates.

	Command or Action	Purpose
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The unicast keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.1.2 route-map PREPEND out`	Applies a route map to incoming or outgoing routes. • In this example, the route map named PREPEND is applied to outbound routes to Router B.
Step 8	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.2 activate`	Enables address exchange for address family IPv4 unicast for the BGP neighbor at 192.168.1.2 on Router B.
Step 9	**exit-address-family** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 10	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 11	**route-map** *map-name* [**permit**| **deny**][*sequence-number*] **Example:** `Router(config)# route-map PREPEND permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named PREPEND is created with a permit clause.
Step 12	**set as-path** {**tag** | **prepend** *as-path-string*} **Example:** `Router(config-route-map)# set as-path prepend 40000 40000`	Modifies an autonomous system path for BGP routes. • Use the **prepend** keyword to "prepend" an arbitrary autonomous system path string to BGP routes. Usually the local autonomous system number is prepended multiple times, increasing the autonomous system path length. • In this example, two additional autonomous system entries are added to the autonomous system path for outbound routes to Router B.
Step 13	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and returns to privileged EXEC mode.
Step 14	**show running-config** **Example:** `Router# show running-config`	Displays the running configuration file.

Examples

Router A

The following partial output of the **show running-config** command shows the configuration from this task.

```
Router# show running-config
.
.
.
router bgp 40000
 neighbor 192.168.1.2 remote-as 45000
 !
 address-family ipv4
 neighbor 192.168.1.2 activate
 neighbor 192.168.1.2 route-map PREPEND out
 no auto-summary
 no synchronization
 network 172.17.1.0 mask 255.255.255.0
 exit-address-family
!
route-map PREPEND permit 10
 set as-path prepend 40000 40000
```

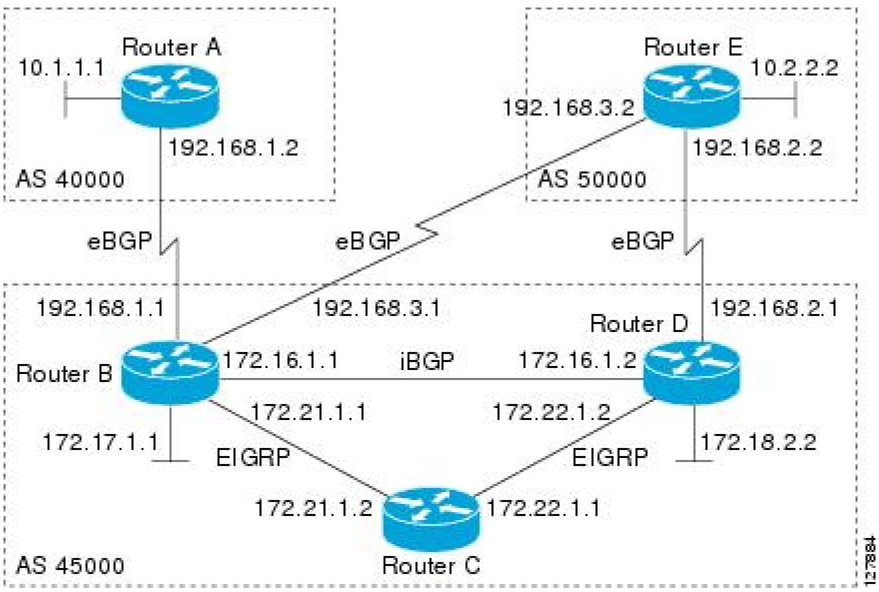
.
.
.

Influencing Inbound Path Selection by Setting the MED Attribute

One of the methods that BGP can use to influence the choice of paths into another autonomous system is to set the MED attribute. The MED attribute indicates (to an external peer) a preferred path to an autonomous system. If there are multiple entry points to an autonomous system, the MED can be used to influence another autonomous system to choose one particular entry point. A metric is assigned using route maps where a lower MED metric is preferred by the software over a higher MED metric.

Perform this task to influence inbound path selection by setting the MED metric attribute. The configuration is performed at Router B and Router D in the figure below. Router B advertises the network 172.16.1.0. to its BGP peer, Router E in autonomous system 50000. Using a simple route map Router B sets the MED metric to 50 for outbound updates. The task is repeated at Router D but the MED metric is set to 120. When Router E receives the updates from both Router B and Router D the MED metric is stored in the BGP routing table. Before forwarding packets to network 172.16.1.0, Router E compares the attributes from peers in the same autonomous system (both Router B and Router D are in autonomous system 45000). The MED metric for Router B is less than the MED for Router D, so Router E will forward the packets through Router B.

Figure 15 Network Topology for Setting the MED Attribute



Use the **bgp always-compare-med** command to compare MED attributes from peers in other autonomous systems.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
7. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
8. **exit**
9. **exit**
10. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
11. **set metric** *value*
12. **end**
13. Repeat Step 1 through Step 12 at Router D.
14. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.

	Command or Action	Purpose
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Router(config-router-af)# network 172.16.1.0 mask 255.255.255.0`	Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.3.2 route-map MED out`	Applies a route map to incoming or outgoing routes. • In this example, the route map named MED is applied to outbound routes to the BGP peer at Router E.
Step 8	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 9	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 10	**route-map** *map-name* [**permit**| **deny**][*sequence-number*] **Example:** `Router(config)# route-map MED permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named MED is created.

	Command or Action	Purpose
Step 11	**set metric** *value* **Example:** `Router(config-route-map)# set metric 50`	Sets the MED metric value.
Step 12	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.
Step 13	Repeat Step 1 through Step 12 at Router D.	--
Step 14	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp 172.17.1.0 255.255.255.0`	(Optional) Displays the entries in the BGP routing table. • Use this command at Router E in the figure above when both Router B and Router D have configured the MED attribute. • Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following output is from Router E in the figure above after this task has been performed at both Router B and Router D. Note the metric (MED) values for the two routes to network 172.16.1.0. The peer 192.168.2.1 at Router D has a metric of 120 for the path to network 172.16.1.0 whereas the peer 192.168.3.1 at Router B has a metric of 50. The entry for the peer 192.168.3.1 at Router B has the word best at the end of the entry to show that Router E will choose to send packets destined for network 172.16.1.0 via Router B because the MED metric is lower.

```
Router# show ip bgp 172.16.1.0
BGP routing table entry for 172.16.1.0/24, version 10
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to update-groups:
     1
  45000
    192.168.2.1 from 192.168.2.1 (192.168.2.1)
      Origin IGP, metric 120, localpref 100, valid, external
  45000
    192.168.3.1 from 192.168.3.1 (172.17.1.99)
      Origin IGP, metric 50, localpref 100, valid, external, best
```

Influencing Outbound Path Selection

BGP can be used to influence the choice of paths for outbound traffic from the local autonomous system. This section contains two methods that BGP can use to influence outbound path selection:

- Using the Local_Pref attribute
- Using the BGP outbound route filter (ORF) capability

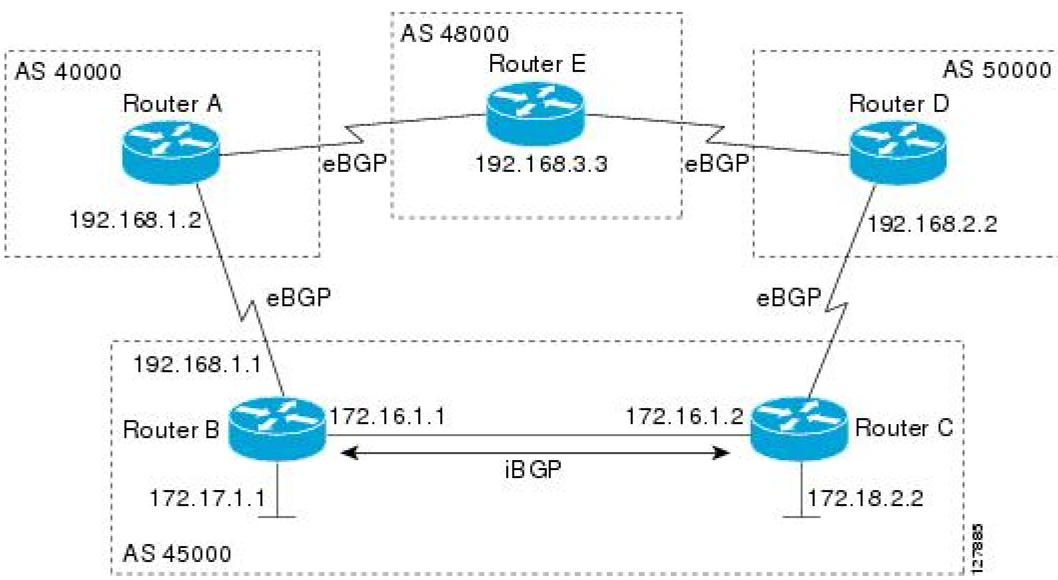
Perform one of the following tasks to influence outbound path selection:

Influencing Outbound Path Selection Using the Local_Pref Attribute

One of the methods to influence outbound path selection is to use the BGP Local-Pref attribute. Perform this task using the local preference attribute to influence outbound path selection. If there are several paths to the same destination the local preference attribute with the highest value indicates the preferred path.

Refer to the figure below for the network topology used in this task. Both Router B and Router C are configured. autonomous system 45000 receives updates for network 192.168.3.0 via autonomous system 40000 and autonomous system 50000. Router B is configured to set the local preference value to 150 for all updates to autonomous system 40000. Router C is configured to set the local preference value for all updates to autonomous system 50000 to 200. After the configuration, local preference information is exchanged within autonomous system 45000. Router B and Router C now see that updates for network 192.168.3.0 have a higher preference value from autonomous system 50000 so all traffic in autonomous system 45000 with a destination network of 192.168.3.0 is sent out via Router C.

Figure 16 Network Topology for Outbound Path Selection



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
5. **bgp default local-preference** *value*
6. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
7. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
8. **neighbor** {*ip-address*| *peer-group-name*} **activate**
9. **end**
10. Repeat Step 1 through Step 9 at Router C but change the IP address of the peer, the autonomous system number, and set the local preference value to 200.
11. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.

	Command or Action	Purpose
Step 5	**bgp default local-preference** *value* **Example:** `Router(config-router)# bgp default local-preference 150`	Changes the default local preference value. • In this example, the local preference is changed to 150 for all updates from autonomous system 40000 to autonomous system 45000. • By default, the local preference value is 100.
Step 6	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 7	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.
Step 8	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.2 activate`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 9	**end** **Example:** `Router(config-router-af)# end`	Exits route map configuration mode and enters privileged EXEC mode.
Step 10	Repeat Step 1 through Step 9 at Router C but change the IP address of the peer, the autonomous system number, and set the local preference value to 200.	--

	Command or Action	Purpose
Step 11	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp 192.168.3.0 255.255.255.0`	Displays the entries in the BGP routing table. • Enter this command at both Router B and Router C and note the Local_Pref value. The route with the highest preference value will be the preferred route to network 192.168.3.0. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Filtering Outbound BGP Route Prefixes

Perform this task to use BGP prefix-based outbound route filtering to influence outbound path selection.

BGP peering sessions must be established, and BGP ORF capabilities must be enabled on each participating router before prefix-based ORF announcements can be received.

Note

- BGP prefix-based outbound route filtering does not support multicast.
- IP addresses that are used for outbound route filtering must be defined in an IP prefix list. BGP distribute lists and IP access lists are not supported.
- Outbound route filtering is configured on only a per-address family basis and cannot be configured under the general session or BGP routing process.
- Outbound route filtering is configured for external peering sessions only.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*]
4. **router bgp** autonomous-system-*number*
5. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
6. **neighbor** *ip-address* **ebgp-multihop** [*hop-count*]
7. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
8. **neighbor** *ip-address* **capability orf prefix-list** [**send** | **receive** | **both**]
9. **neighbor** {*ip-address*| *peer-group-name*} **prefix-list** *prefix-list-name*{**in** | **out**}
10. **end**
11. **clear ip bgp** {*ip-address* | *****} **in prefix-filter**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list FILTER seq 10 permit 192.168.1.0/24`	Creates a prefix list for prefix-based outbound route filtering. • Outbound route filtering supports prefix length matching, wildcard-based prefix matching, and exact address prefix matching on a per address-family basis. • The prefix list is created to define the outbound route filter. The filter must be created when the outbound route filtering capability is configured to be advertised in send mode or both mode. It is not required when a peer is configured to advertise receive mode only. • The example creates a prefix list named FILTER that defines the 192.168.1.0/24 subnet for outbound route filtering.
Step 4	**router bgp** autonomous-system-*number* **Example:** `Router(config)# router bgp 100`	Enters router configuration mode, and creates a BGP routing process.
Step 5	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 10.1.1.1 remote-as 200`	Establishes peering with the specified neighbor or peer group. BGP peering must be established before ORF capabilities can be exchanged. • The example establishes peering with the 10.1.1.1 neighbor.
Step 6	**neighbor** *ip-address* **ebgp-multihop** [*hop-count*] **Example:** `Router(config-router)# neighbor 10.1.1.1 ebgp-multihop`	Accepts or initiates BGP connections to external peers residing on networks that are not directly connected.

	Command or Action	Purpose
Step 7	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands. **Note** Outbound route filtering is configured on a per-address family basis.
Step 8	**neighbor** *ip-address* **capability orf prefix-list** [**send** | **receive** | **both**] **Example:** `Router(config-router-af)# neighbor 10.1.1.1 capability orf prefix-list both`	Enables the ORF capability on the local router, and enables ORF capability advertisement to the BGP peer specified with the *ip-address* argument. • The **send** keyword configures a router to advertise ORF send capabilities. • The **receive** keyword configures a router to advertise ORF receive capabilities. • The **both** keyword configures a router to advertise send and receive capabilities. • The remote peer must be configured to either send or receive ORF capabilities before outbound route filtering is enabled. • The example configures the router to advertise send and receive capabilities to the 10.1.1.1 neighbor.
Step 9	**neighbor** {*ip-address*| *peer-group-name*} **prefix-list** *prefix-list-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 10.1.1.1 prefix-list FILTER in`	Applies an inbound prefix-list filter to prevent distribution of BGP neighbor information. • In this example, the prefix list named FILTER is applied to incoming advertisements from the 10.1.1.1 neighbor, which prevents distribution of the 192.168.1.0/24 subnet.
Step 10	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode, and enters privileged EXEC mode.

	Command or Action	Purpose
Step 11	**clear ip bgp** {*ip-address* | *****} **in prefix-filter** **Example:** `Router# clear ip bgp 10.1.1.1 in prefix-filter`	Clears BGP outbound route filters and initiates an inbound soft reset. • A single neighbor or all neighbors can be specified. **Note** The inbound soft refresh must be initiated with the **clear ip bgp** command in order for this feature to function.

Configuring BGP Peering with ISPs

BGP was developed as an interdomain routing protocol and connecting to ISPs is one of the main functions of BGP. Depending on the size of your network and the purpose of your business, there are many different ways to connect to your ISP. Multihoming to one or more ISPs provides redundancy in case an external link to an ISP fails. This section introduces some optional tasks that can be used to connect to a service provider using multihoming techniques. Smaller companies may use just one ISP but require a backup route to the ISP. Larger companies may have access to two ISPs, using one of the connections as a backup, or may need to configure a transit autonomous system.

Perform one of the following optional tasks to connect to one or more ISPs:

- Configuring Multihoming with Two ISPs, page 155
- Multihoming with a Single ISP, page 159
- Configuring Multihoming to Receive the Full Internet Routing Table, page 167

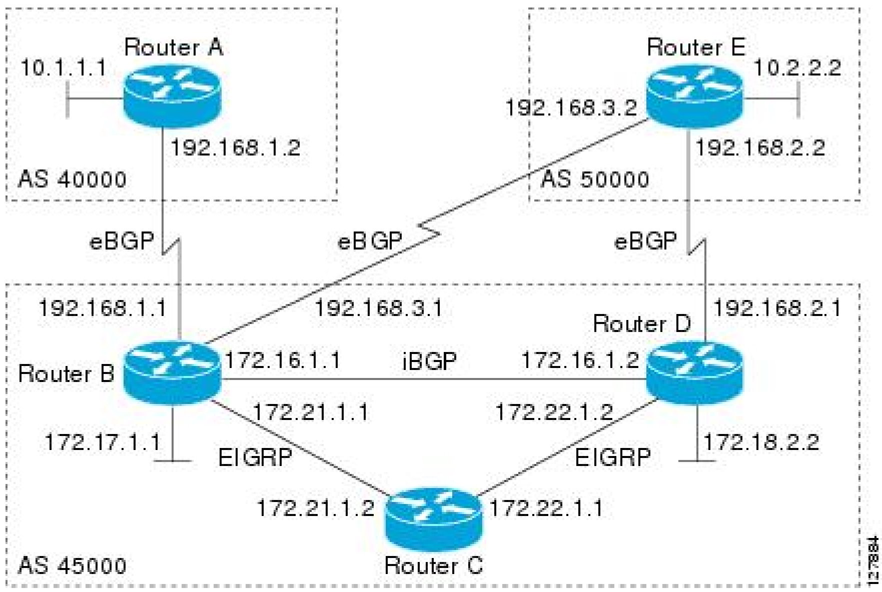
Configuring Multihoming with Two ISPs

Perform this task to configure your network to access two ISPs. where one ISP is the preferred route and the second ISP is a backup route. In the figure below Router B in autonomous system 45000 has BGP peers in two ISPs, autonomous system 40000 and autonomous system 50000. Using this task, Router B will be configured to prefer the route to the BGP peer at Router A in autonomous system 40000.

All routes learned from this neighbor will have an assigned weight. The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network.

Note The weights assigned with the **set weight** route-map configuration command override the weights assigned using the **neighbor weight** command.

Figure 17 Multihoming with Two ISPs



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **network** *network-number* [**mask** *network-mask*]
7. **neighbor** {*ip-address*| *peer-group-name*} **weight** *number*
8. **exit**
9. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
10. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
11. **neighbor** {*ip-address*| *peer-group-name*} **weight** *number*
12. **end**
13. **clear ip bgp** {*****| *ip-address*| *peer-group-name*} [**soft** [**in** | **out**]]
14. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode, and creates a BGP routing process.
Step 4	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.

	Command or Action	Purpose
Step 7	**neighbor** {*ip-address*| *peer-group-name*} **weight** *number* **Example:** `Router(config-router-af)# neighbor 192.168.1.2 weight 150`	Assigns a weight to a BGP peer connection. • In this example, the weight attribute for routes received from the BGP peer 192.168.1.2 is set to 150.
Step 8	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 9	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 10	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 11	**neighbor** {*ip-address*| *peer-group-name*} **weight** *number* **Example:** `Router(config-router-af)# neighbor 192.168.3.2 weight 100`	Assigns a weight to a BGP peer connection. • In this example, the weight attribute for routes received from the BGP peer 192.168.3.2 is set to 100.
Step 12	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 13	**clear ip bgp** {*****| *ip-address*| *peer-group-name*} [**soft** [**in** | **out**]] **Example:** `Router# clear ip bgp *`	(Optional) Clears BGP outbound route filters and initiates an outbound soft reset. A single neighbor or all neighbors can be specified.
Step 14	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp`	Displays the entries in the BGP routing table. • Enter this command at Router B to see the weight attribute for each route to a BGP peer. The route with the highest weight attribute will be the preferred route to network 172.17.1.0. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following example shows the BGP routing table at Router B with the weight attributes assigned to routes. The route through 192.168.3.2 (Router E in the figure above) has the highest weight attribute and will be the preferred route to network 172.17.1.0.

```
BGP table version is 8, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0           100 40000 i
*> 10.2.2.0/24      192.168.3.2              0           150 50000 i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
```

Multihoming with a Single ISP

Perform this task to configure your network to access one of two connections to a single ISP, where one of the connections is the preferred route and the second connection is a backup route. In the figure above Router E in autonomous system 50000 has two BGP peers in a single autonomous system, autonomous system 45000. Using this task, autonomous system 50000 does not learn any routes from autonomous system 45000 and is sending its own routes using BGP. This task is configured at Router E in the figure above and covers three features about multihoming to a single ISP:

- Outbound traffic--Router E will forward default routes and traffic to autonomous system 45000 with Router B as the primary link and Router D as the backup link. Static routes are configured to both Router B and Router D with a lower distance configured for the link to Router B.
- Inbound traffic--Inbound traffic from autonomous system 45000 is configured to be sent from Router B unless the link fails when the backup route is to send traffic from Router D. To achieve this, outbound filters are set using the MED metric.
- Prevention of transit traffic--A route map is configured at Router E in autonomous system 50000 to block all incoming BGP routing updates to prevent autonomous system 50000 from receiving transit traffic from the ISP in autonomous system 45000.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **network** *network-number* [**mask** *network-mask*][**route-map** *route-map-name*]
7. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
8. Repeat Step 7 to apply another route map to the neighbor specified in Step 7.
9. **exit**
10. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
11. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
12. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
13. Repeat Step 10 to apply another route map to the neighbor specified in Step 10.
14. **exit**
15. **exit**
16. **ip route** *prefix mask* {*ip-address* | *interface-type interface-number*[*ip-address*]} [*distance*] [*name*] [**permanent**| **track** *number*][**tag** *tag*]
17. Repeat Step 14 to establish another static route.
18. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
19. **set metric** *value*
20. **exit**
21. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
22. **set metric** *value*
23. **exit**
24. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
25. **end**
26. **show ip route** [*ip-address*] [*mask*] [**longer-prefixes**]
27. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.2.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • In this example, the BGP peer at Router D is added to the BGP routing table.
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Router(config-router-af)# network 10.2.2.0 mask 255.255.255.0`	Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.

	Command or Action	Purpose

Step 7	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
	
	Example:
	`Router(config-router-af)# neighbor 192.168.2.1 route-map BLOCK in`
	
	Example:
	`Router(config-router-af)# neighbor 192.168.2.1 route-map SETMETRIC1 out`
		Applies a route map to incoming or outgoing routes.
		• In the first example, the route map named BLOCK is applied to inbound routes at Router E.
		• In the second example, the route map named SETMETRIC1 is applied to outbound routes to Router D.
		Note Two examples are shown here because the task example requires both these statements to be configured.

Step 8	Repeat Step 7 to apply another route map to the neighbor specified in Step 7.	--
Step 9	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • In this example, the BGP peer at Router D is added to the BGP routing table.
Step 11	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.

	Command or Action	Purpose

Step 12	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
	
	Example:
	
	`Router(config-router-af)# neighbor 192.168.3.1 route-map BLOCK in`
	
	Example:
	
	`Router(config-router-af)# neighbor 192.168.3.1 route-map SETMETRIC2 out`
		Applies a route map to incoming or outgoing routes.
		
		• In the first example, the route map named BLOCK is applied to inbound routes at Router E.
		• In the second example, the route map named SETMETRIC2 is applied to outbound routes to Router D.
		
		Note Two examples are shown here because the task example requires both these statements to be configured.

Step 13	Repeat Step 10 to apply another route map to the neighbor specified in Step 10.	--
Step 14	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 15	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.

	Command or Action	Purpose

Step 16	**ip route** *prefix mask* {*ip-address* | *interface-type interface-number*[*ip-address*]} [*distance*] [*name*][**permanent**| **track** *number*][**tag** *tag*]
	
	Example:
	`Router(config)# ip route 0.0.0.0 0.0.0.0 192.168.2.1 50`
	
	Example:
	`Router(config)# ip route 0.0.0.0 0.0.0.0 192.168.2.1 50`
	
	Example:
	`and`
	
	Example:
	`Router(config)# ip route 0.0.0.0 0.0.0.0 192.168.3.1 40`
		Establishes a static route.
		• In the first example, a static route to BGP peer 192.168.2.1 is established and given an administrative distance of 50.
		• In the second example, a static route to BGP peer 192.168.3.1 is established and given an administrative distance of 40. The lower administrative distance makes this route via Router B the preferred route.
		Note Two examples are shown here because the task example requires both these statements to be configured.

Step 17	Repeat Step 14 to establish another static route.	--
Step 18	**route-map** *map-name* [**permit**| **deny**] [*sequence-number*] **Example:** `Router(config)# route-map SETMETRIC1 permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named SETMETRIC1 is created.
Step 19	**set metric** *value* **Example:** `Router(config-route-map)# set metric 100`	Sets the MED metric value.
Step 20	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 21	**route-map** *map-name* [**permit**| **deny**] [*sequence-number*] **Example:** `Router(config)# route-map SETMETRIC2 permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named SETMETRIC2 is created.
Step 22	**set metric** *value* **Example:** `Router(config-route-map)# set metric 50`	Sets the MED metric value.
Step 23	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 24	**route-map** *map-name* [**permit**| **deny**] [*sequence-number*] **Example:** `Router(config)# route-map BLOCK deny 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named BLOCK is created to block all incoming routes from autonomous system 45000.
Step 25	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.
Step 26	**show ip route** [*ip-address*] [*mask*] [**longer-prefixes**] **Example:** `Router# show ip route`	(Optional) Displays route information from the routing tables. • Use this command at Router E in the figure above after Router B and Router D have received update information containing the MED metric from Router E. • Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

	Command or Action	Purpose
Step 27	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp 172.17.1.0 255.255.255.0`	(Optional) Displays the entries in the BGP routing table. • Use this command at Router E in the figure above after Router B and Router D have received update information containing the MED metric from Router E. • Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following example shows output from the **show ip route** command entered at Router E after this task has been configured and Router B and Router D have received update information containing the MED metric. Note that the gateway of last resort is set as 192.168.3.1, which is the route to Router B.

```
Router# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 192.168.3.1 to network 0.0.0.0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.2.2.0 is directly connected, Ethernet0/0
C    192.168.2.0/24 is directly connected, Serial3/0
C    192.168.3.0/24 is directly connected, Serial2/0
S*   0.0.0.0/0 [40/0] via 192.168.3.1
```

The following example shows output from the **show ip bgp** command entered at Router E after this task has been configured and Router B and Router D have received routing updates. The route map BLOCK has denied all routes coming in from autonomous system 45000 so the only network shown is the local network.

```
Router# show ip bgp
BGP table version is 2, local router ID is 10.2.2.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.2.0/24      0.0.0.0                  0         32768 i
```

The following example shows output from the **show ip bgp** command entered at Router B after this task has been configured at Router E and Router B has received routing updates. Note the metric of 50 for network 10.2.2.0.

```
Router# show ip bgp
BGP table version is 7, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 40000 i
*> 10.2.2.0/24      192.168.3.2             50             0 50000 i
*> 172.16.1.0/24    0.0.0.0                  0         32768 i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
```

The following example shows output from the **show ip bgp** command entered at Router D after this task has been configured at Router E and Router D has received routing updates. Note the metric of 100 for network 10.2.2.0.

```
Router# show ip bgp
BGP table version is 3, local router ID is 192.168.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.2.0/24      192.168.2.2            100             0 50000 i
*> 172.16.1.0/24    0.0.0.0                  0         32768 i
```

Configuring Multihoming to Receive the Full Internet Routing Table

Perform this task to configure your network to build neighbor relationships with other routers in other autonomous systems while filtering outbound routes. In this task the full Internet routing table will be received from the service providers in the neighboring autonomous systems but only locally originated routes will be advertised to the service providers. This task is configured at Router B in the figure above and uses an access list to permit only locally originated routes and a route map to ensure that only the locally originated routes are advertised outbound to other autonomous systems.

Note Be aware that receiving the full Internet routing table from two ISPs may use all the memory in smaller routers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **network** *network-number* [**mask** *network-mask*]
7. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
8. **exit**
9. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
10. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
11. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**}
12. **exit**
13. **exit**
14. **ip as-path access-list** *access-list-number* {**deny** | **permit**} *as-regular-expression*
15. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
16. **match as-path** *path-list-number*
17. **end**
18. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.

	Command or Action	Purpose
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.1.2 route-map localonly out`	Applies a route map to incoming or outgoing routes. • In this example, the route map named localonly is applied to outbound routes to Router A.
Step 8	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 9	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 10	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 11	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.3.2 route-map localonly out`	Applies a route map to incoming or outgoing routes. • In this example, the route map named localonly is applied to outbound routes to Router E.

	Command or Action	Purpose
Step 12	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 13	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 14	**ip as-path access-list** *access-list-number* {**deny** | **permit**} *as-regular-expression* **Example:** `Router(config)# ip as-path access-list 10 permit ^$`	Defines a BGP-related access list. • In this example, the access list number 10 is defined to permit only locally originated BGP routes.
Step 15	**route-map** *map-name* [**permit**| **deny**][*sequence-number*] **Example:** `Router(config)# route-map localonly permit 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named localonly is created.
Step 16	**match as-path** *path-list-number* **Example:** `Router(config-route-map)# match as-path 10`	Matches a BGP autonomous system path access list. • In this example, the BGP autonomous system path access list created in Step 12 is used for the match clause.
Step 17	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.
Step 18	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp`	Displays the entries in the BGP routing table. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following example shows the BGP routing table for Router B in the figure above after this task has been configured. Note that the routing table contains the information about the networks in the autonomous systems 40000 and 50000.

```
BGP table version is 5, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 40000 i
*> 10.2.2.0/24      192.168.3.2              0             0 50000 i
*> 172.17.1.0/24    0.0.0.0                  0         32768 i
```

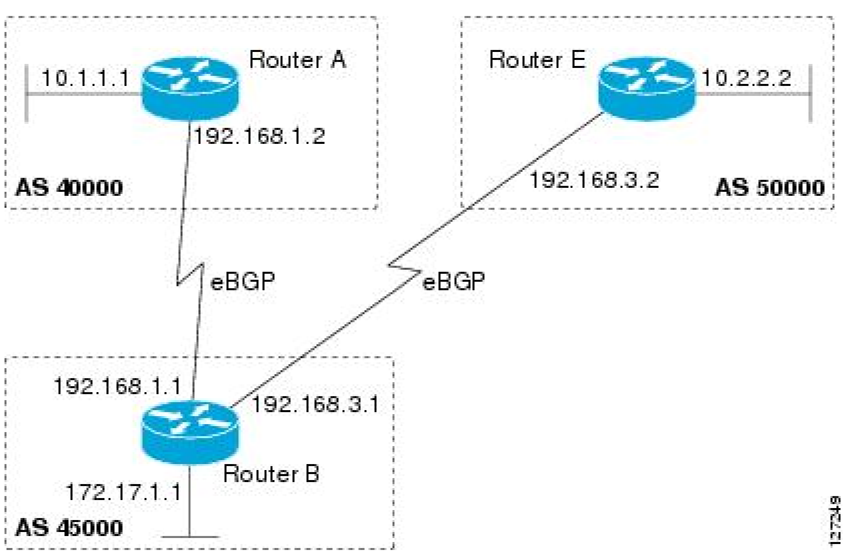
Configuring BGP Policies

The tasks in this section help you configure BGP policies that filter the traffic in your BGP network. The following optional tasks demonstrate some of the various methods by which traffic can be filtered in your BGP network:

Filtering BGP Prefixes with Prefix Lists

Perform this task to use prefix lists to filter BGP route information. The task is configured at Router B in the figure below where both Router A and Router E are set up as BGP peers. A prefix list is configured to permit only routes from the network 10.2.2.0/24 to be outbound. In effect, this will restrict the information that is received from Router E to be forwarded to Router A. Optional steps are included to display the prefix list information and to reset the hit count.

Figure 18 BGP Topology for Configuring BGP Policies Tasks



Note The **neighbor prefix-list** and the **neighbor distribute-list** commands are mutually exclusive for a BGP peer.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. Repeat Step 5 for all BGP peers.
6. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
7. **network** *network-number* [**mask** *network-mask*]
8. **aggregate-address** *address mask* [**as-set**]
9. **neighbor** *ip-address* **prefix-list** *list-name* {**in** | **out**}
10. **exit**
11. **exit**
12. **ip prefix-list** *list-name* [**seq** *seq-number*] {**deny** *network* / *length*| **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*] [**eq** *eq-value*]
13. **end**
14. **show ip prefix-list** [**detail** | **summary**] [*prefix-list-name* [**seq** *seq-number* | *network* / *length* [**longer** | **first-match**]]]
15. **clear ip prefix-list** {*****| *ip-address*| *peer-group-name*} **out**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address of the neighbor in the specified autonomous system to the BGP neighbor table of the local router.
Step 5	Repeat Step 5 for all BGP peers.	--
Step 6	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 7	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.
Step 8	**aggregate-address** *address mask* [**as-set**] **Example:** `Router(config-router-af)# aggregate-address 172.0.0.0 255.0.0.0`	Creates an aggregate entry in a BGP routing table. • A specified route must exist in the BGP table. • Use the **aggregate-address** command with no keywords to create an aggregate entry if any more-specific BGP routes are available that fall in the specified range. **Note** Only partial syntax is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

	Command or Action	Purpose
Step 9	**neighbor** *ip-address* **prefix-list** *list-name* {**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.1.2 prefix-list super172 out`	Distributes BGP neighbor information as specified in a prefix list. • In this example, a prefix list called super172 is set for outgoing routes to Router A.
Step 10	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 11	**exit** **Example:** `Router(config-router) exit`	Exits router configuration mode and enters global configuration mode.
Step 12	**ip prefix-list** *list-name* [**seq** *seq-number*] {**deny** *network* / *length*| **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*] [**eq** *eq-value*] **Example:** `Router(config)# ip prefix-list super172 permit 172.0.0.0/8`	Defines a BGP-related prefix list and enters access list configuration mode. • In this example, the prefix list called super172 is defined to permit only route 172.0.0.0/8 to be forwarded. • All other routes will be denied because there is an implicit deny at the end of all prefix lists.
Step 13	**end** **Example:** `Router(config-access-list)# end`	Exits access list configuration mode and enters privileged EXEC mode.
Step 14	**show ip prefix-list** [**detail** | **summary**] [*prefix-list-name* [**seq** *seq-number* | *network* / *length* [**longer** | **first-match**]]] **Example:** `Router# show ip prefix-list detail super172`	Displays information about prefix lists. • In this example, details of the prefix list named super172 will be displayed, including the hit count. Hit count is the number of times the entry has matched a route.

	Command or Action	Purpose
Step 15	**clear ip prefix-list** {*****| *ip-address*| *peer-group-name*} **out** **Example:** `Router# clear ip prefix-list super172 out`	Resets the hit count of the prefix list entries. • In this example, the hit count for the prefix list called super172 will be reset.

Examples

The following output from the **show ip prefix-list** command shows details of the prefix list named super172, including the hit count. The **clear ip prefix-list**command is entered to reset the hit count and the **show ip prefix-list** command is entered again to show the hit count reset to 0.

```
Router# show ip prefix-list detail super172
ip prefix-list super172:
   count: 1, range entries: 0, sequences: 5 - 5, refcount: 4
   seq 5 permit 172.0.0.0/8 (hit count: 1, refcount: 1)
Router# clear ip prefix-list super172
Router# show ip prefix-list detail super172
ip prefix-list super172:
   count: 1, range entries: 0, sequences: 5 - 5, refcount: 4
   seq 5 permit 172.0.0.0/8 (hit count: 0, refcount: 1)
```

Filtering BGP Prefixes with AS-path Filters

Perform this task to filter BGP prefixes using AS-path filters with an access list based on the value of the AS-path attribute to filter route information. An AS-path access list is configured at Router B in the figure above. The first line of the access list denies all matches to the AS-path 50000 and the second line allows all other paths. The router uses the **neighbor filter-list** command to specify the AS-path access list as an outbound filter. After the filtering is enabled, traffic can be received from both Router A and Router C but updates originating from autonomous system 50000 (Router C) are not forwarded by Router B to Router A. If any updates from Router C originated from another autonomous system, they would be forwarded because they would contain both autonomous system 50000 plus another autonomous system number, and that would not match the AS-path access list.

Note In Cisco IOS Releases 12.0(22)S, 12.2(15)T, 12.2(18)S, and later releases, the maximum number of autonomous system access lists that can be configured with the **ip as-path access-list** command is increased from 199 to 500.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. Repeat Step 4 for all BGP peers.
6. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
7. **network** *network-number* [**mask** *network-mask*]
8. **neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number*{**in** | **out**}
9. **exit**
10. **exit**
11. **ip as-path access-list** *access-list-number* {**deny** | **permit**} *as-regular-expression*
12. Repeat Step 11 for all entries required in the AS-path access list.
13. **end**
14. **show ip bgp regexp** *as-regular-expression*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system BGP neighbor table of the local router.

	Command or Action	Purpose
Step 5	Repeat Step 4 for all BGP peers.	--
Step 6	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 7	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates. **Note** Only partial syntax is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.1.2 filter-list 100 out`	Distributes BGP neighbor information as specified in a prefix list. • In this example, an access list number 100 is set for outgoing routes to Router A.
Step 9	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and returns to router configuration mode.
Step 10	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and returns to global configuration mode.

	Command or Action	Purpose

Step 11	**ip as-path access-list** *access-list-number* {**deny** | **permit**} *as-regular-expression*
	
	Example:
	`Router(config)# ip as-path access-list 100 deny ^50000$`
	
	Example:
	`Router(config)# ip as-path access-list 100 permit .*`
		Defines a BGP-related access list and enters access list configuration mode.
		• In the first example, access list number 100 is defined to deny any AS-path that starts and ends with 50000.
		• In the second example, all routes that do not match the criteria in the first example of the AS-path access list will be permitted. The period and asterisk symbols imply that all characters in the AS-path will match, so Router B will forward those updates to Router A.
		Note Two examples are shown here because the task example requires both these statements to be configured.

Step 12	Repeat Step 11 for all entries required in the AS-path access list.	--
Step 13	**end** **Example:** `Router(config-access-list)# end`	Exits access list configuration mode and returns to privileged EXEC mode.
Step 14	**show ip bgp regexp** *as-regular-expression* **Example:** `Router# show ip bgp regexp ^50000$`	Displays routes that match the regular expression. • To verify the regular expression, you can use this command. • In this example, all paths that match the expression "starts and ends with 50000" will be displayed.

Examples

The following output from the **show ip bgp regexp** command shows the autonomous system paths that match the regular expression--start and end with AS-path 50000:

```
Router# show ip bgp regexp ^50000$
BGP table version is 9, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.2.0/24      192.168.3.2              0           150 50000 i
```

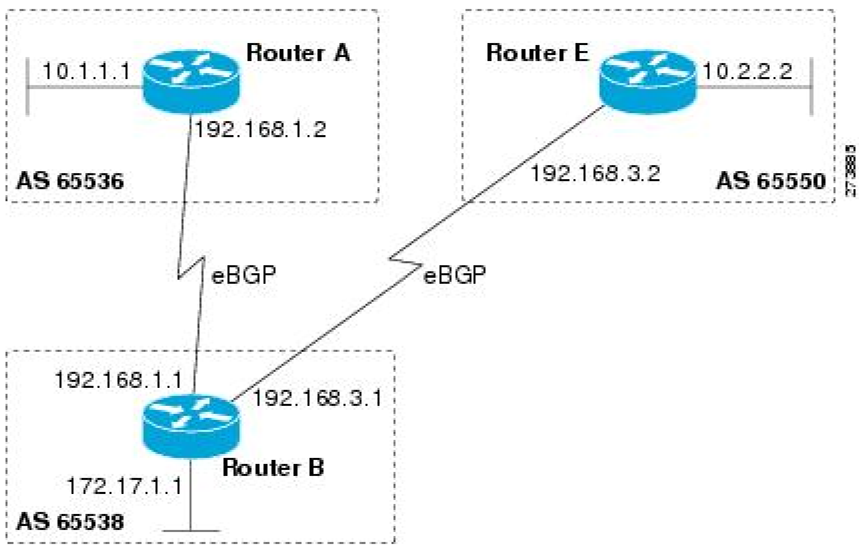
Filtering BGP Prefixes with AS-path Filters Using 4-Byte Autonomous System Numbers

In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)SXI1, and later releases, BGP support for 4-octet (4-byte) autonomous system numbers was introduced. The 4-byte autonomous system numbers in this task are formatted in the default asplain (decimal value) format, for example, Router B is in autonomous system number 65538 in the figure below. For more details about the introduction of 4-byte autonomous system numbers, see BGP Autonomous System Number Formats, page 130.

Perform this task to filter BGP prefixes with AS-path filters using 4-byte autonomous system numbers with an access list based on the value of the AS-path attribute to filter route information. An AS-path access list is configured at Router B in the figure below. The first line of the access list denies all matches to the AS-path 65550 and the second line allows all other paths. The router uses the **neighbor filter-list** command to specify the AS-path access list as an outbound filter. After the filtering is enabled, traffic can be received from both Router A and Router E but updates originating from autonomous system 65550 (Router E) are not forwarded by Router B to Router A. If any updates from Router E originated from another autonomous system, they would be forwarded because they would contain both autonomous system 65550 plus another autonomous system number, and that would not match the AS-path access list.

Note In Cisco IOS Releases 12.0(22)S, 12.2(15)T, 12.2(18)S, and later releases, the maximum number of autonomous system access lists that can be configured with the **ip as-path access-list** command is increased from 199 to 500.

Figure 19 BGP Topology for Filtering BGP Prefixes with AS-path Filters Using 4-Byte Autonomous System Numbers



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. Repeat Step 4 for all BGP peers.
6. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
7. **network** *network-number* [**mask** *network-mask*]
8. **neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number*{**in** | **out**}
9. **exit**
10. **exit**
11. **ip as-path access-list** *access-list-number* {**deny** | **permit**} *as-regular-expression*
12. Repeat Step 11 for all entries required in the AS-path access list.
13. **end**
14. **show ip bgp regexp** *as-regular-expression*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 65538`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.1.2 remote-as 65536`	Adds the IP address or peer group name of the neighbor in the specified autonomous system BGP neighbor table of the local router. • In this example, the IP address for the neighbor at Router A is added.

	Command or Action	Purpose
Step 5	Repeat Step 4 for all BGP peers.	--
Step 6	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 7	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates. **Note** Only partial syntax is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.1.2 filter-list 99 out`	Distributes BGP neighbor information as specified in a prefix list. • In this example, an access list number 99 is set for outgoing routes to Router A.
Step 9	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and returns to router configuration mode.
Step 10	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and returns to global configuration mode.

	Command or Action	Purpose

Step 11	**ip as-path access-list** *access-list-number* {**deny** | **permit**} *as-regular-expression*
	
	Example:
	
	`Router(config)# ip as-path access-list 99 deny ^65550$`
	
	Example:
	
	`and`
	
	Example:
	
	`Router(config)# ip as-path access-list 99 permit .*`
		Defines a BGP-related access list and enters access list configuration mode.
		
		• In the first example, access list number 99 is defined to deny any AS-path that starts and ends with 65550.
		• In the second example, all routes that do not match the criteria in the first example of the AS-path access list will be permitted. The period and asterisk symbols imply that all characters in the AS-path will match, so Router B will forward those updates to Router A.
		
		Note Two examples are shown here because the task example requires both these statements to be configured.

Step 12	Repeat Step 11 for all entries required in the AS-path access list.	--
Step 13	**end** **Example:** `Router(config-access-list)# end`	Exits access list configuration mode and returns to privileged EXEC mode.
Step 14	**show ip bgp regexp** *as-regular-expression* **Example:** `Router# show ip bgp regexp ^65550$`	Displays routes that match the regular expression. • To verify the regular expression, you can use this command. • In this example, all paths that match the expression "starts and ends with 65550" will be displayed.

Examples

The following output from the **show ip bgp regexp** command shows the autonomous system paths that match the regular expression--start and end with AS-path 65550:

```
RouterB# show ip bgp regexp ^65550$
BGP table version is 4, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.2.0/24      192.168.3.2              0             0 65550 i
```

Filtering Traffic Using Community Lists

Perform this task to filter traffic by creating BGP community lists and then reference them within a route map to control incoming routes. BGP communities provide a method of filtering inbound or outbound routes for large, complex networks. Instead of compiling long access or prefix lists of individual peers, BGP allows grouping of peers with identical routing policies even though they reside in different autonomous systems or networks.

In this task, Router B in the figure above is configured with several route maps and community lists to control incoming routes.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **neighbor** {*ip-address* | *peer-group-name*} **route-map** *route-map-name*{**in** | **out**}
7. exit
8. **exit**
9. **route-map** *map-name* [**permit** | **deny**] [*sequence-number*]
10. **match community** {*standard-list-number* | *expanded-list-number* | *community-list-name* [**exact**]}
11. **set weight** *weight*
12. **exit**
13. **route-map** *map-name* [**permit** | **deny**] [*sequence-number*]
14. **match community** {*standard-list-number* | *expanded-list-number* | *community-list-name* [**exact**]}
15. **set community** *community-number*
16. **exit**
17. **ip community-list** {*standard-list-number*| **standard** *list-name* {**deny** | **permit**} [*community-number*] [*AA*:*NN*] [**internet**] [**local-AS**] [**no-advertise**] [**no-export**]} | {*expanded-list-number* | **expanded** *list-name* {**deny** | **permit**} *regular-expression*}
18. Repeat Step 15 to create all the required community lists.
19. **exit**
20. **show ip community-list** [*standard-list-number* | *expanded-list-number* | *community-list-name*][**exact-match**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system BGP neighbor table of the local router.
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *route-map-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 192.168.3.2 route-map 2000 in`	Applies a route map to inbound or outbound routes. • In this example, the route map called 2000 is applied to inbound routes from the BGP peer at 192.168.3.2.
Step 7	exit **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 8	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 9	**route-map** *map-name* [**permit** | **deny**] [*sequence-number*] **Example:** `Router(config)# route-map 2000 permit 10`	Creates a route map and enters route map configuration mode. • In this example, the route map called 2000 is defined.
Step 10	**match community** {*standard-list-number* | *expanded-list-number* | *community-list-name* [**exact**]} **Example:** `Router(config-route-map)# match community 1`	Matches a BGP community list. • In this example, the community attribute is matched to community list 1.
Step 11	**set weight** *weight* **Example:** `Router(config-route-map)# set weight 30`	Specifies the BGP weight for the routing table. • In this example, any route that matches community list 1 will have the BGP weight set to 30.
Step 12	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 13	**route-map** *map-name* [**permit** | **deny**] [*sequence-number*] **Example:** `Router(config)# route-map 3000 permit 10`	Creates a route map and enters route map configuration mode. • In this example, the route map called 3000 is defined.
Step 14	**match community** {*standard-list-number* | *expanded-list-number* | *community-list-name* [**exact**]} **Example:** `Router(config-route-map)# match community 2`	Matches a BGP community list. • In this example, the community attribute is matched to community list 2.

	Command or Action	Purpose
Step 15	**set community** *community-number* **Example:** `Router(config-route-map)# set community 99`	Sets the BGP communities attribute. • In this example, any route that matches community list 2 will have the BGP community attribute set to 99.
Step 16	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.

Step 17	**ip community-list** {*standard-list-number*| **standard** *list-name* {**deny** | **permit**} [*community-number*] [*AA*:*NN*] [**internet**] [**local-AS**] [**no-advertise**] [**no-export**]} | {*expanded-list-number* | **expanded** *list-name* {**deny** | **permit**} *regular-expression*}
	
	Example:
	
	`Router(config)# ip community-list 1 permit 100`
	
	Example:
	
	`and`
	
	Example:
	
	`Router(config)# ip community-list 2 permit internet`
		Creates a community list for BGP and controls access to it.
		
		• In the first example, community list 1 permits routes with a community attribute of 100. Router C routes all have community attribute of 100 so their weight will be set to 30.
		• In the second example, community list 2 effectively permits all routes by using the **internet**keyword. Any routes that did not match community list 1 are checked against community list 2. All routes are permitted but no changes are made to the route attributes.
		
		Note Two examples are shown here because the task example requires both these statements to be configured.

Step 18	Repeat Step 15 to create all the required community lists.	--
Step 19	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 20	**show ip community-list** [*standard-list-number* | *expanded-list-number* | *community-list-name*] [**exact-match**] **Example:** `Router# show ip community-list 1`	Displays configured BGP community list entries.

Examples

The following sample output verifies that community list 1 has been created, with the output showing that community list 1 permits routes with a community attribute of 100:

```
Router# show ip community-list 1
Community standard list 1
     permit 100
```

The following sample output verifies that community list 2 has been created, with the output showing that community list 2 effectively permits all routes by using the **internet**keyword:

```
Router# show ip community-list 2
Community standard list 2
     permit internet
```

Filtering Traffic Using Extended Community Lists

Perform this task to filter traffic by creating an extended BGP community list to control outbound routes. BGP communities provide a method of filtering inbound or outbound routes for large, complex networks. Instead of compiling long access or prefix lists of individual peers, BGP allows grouping of peers with identical routing policies even though they reside in different autonomous systems or networks.

In this task, Router B in the figure above is configured with an extended named community list to specify that the BGP peer at 192.168.1.2 is not sent advertisements about any path through or from autonomous system 50000. The IP extended community-list configuration mode is used and the ability to resequence entries is shown.

Note A sequence number is applied to all extended community list entries by default regardless of the configuration mode. Explicit sequencing and resequencing of extended community list entries can be configured only in IP extended community-list configuration mode and not in global configuration mode.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip extcommunity-list** {*expanded-list-number*| **expanded** *list-name*| *standard-list-number* | **standard** *list-name*}
4. [*sequence-number*] {**deny**[*regular-expression*] | **exit** | **permit**[*regular-expression*]}
5. Repeat Step 4 for all the required permit or deny entries in the extended community list.
6. **resequence** [*starting-sequence*][*sequence-increment*]
7. **exit**
8. **router bgp** *autonomous-system-number*
9. **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
10. Repeat Step 10 for all the required BGP peers.
11. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
12. **network** *network-number* [**mask** *network-mask*]
13. **end**
14. **show ip extcommunity-list** [*list-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip extcommunity-list** {*expanded-list-number*| **expanded** *list-name*| *standard-list-number* | **standard** *list-name*} **Example:** `Router(config)# ip extcommunity-list expanded DENY50000`	Enters IP extended community-list configuration mode to create or configure an extended community list. • In this example, the expanded community list DENY50000 is created.

	Command or Action	Purpose

Step 4	[*sequence-number*] {**deny**[*regular-expression*] | **exit** | **permit**[*regular-expression*]}
	
	Example:
	`Router(config-extcomm-list)# 10 deny _50000_`
	
	Example:
	`and`
	
	Example:
	`Router(config-extcomm-list)# 20 deny ^50000 .*`
		Configures an expanded community list entry.
		• In the first example, an expanded community list entry with the sequence number 10 is configured to deny advertisements about paths from autonomous system 50000.
		• In the second example, an expanded community list entry with the sequence number 20 is configured to deny advertisements about paths through autonomous system 50000.
		Note Two examples are shown here because the task example requires both these statements to be configured.
		Note Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Step 5	Repeat Step 4 for all the required permit or deny entries in the extended community list.	--
Step 6	**resequence** [*starting-sequence*][*sequence-increment*] **Example:** `Router(config-extcomm-list)# resequence 50 100`	Resequences expanded community list entries. • In this example, the sequence number of the first expanded community list entry is set to 50 and subsequent entries are set to increment by 100. The second expanded community list entry is therefore set to 150. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 7	**exit** **Example:** `Router(config-extcomm-list)# exit`	Exits expanded community-list configuration mode and enters global configuration mode.
Step 8	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.

	Command or Action	Purpose
Step 9	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system BGP neighbor table of the local router.
Step 10	Repeat Step 10 for all the required BGP peers.	--
Step 11	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. **Note** The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 12	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this autonomous system and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 13	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.
Step 14	**show ip extcommunity-list** [*list-name*] **Example:** `Router# show ip extcommunity-list DENY50000`	Displays configured BGP expanded community list entries.

Examples

The following sample output verifies that the BGP expanded community list DENY50000 has been created, with the output showing that the entries to deny advertisements about autonomous system 50000 have been resequenced from 10 and 20 to 50 and 150:

```
Router# show ip extcommunity-list DENY50000
Expanded extended community-list DENY50000
     50 deny _50000_
     150 deny ^50000 .*
```

Filtering Traffic Using a BGP Route Map Policy List

Perform this task to create a BGP policy list and then reference it within a route map.

A policy list is like a route map that contains only match clauses. With policy lists there are no changes to match clause semantics and route map functions. The match clauses are configured in policy lists with permit and deny statements and the route map evaluates and processes each match clause to permit or deny routes based on the configuration. AND and OR semantics in the route map function the same way for policy lists as they do for match clauses.

Policy lists simplify the configuration of BGP routing policy in medium-size and large networks. The network operator can reference preconfigured policy lists with groups of match clauses in route maps and easily apply general changes to BGP routing policy. The network operator no longer needs to manually reconfigure each recurring group of match clauses that occur in multiple route map entries.

Perform this task to create a BGP policy list to filter traffic that matches the autonomous system path and MED of a router and then create a route map to reference the policy list.

BGP routing must be configured in your network and BGP neighbors must be established.

Note

- BGP route map policy lists do not support the configuration of IP version 6 (IPv6) match clauses in policy lists.
- Policy lists are not supported in versions of Cisco IOS software prior to Cisco IOS Releases 12.0(22)S and 12.2(15)T. Reloading a router that is running an older version of Cisco IOS software may cause some routing policy configurations to be lost.
- Policy lists support only match clauses and do not support set clauses. However, policy lists can coexist, within the same route map entry, with match and set clauses that are configured separately from the policy lists.
- Policy lists are supported only by BGP. They are not supported by other IP routing protocols. This limitation does not interfere with normal operations of a route map, including redistribution, because policy list functions operate transparently within BGP and are not visible to other IP routing protocols.
- Policy lists support only match clauses and do not support set clauses. However, policy lists can coexist, within the same route map entry, with match and set clauses that are configured separately from the policy lists. The first route map example configures AND semantics, and the second route map configuration example configures semantics. Both examples in this section show sample route map configurations that reference policy lists and separate match and set clauses in the same configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip policy-list** *policy-list-name* {**permit** | **deny**}
4. **match as-path** *as-number*
5. **match metric** *metric*
6. **exit**
7. **route-map** *map-name* [**permit** | **deny**] [*sequence-number*]
8. **match ip address** {*access-list-number* | *access-list-name*} [... *access-list-number* | ... *access-list-name*]
9. **match policy-list** *policy-list-name*
10. **set community** *community-number* [**additive**] [*well-known-community*] | **none**}
11. **set local-preference** *preference-value*
12. **end**
13. **show ip policy-list** [*policy-list-name*]
14. **show route-map** [*route-map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip policy-list** *policy-list-name* {**permit** | **deny**} **Example:** `Router(config)# ip policy-list POLICY-LIST-NAME-1 permit`	Enters policy list configuration mode and creates a BGP policy list that will permit routes that are allowed by the match clauses that follow.
Step 4	**match as-path** *as-number* **Example:** `Router(config-policy-list)# match as-path 500`	Creates a match clause to permit routes from the specified autonomous system path.

	Command or Action	Purpose
Step 5	**match metric** *metric* **Example:** `Router(config-policy-list)# match metric 10`	Creates a match clause to permit routes with the specified metric.
Step 6	**exit** **Example:** `Router(config-policy-list)# exit`	Exits policy list configuration mode and enters global configuration mode.
Step 7	**route-map** *map-name* [**permit** | **deny**] [*sequence-number*] **Example:** `Router(config)# route-map MAP-NAME-1 permit 10`	Creates a route map and enters route map configuration mode.
Step 8	**match ip address** {*access-list-number* | *access-list-name*} [... *access-list-number* | ... *access-list-name*] **Example:** `Router(config-route-map)# match ip address 1`	Creates a match clause to permit routes that match the specified *access-list-number*or *access-list-name*argument.
Step 9	**match policy-list** *policy-list-name* **Example:** `Router(config-route-map)# match policy-list POLICY-LIST-NAME-1`	Creates a clause that will match the specified policy list. • All match clauses within the policy list will be evaluated and processed. Multiple policy lists can referenced with this command. • This command also supports AND or OR semantics like a standard match clause.

Step 10	**set community** *community-number* [**additive**] [*well-known-community*] | **none**}
	
	Example:
	
	`Router(config-route-map)#`
	` set community 10:`
	`1`
		Creates a clause to set or remove the specified community.

	Command or Action	Purpose
Step 11	**set local-preference** *preference-value* **Example:** `Router(config-route-map)# set local-preference 140`	Creates a clause to set the specified local preference value.
Step 12	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.
Step 13	**show ip policy-list** [*policy-list-name*] **Example:** `Router# show ip policy-list POLICY-LIST-NAME-1`	Display information about configured policy lists and policy list entries.
Step 14	**show route-map** [*route-map-name*] **Example:** `Router# show route-map`	Displays locally configured route maps and route map entries.

Examples

The following sample output verifies that a policy list has been created, with the output displaying the policy list name and configured match clauses:

```
Router# show ip policy-list
 POLICY-LIST-NAME-1
policy-list POLICY-LIST-NAME-1 permit
  Match clauses:
    metric 20
    as-path (as-path filter): 1
```

Note A policy list name can be specified when the **show ip policy-list** command is entered. This option can be useful for filtering the output of this command and verifying a single policy list.

The following sample output from the **show route-map** command verifies that a route map has been created and a policy list is referenced. The output of this command displays the route map name and policy lists that are referenced by the configured route maps.

```
Router# show route-map
route-map ROUTE-MAP-NAME-1, deny, sequence 10
  Match clauses:
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
route-map ROUTE-MAP-NAME-1, permit, sequence 10
```

```
Match clauses:
  IP Policy lists:
      POLICY-LIST-NAME-1
Set clauses:
Policy routing matches: 0 packets, 0 bytes
```

Filtering Traffic Using Continue Clauses in a BGP Route Map

Perform this task to filter traffic using continue clauses in a BGP route map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
6. **neighbor** {*ip-address*| *peer-group-name*} **route-map** *map-name*{**in** | **out**}
7. **exit**
8. **exit**
9. **route-map** *map-name* {**permit** | **deny**} [*sequence-number*]
10. **match ip address** {*access-list-number* | *access-list-name*} [... *access-list-number* | ... *access-list-name*]
11. **set community** *community-number* [**additive**] [*well-known-community*] | **none**}
12. **continue** [*sequence-number*]
13. **end**
14. **show route-map** [*map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 50000`	Enters router configuration mode, and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 10.0.0.1 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 5	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**neighbor** {*ip-address*| *peer-group-name*} **route-map** *map-name*{**in** | **out**} **Example:** `Router(config-router-af)# neighbor 10.0.0.1 route-map ROUTE-MAP-NAME in`	Applies the inbound route map to routes received from the specified neighbor, or applies an outbound route map to routes advertised to the specified neighbor.
Step 7	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 8	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 9	**route-map** *map-name* {**permit** | **deny**} [*sequence-number*] **Example:** `Router(config)# route-map ROUTE-MAP-NAME permit 10`	Enters route-map configuration mode to create or configure a route map.

	Command or Action	Purpose
Step 10	**match ip address** {*access-list-number* | *access-list-name*} [... *access-list-number* | ... *access-list-name*] **Example:** `Router(config-route-map)# match ip address 1`	Configures a **match** command that specifies the conditions under which policy routing and route filtering occur. • Multiple **match** commands can be configured. If a **match** command is configured, a match must occur in order for the continue statement to be executed. If a **match** command is not configured, set and continue clauses will be executed. **Note** The **match** and **set** commands used in this task are examples that are used to help describe the operation of the **continue** command. For a list of specific **match** and **set** commands, see the **continue** command in the *Cisco IOS IP Routing: BGP Command Reference*.
Step 11	**set community** *community-number* [**additive**] [*well-known-community*] | **none**} **Example:** `Router(config-route-map)# set community 10: 1`	Configures a **set** command that specifies the routing action to perform if the criteria enforced by the **match** commands are met. • Multiple **set** commands can be configured. • In this example, a clause is created to set the specified community.
Step 12	**continue** [*sequence-number*] **Example:** `Router(config-route-map)# continue`	Configures a route map to continue to evaluate and execute match statements after a successful match occurs. • If a sequence number is configured, the continue clause will go to the route map with the specified sequence number. • If no sequence number is specified, the continue clause will go to the route map with the next sequence number. This behavior is called an "implied continue." **Note** Continue clauses in outbound route maps are supported only in Cisco IOS Release 12.0(31)S, 12.2(33)SB, 12.2(33)SRB, 12.2(33)SXI, 12.4(4)T, and later releases.
Step 13	**end** **Example:** `Router(config-route-map)# end`	Exits route-map configuration mode and enters privileged EXEC mode.
Step 14	**show route-map** [*map-name*] **Example:** `Router# show route-map`	(Optional) Displays locally configured route maps. The name of the route map can be specified in the syntax of this command to filter the output.

Examples

The following sample output shows how to verify the configuration of continue clauses using the **show route-map** command. The output displays configured route maps including the match, set, and continue clauses.

```
Router# show route-map
route-map MARKETING, permit, sequence 10
  Match clauses:
    ip address (access-lists): 1
    metric 10
  Continue: sequence 40
  Set clauses:
    as-path prepend 10
  Policy routing matches: 0 packets, 0 bytes
route-map MARKETING, permit, sequence 20
  Match clauses:
    ip address (access-lists): 2
    metric 20
  Set clauses:
    as-path prepend 10 10
  Policy routing matches: 0 packets, 0 bytes
route-map MARKETING, permit, sequence 30
  Match clauses:
  Continue: to next entry 40
  Set clauses:
    as-path prepend 10 10 10
  Policy routing matches: 0 packets, 0 bytes
route-map MARKETING, permit, sequence 40
  Match clauses:
    community (community-list filter): 10:1
  Set clauses:
    local-preference 104
  Policy routing matches: 0 packets, 0 bytes
route-map MKTG-POLICY-MAP, permit, sequence 10
  Match clauses:
  Set clauses:
    community 655370
  Policy routing matches: 0 packets, 0 bytes
```

Configuration Examples for Connecting to a Service Provider Using External BGP

Influencing Inbound Path Selection Examples

The following example shows how you can use route maps to modify incoming data from a neighbor. Any route received from 10.222.1.1 that matches the filter parameters set in autonomous system access list 200 will have its weight set to 200 and its local preference set to 250, and it will be accepted.

```
router bgp 100
!
 neighbor 10.222.1.1 route-map FIX-WEIGHT in
 neighbor 10.222.1.1 remote-as 1
!
ip as-path access-list 200 permit ^690$
ip as-path access-list 200 permit ^1800
!
route-map FIX-WEIGHT permit 10
 match as-path 200
 set local-preference 250
 set weight 200
```

In the following example, the route map named finance marks all paths originating from autonomous system 690 with an MED metric attribute of 127. The second permit clause is required so that routes not matching autonomous system path list 1 will still be sent to neighbor 10.1.1.1.

```
router bgp 65000
 neighbor 10.1.1.1 route-map finance out
!
ip as-path access-list 1 permit ^690_
ip as-path access-list 2 permit .*
!
route-map finance permit 10
 match as-path 1
 set metric 127
!
route-map finance permit 20
 match as-path 2
```

Inbound route maps could perform prefix-based matching and set various parameters of the update. Inbound prefix matching is available in addition to autonomous system path and community list matching. The following example shows how the **set local-preference** route map configuration command sets the local preference of the inbound prefix 172.20.0.0/16 to 120:

```
!
router bgp 65100
 network 10.108.0.0
 neighbor 10.108.1.1 remote-as 65200
 neighbor 10.108.1.1 route-map set-local-pref in
!
route-map set-local-pref permit 10
 match ip address 2
 set local preference 120
!
route-map set-local-pref permit 20
!
access-list 2 permit 172.20.0.0 0.0.255.255
access-list 2 deny any
```

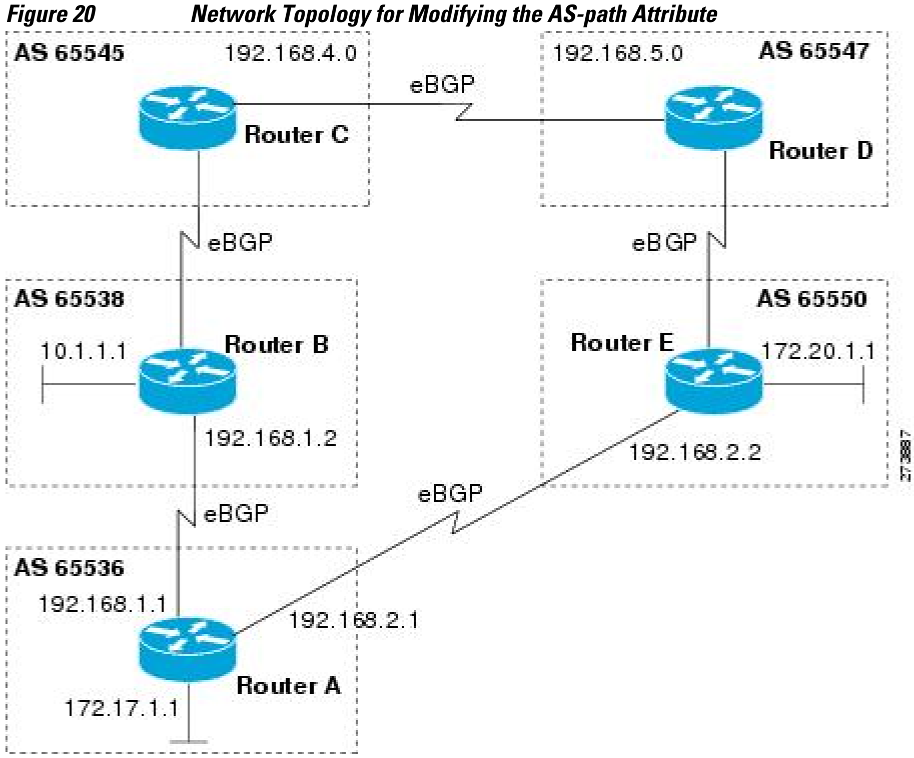
Influencing Inbound Path Selection by Modifying the AS-path Attribute Using 4-Byte AS Numbers Example

This example shows how to configure BGP to influence the inbound path selection for traffic destined for the 172.17.1.0 network by modifying the AS-path attribute. In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SXI1, and later releases, BGP support for 4-octet (4-byte) autonomous system numbers was introduced. The 4-byte autonomous system numbers in this example are formatted in the default asplain (decimal value) format; for example, Router B is in autonomous system number 65538 in the figure below. For more details about the introduction of 4-byte autonomous system numbers, see BGP Autonomous System Number Formats, page 130.

One of the methods that BGP can use to influence the choice of paths in another autonomous system is to modify the AS-path attribute. For example, in the figure below, Router A advertises its own network, 172.17.1.0, to its BGP peers in autonomous system 65538 and autonomous system 65550. When the routing information is propagated to autonomous system 65545, the routers in autonomous system 65545 have network reachability information about network 172.17.1.0 from two different routes. The first route is from autonomous system 65538 with an AS-path consisting of 65538, 65536. The second route is through autonomous system 65547 with an AS-path of 65547, 65550, 65536. If all other BGP attribute values are the same, Router C in autonomous system 65545 would choose the route through autonomous system 65538 for traffic destined for network 172.17.1.0 because it is the shortest route in terms of autonomous systems traversed.

Autonomous system 65536 now receives all traffic from autonomous system 65545 for the 172.17.1.0 network through Router B in autonomous system 65538. If, however, the link between autonomous system 65538 and autonomous system 65536 is a really slow and congested link, the **set as-path prepend**command can be used at Router A to influence inbound path selection for the 172.17.1.0 network by making the route through autonomous system 65538 appear to be longer than the path through autonomous system 65550. The configuration is done at Router A in the figure below by applying a route map to the outbound BGP updates to Router B. Using the **set as-path prepend**command, all the outbound BGP updates from Router A to Router B will have their AS-path attribute modified to add the local autonomous system number 65536 twice. After the configuration, autonomous system 65545 receives updates about the 172.17.1.0 network through autonomous system 65538. The new AS-path is 65538, 65536, 65536, 65536, which is now longer than the AS-path from autonomous system 65547 (unchanged at a value of 65547, 65550, 65536). Networking devices in autonomous system 65545 will now prefer the route through autonomous system 65547 to forward packets with a destination address in the 172.17.1.0 network.

Figure 20 Network Topology for Modifying the AS-path Attribute



The configuration for this example is performed at Router A in the figure above.

```
router bgp 65536
 address-family ipv4 unicast
  network 172.17.1.0 mask 255.255.255.0
  neighbor 192.168.1.2 remote-as 65538
  neighbor 192.168.1.2 activate
  neighbor 192.168.1.2 route-map PREPEND out
  exit-address-family
 exit
route-map PREPEND permit 10
set as-path prepend 65536 65536
```

Influencing Outbound Path Selection Examples

The following example creates an outbound route filter and configures Router-A (10.1.1.1) to advertise the filter to Router-B (172.16.1.2). An IP prefix list named FILTER is created to specify the 192.168.1.0/24 subnet for outbound route filtering. The ORF send capability is configured on Router-A so that Router-A can advertise the outbound route filter to Router-B.

Router-A Configuration (Sender)

```
ip prefix-list FILTER seq 10 permit 192.168.1.0/24
!
router bgp 65100
 address-family ipv4 unicast
 neighbor 172.16.1.2 remote-as 65200
 neighbor 172.16.1.2 ebgp-multihop
 neighbor 172.16.1.2 capability orf prefix-list send
 neighbor 172.16.1.2 prefix-list FILTER in
 end
```

Router-B Configuration (Receiver)

The following example configures Router-B to advertise the ORF receive capability to Router-A. Router-B will install the outbound route filter, defined in the FILTER prefix list, after ORF capabilities have been exchanged. An inbound soft reset is initiated on Router-B at the end of this configuration to activate the outbound route filter.

```
router bgp 65200
 address-family ipv4 unicast
 neighbor 10.1.1.1 remote-as 65100
 neighbor 10.1.1.1 ebgp-multihop 255
 neighbor 10.1.1.1 capability orf prefix-list receive
 end
clear ip bgp 10.1.1.1 in prefix-filter
```

The following example shows how the route map named set-as-path is applied to outbound updates to the neighbor 10.69.232.70. The route map will prepend the autonomous system path "65100 65100" to routes that pass access list 1. The second part of the route map is to permit the advertisement of other routes.

```
router bgp 65100
 network 172.16.0.0
 network 172.17.0.0
 neighbor 10.69.232.70 remote-as 65200
 neighbor 10.69.232.70 route-map set-as-path out
!
route-map set-as-path 10 permit
 match address 1
 set as-path prepend 65100 65100
!
route-map set-as-path 20 permit
 match address 2
```

```
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 1 permit 172.17.0.0 0.0.255.255
!
access-list 2 permit 0.0.0.0 255.255.255.255
```

Filtering BGP Prefixes with Prefix Lists Examples

This section contains the following examples:

Filtering BGP Prefixes Using a Single Prefix List

The following example shows how a prefix list denies the default route 0.0.0.0/0:

```
ip prefix-list abc deny 0.0.0.0/0
```

The following example shows how a prefix list permits a route that matches the prefix 10.0.0.0/8:

```
ip prefix-list abc permit 10.0.0.0/8
```

The following example shows how to configure the BGP process so that it accepts only prefixes with a prefix length of /8 to /24:

```
router bgp 40000
 network 10.20.20.0
 distribute-list prefix max24 in
!
ip prefix-list max24 seq 5 permit 0.0.0.0/0 ge 8 le 24
```

The following example configuration shows how to conditionally originate a default route (0.0.0.0/0) in RIP when a prefix 10.1.1.0/24 exists in the routing table:

```
ip prefix-list cond permit 10.1.1.0/24
!
route-map default-condition permit 10
 match ip address prefix-list cond
!
router rip
 default-information originate route-map default-condition
```

The following example shows how to configure BGP to accept routing updates from 192.168.1.1 only, besides filtering on the prefix length:

```
router bgp 40000
 distribute-list prefix max24 gateway allowlist in
!
ip prefix-list allowlist seq 5 permit 192.168.1.1/32
!
```

The following example shows how to direct the BGP process to filter incoming updates to the prefix using *name1*, and match the gateway (next hop) of the prefix being updated to the prefix list *name2*, on GigabitEthernet interface 0/0/0:

```
router bgp 103
 distribute-list prefix name1 gateway name2 in gigabitethernet 0/0/0
```

Filtering BGP Prefixes Using a Group of Prefixes

The following example shows how to configure BGP to permit routes with a prefix length up to 24 in network 192/8:

```
ip prefix-list abc permit 192.0.0.0/8 le 24
```

The following example shows how to configure BGP to deny routes with a prefix length greater than 25 in 192/8:

```
ip prefix-list abc deny 192.0.0.0/8 ge 25
```

The following example shows how to configure BGP to permit routes with a prefix length greater than 8 and less than 24 in all address space:

```
ip prefix-list abc permit 0.0.0.0/0 ge 8 le 24
```

The following example shows how to configure BGP to deny routes with a prefix length greater than 25 in all address space:

```
ip prefix-list abc deny 0.0.0.0/0 ge 25
```

The following example shows how to configure BGP to deny all routes in network 10/8, because any route in the Class A network 10.0.0.0/8 is denied if its mask is less than or equal to 32 bits:

```
ip prefix-list abc deny 10.0.0.0/8 le 32
```

The following example shows how to configure BGP to deny routes with a mask greater than 25 in 192.168.1.0/24:

```
ip prefix-list abc deny 192.168.1.0/24 ge 25
```

The following example shows how to configure BGP to permit all routes:

```
ip prefix-list abc permit 0.0.0.0/0 le 32
```

Adding or Deleting Prefix List Entries

You can add or delete individual entries in a prefix list if a prefix list has the following initial configuration:

```
ip prefix-list abc deny 0.0.0.0/0 le 7
ip prefix-list abc deny 0.0.0.0/0 ge 25
ip prefix-list abc permit 192.168.0.0/15
```

The following example shows how to delete an entry from the prefix list so that 192.168.0.0 is not permitted, and add a new entry that permits 10.0.0.0/8:

```
no ip prefix-list abc permit 192.168.0.0/15
ip prefix-list abc permit 10.0.0.0/8
```

The new configuration is as follows:

```
ip prefix-list abc deny 0.0.0.0/0 le 7
ip prefix-list abc deny 0.0.0.0/0 ge 25
ip prefix-list abc permit 10.0.0.0/8
```

Filtering Traffic Using Community Lists Examples

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

The first example shows how the route map named set-community is applied to the outbound updates to the neighbor 172.16.232.50. The routes that pass access list 1 have the special community attribute value no-export. The remaining routes are advertised normally. This special community value automatically prevents the advertisement of those routes by the BGP speakers in autonomous system 200.

```
router bgp 100
 neighbor 172.16.232.50 remote-as 200
 neighbor 172.16.232.50 send-community
 neighbor 172.16.232.50 route-map set-community out
!
route-map set-community permit 10
 match address 1
 set community no-export
!
route-map set-community permit 20
 match address 2
```

The second example shows how the route map named *set-community* is applied to the outbound updates to neighbor 172.16.232.90. All the routes that originate from autonomous system 70 have the community values 200 200 added to their already existing values. All other routes are advertised as normal.

```
route-map bgp 200
 neighbor 172.16.232.90 remote-as 100
 neighbor 172.16.232.90 send-community
 neighbor 172.16.232.90 route-map set-community out
!
route-map set-community permit 10
 match as-path 1
 set community 200 200 additive
!
route-map set-community permit 20
!
ip as-path access-list 1 permit 70$
ip as-path access-list 2 permit .*
```

Filtering Traffic Using AS-path Filters Example

The following example shows BGP path filtering by neighbor. Only the routes that pass autonomous system path access list 2 will be sent to 192.168.12.10. Similarly, only routes passing access list 3 will be accepted from 192.168.12.10.

```
router bgp 200
 neighbor 192.168.12.10 remote-as 100
 neighbor 192.168.12.10 filter-list 1 out
 neighbor 192.168.12.10 filter-list 2 in
 exit
ip as-path access-list 1 permit _109_
ip as-path access-list 2 permit _200$
ip as-path access-list 2 permit ^100$
ip as-path access-list 3 deny _690$
ip as-path access-list 3 permit .*
```

Filtering Traffic with AS-path Filters Using 4-Byte Autonomous System Numbers Examples

Asplain Default Format in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)SXI1, and Later Releases

The following example is available in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases and shows BGP path filtering by neighbor using 4-byte autonomous system numbers in asplain format. Only the routes that pass autonomous system path access list 2 will be sent to 192.168.3.2.

```
ip as-path access-list 2 permit ^65536$
router bgp 65538
 address-family ipv4 unicast
  neighbor 192.168.3.2 remote-as 65550
  neighbor 192.168.3.2 activate
  neighbor 192.168.3.2 filter-list 2 in
  end
```

Asdot Default Format in Cisco IOS Release 12.0(32)S12, and 12.4(24)T

The following example available in Cisco IOS Release 12.0(32)S12, 12.4(24)T, and later releases shows BGP path filtering by neighbor using 4-byte autonomous system numbers in asdot format. Only the routes that pass autonomous system path access list 2 will be sent to 192.168.3.2.

Note In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, this example works if you have configured asdot as the default display format using the **bgp asnotation dot** command.

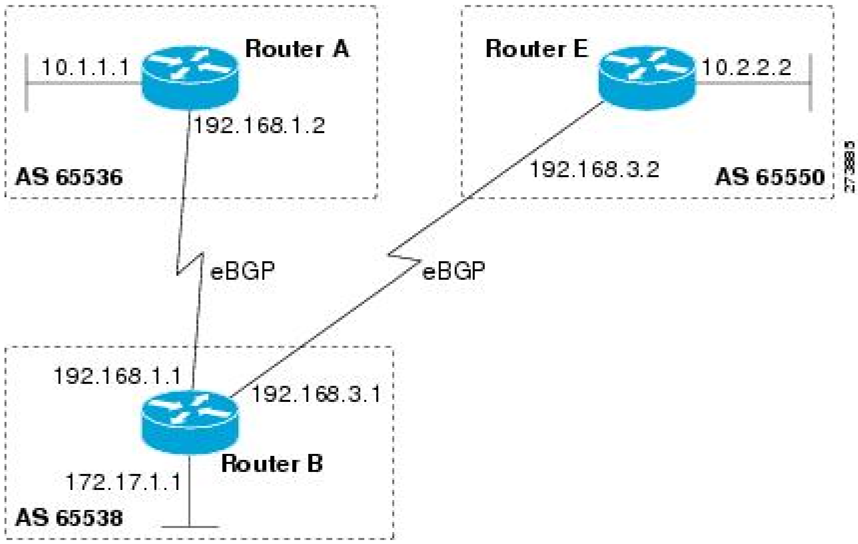
```
ip as-path access-list 2 permit ^1\.0$
router bgp 1.2
 address-family ipv4 unicast
  neighbor 192.168.3.2 remote-as 1.14
  neighbor 192.168.3.2 filter-list 2 in
  end
```

Filtering Traffic Using Extended Community Lists with 4-Byte Autonomous System Numbers Example

Asplain Default Format in Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)SXI1, and Later Releases

The following example shows how to filter traffic by creating an extended BGP community list to control outbound routes. In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, 12.2(33)SXI1, and later releases, extended BGP communities support 4-byte autonomous system numbers in the regular expressions in asplain by default. Extended community attributes are used to configure, filter, and identify routes for VRF instances and MPLS VPNs. The **ip extcommunity-list**command is used to configure named or numbered extended community lists. All of the standard rules of access lists apply to the configuration of extended community lists. Regular expressions are supported by the expanded range of extended community list numbers.

Figure 21 *BGP Topology for Filtering Traffic Using Extended Community Lists with 4-Byte Autonomous System Numbers in Asplain Format*



Note A sequence number is applied to all extended community list entries by default regardless of the configuration mode. Explicit sequencing and resequencing of extended community list entries can be configured only in IP extended community-list configuration mode and not in global configuration mode.

In this exam the figure above is configured with an extended named community list to specify that the BGP peer at 192.1681.2 is not sent advertisements about any path through or from the 4-byte autonomous system 65550. The IP extended community-list configuration mode is used, and the ability to resequence entries is shown.

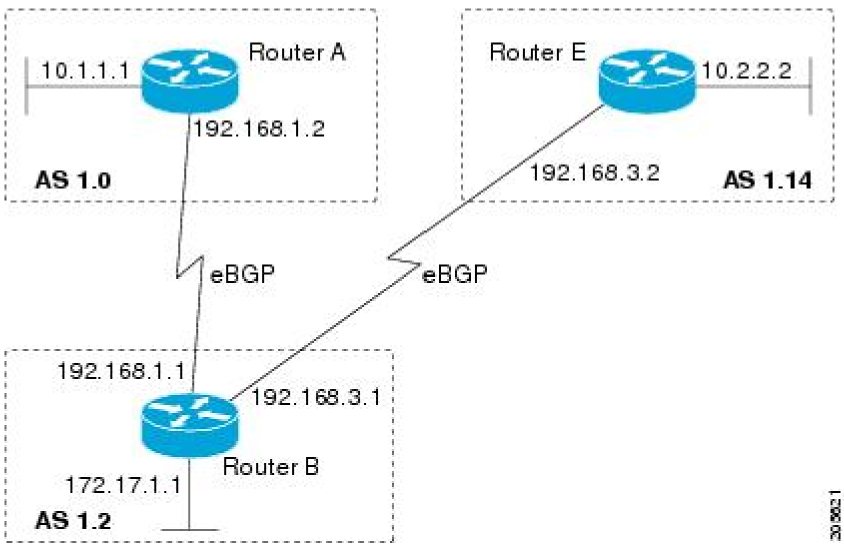
```
ip extcommunity-list expanded DENY65550
 10 deny _65550_
 20 deny ^65550 .*
 resequence 50 100
 exit
router bgp 65538
 network 172.17.1.0 mask 255.255.255.0
 address-family ipv4 unicast
  neighbor 192.168.3.2 remote-as 65550
  neighbor 192.168.1.2 remote-as 65536
  neighbor 192.168.3.2 activate
  neighbor 192.168.1.2 activate
  end
show ip extcommunity-list DENY65550
```

Asdot Default Format in Cisco IOS Release 12.0(32)S12, and 12.4(24)T

The following example shows how to filter traffic by creating an extended BGP community list to control outbound routes. In Cisco IOS Release 12.0(32)S12, 12.4(24)T, and later releases, extended BGP communities support 4-byte autonomous system numbers in the regular expressions in asdot format only. Extended community attributes are used to configure, filter, and identify routes for VRF instances and MPLS VPNs. The **ip extcommunity-list**command is used to configure named or numbered extended community lists. All of the standard rules of access lists apply to the configuration of extended community lists. Regular expressions are supported by the expanded range of extended community list numbers.

Note In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SXI1, and later releases, this example works if you have configured asdot as the default display format using the **bgp asnotation dot** command.

Figure 22 BGP Topology for Filtering Traffic Using Extended Community Lists with 4-Byte Autonomous System Numbers in Asdot Format



Note A sequence number is applied to all extended community list entries by default regardless of the configuration mode. Explicit sequencing and resequencing of extended community list entries can be configured only in IP extended community-list configuration mode and not in global configuration mode.

In this exam the figure above is configured with an extended named community list to specify that the BGP peer at 192.1681.2 is not sent advertisements about any path through or from the 4-byte autonomous system 65550. The IP extended community-list configuration mode is used, and the ability to resequence entries is shown.

```
ip extcommunity-list expanded DENY114
 10 deny _1\.14_
 20 deny ^1\.14 .*
 resequence 50 100
 exit
router bgp 1.2
 network 172.17.1.0 mask 255.255.255.0
 address-family ipv4 unicast
  neighbor 192.168.3.2 remote-as 1.14
  neighbor 192.168.1.2 remote-as 1.0
  neighbor 192.168.3.2 activate
  neighbor 192.168.1.2 activate
  end
show ip extcommunity-list DENY114
```

Filtering Traffic Using a BGP Route Map Example

The following example shows how to use an address family to configure BGP so that any unicast and multicast routes from neighbor 10.1.1.1 are accepted if they match access list 1:

```
route-map filter-some-multicast
 match ip address 1
 exit
router bgp 65538
 neighbor 10.1.1.1 remote-as 65537
 address-family ipv4 unicast
  neighbor 10.1.1.1 activate
  neighbor 10.1.1.1 route-map filter-some-multicast in
  exit
 exit
router bgp 65538
 neighbor 10.1.1.1 remote-as 65537
 address-family ipv4 multicast
  neighbor 10.1.1.1 activate
  neighbor 10.1.1.1 route-map filter-some-multicast in
  end
```

Filtering Traffic Using Continue Clauses in a BGP Route Map Examples

The following example shows continue clause configuration in a route map sequence.

Note Continue clauses in outbound route maps are supported only in Cisco IOS Release 12.0(31)S, 12.2(33)SB, 12.2(33)SRB, 12.2(33)SXI, 12.4(4)T, and later releases.

The first continue clause in route map entry 10 indicates that the route map will go to route map entry 30 if a successful matches occurs. If a match does not occur, the route map will "fall through" to route map entry 20. If a successful match occurs in route map entry 20, the set action will be executed and the route map will not evaluate any additional route map entries. Only the first successful match ip address clause is supported.

If a successful match does not occur in route map entry 20, the route map will "fall through" to route map entry 30. This sequence does not contain a match clause, so the set clause will be automatically executed and the continue clause will go to the next route map entry because a sequence number is not specified.

If there are no successful matches, the route map will "fall through" to route map entry 30 and execute the set clause. A sequence number is not specified for the continue clause so route map entry 40 will be evaluated.

There are two behaviors that can occur when the same **set** command is repeated in subsequent continue clause entries. For **set** commands that configure an additive or accumulative value (for example, **set community additive**, **set extended community additive**, and **set as-path prepend**), subsequent values are added by subsequent entries. The following example illustrates this behavior. After each set of match clauses, a **set as-path prepend** command is configured to add an autonomous system number to the as-path. After a match occurs, the route map stops evaluating match clauses and starts executing the set clauses, in the order in which they were configured. Depending on how many successful match clauses occur, the as-path is prepended by one, two, or three autonomous system numbers.

```
route-map ROUTE-MAP-NAME permit 10
 match ip address 1
 match metric 10
 set as-path prepend 10
 continue 30
 !
```

```
route-map ROUTE-MAP-NAME permit 20
 match ip address 2
 match metric 20
 set as-path prepend 10 10
 !
route-map ROUTE-MAP-NAME permit 30
 set as-path prepend 10 10 10
 continue
 !
route-map ROUTE-MAP-NAME permit 40
 match community 10:1
 set local-preference 104
```

In this example, the same **set** command is repeated in subsequent continue clause entries but the behavior is different from the first example. For **set**commands that configure an absolute value, the value from the last instance will overwrite the previous value(s). The following example illustrates this behavior. The set clause value in sequence 20 overwrites the set clause value from sequence 10. The next hop for prefixes from the 172.16/16 network is set to 10.2.2.2 and not 10.1.1.1.

```
ip prefix-list 1 permit 172.16.0.0/16
ip prefix-list 2 permit 192.168.1.0/24
route-map RED permit 10
 match ip address prefix-list 1
 set ip next hop 10.1.1.1
 continue 20
 exit
route-map RED permit 20
 match ip address prefix-list 2
 set ip next hop 10.2.2.2
 end
```

Note Route maps have a linear behavior and not a nested behavior. Once a route is matched in a route map permit entry with a continue command clause, it will not be processed by the implicit deny at the end of the route-map. The following example illustrates this case.

In the following example, when routes match an as-path of 10, 20, or 30, the routes are permitted and the continue clause jumps over the explicit deny clause to process the match ip address prefix list. If a match occurs here, the route metric is set to 100. Only routes that do not match an as-path of 10, 20, or 30 and do match a community number of 30 are denied. To deny other routes, you must configure an explicit deny statement.

```
route-map test permit 10
 match as-path 10 20 30
 continue 30
 exit
route-map test deny 20
 match community 30
 exit
route-map test permit 30
 match ip address prefix-list 1
 set metric 100
 exit
```

Where to Go Next

- To configure advanced BGP feature tasks, proceed to the "Configuring Advanced BGP Features" module.
- To configure BGP neighbor session options, proceed to the "Configuring BGP Neighbor Session Options" module.

- To configure internal BGP tasks, proceed to the "Configuring Internal BGP Features" module.

Additional References

The following sections provide references related to connecting to a service provider using external BGP.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
BGP commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
BGP overview	"Cisco BGP Overview" module
Configuring basic BGP tasks	"Configuring a Basic BGP Network" module
BGP fundamentals and description	*Large-Scale IP Network Solutions* , Khalid Raza and Mark Turner, Cisco Press, 2000
Implementing and controlling BGP in scalable networks	*Building Scalable Cisco Networks* , Catherine Paquet and Diane Teare, Cisco Press, 2001
Interdomain routing basics	*Internet Routing Architectures* , Bassam Halabi, Cisco Press, 1997

Standards

Standard	Title
MDT SAFI	MDT SAFI

MIBs

MIB	MIBs Link
CISCO-BGP4-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1772	*Application of the Border Gateway Protocol in the Internet*

RFC	Title
RFC 1773	*Experience with the BGP Protocol*
RFC 1774	*BGP-4 Protocol Analysis*
RFC 1930	*Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)*
RFC 2519	*A Framework for Inter-Domain Route Aggregation*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 2918	*Route Refresh Capability for BGP-4*
RFC 3392	*Capabilities Advertisement with BGP-4*
RFC 4271	A Border Gateway Protocol 4 (BGP-4)
RFC 4893	BGP Support for Four-Octet AS Number Space
RFC 5396	Textual Representation of Autonomous system (AS) Numbers
RFC 5398	Autonomous System (AS) Number Reservation for Documentation Use

Technical Assistance.

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Connecting to a Service Provider Using External BGP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 16 Feature Information for Connecting to a Service Provider Using External BGP

Feature Name	Releases	Feature Configuration Information
BGP Increased Support of Numbered AS-Path Access Lists to 500	12.0(22)S 12.2(15)T 12.2(18)S 12.2(18)SXD 12.2(27)SBC 15.0(1)S	The BGP Increased Support of Numbered AS-Path Access Lists to 500 feature increases the maximum number of autonomous systems access lists that can be configured using the **ip as-path access-list** command from 199 to 500.
BGP Named Community Lists	12.2(8)T 12.2(14)S 15.0(1)S	The BGP Named Community Lists feature introduces a new type of community list called the named community list. The BGP Named Community Lists feature allows the network operator to assign meaningful names to community lists and increases the number of community lists that can be configured. A named community list can be configured with regular expressions and with numbered community lists. All rules of numbered communities apply to named community lists except that there is no limitation on the number of community attributes that can be configured for a named community list.

Feature Name	Releases	Feature Configuration Information
BGP Prefix-Based Outbound Route Filtering	12.0(22)S 12.2(4)T 12.2(14)S 15.0(1)S	The BGP Prefix-Based Outbound Route Filtering feature uses BGP ORF send and receive capabilities to minimize the number of BGP updates that are sent between BGP peers. Configuring this feature can help reduce the amount of system resources required for generating and processing routing updates by filtering out unwanted routing updates at the source. For example, this feature can be used to reduce the amount of processing required on a router that is not accepting full routes from a service provider network.
BGP Route-Map Continue	12.0(24)S 12.2(18)S 12.2(18)SXD 12.2(27)SBC 12.3(2)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Route-Map Continue feature introduces the continue clause to BGP route map configuration. The continue clause allows for more programmable policy configuration and route filtering and introduces the capability to execute additional entries in a route map after an entry is executed with successful match and set clauses. Continue clauses allow the network operator to configure and organize more modular policy definitions so that specific policy configurations need not be repeated within the same route map.
BGP Route-Map Continue Support for an Outbound Policy	12.0(31)S 12.2(33)SB 12.2(33)SRB 12.2(33)SXI 12.4(4)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Route-Map Continue Support for an Outbound Policy feature introduces support for continue clauses to be applied to outbound route maps.

Feature Name	Releases	Feature Configuration Information
BGP Route-Map Policy List Support	12.0(22)S 12.2(15)T 12.2(18)S 12.2(18)SXD 12.2(27)SBC 15.0(1)S	The BGP Route-Map Policy List Support feature introduces new functionality to BGP route maps. This feature adds the capability for a network operator to group route map match clauses into named lists called policy lists. A policy list functions like a macro. When a policy list is referenced in a route map, all of the match clauses are evaluated and processed as if they had been configured directly in the route map. This enhancement simplifies the configuration of BGP routing policy in medium-size and large networks because a network operator can preconfigure policy lists with groups of match clauses and then reference these policy lists within different route maps. The network operator no longer needs to manually reconfigure each recurring group of match clauses that occur in multiple route map entries.

Feature Name	Releases	Feature Configuration Information
BGP Support for 4-Byte ASN	12.0(32)S12 12.0(32)SY8 12.0(33)S3 12.2(33)SRE 12.2(33)XNE 12.2(33)SXI1 12.4(24)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Support for 4-Byte ASN feature introduced support for 4-byte autonomous system numbers. Because of increased demand for autonomous system numbers, in January 2009 the IANA will start to allocate 4-byte autonomous system numbers in the range from 65536 to 4294967295. In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, and 12.2(33)SXI1, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default regular expression match and output display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain format and the asdot format as described in RFC 5396. To change the default regular expression match and output display of 4-byte autonomous system numbers to asdot format, use the **bgp asnotation dot** command. In Cisco IOS Release 12.0(32)S12, and 12.4(24)T, the Cisco implementation of 4-byte autonomous system numbers uses asdot as the only configuration format, regular expression match, and output display, with no asplain support. The following commands were introduced or modified by this feature: **bgp asnotation dot**, **bgp confederation identifier**, **bgp confederation peers**, all **clear ip bgp**commands that configure an autonomous system number, **ip as-path access-list**, **ip extcommunity-list**, **match source-protocol**, **neighbor local-**

Feature Name	Releases	Feature Configuration Information
		as, **neighbor remote-as**, **neighbor soo**, **redistribute (**IP**)**, **router bgp**, **route-target**, **set as-path**, **set extcommunity**, **set origin**, **soo**, all **show ip bgp** commands that display an autonomous system number, and **show ip extcommunity-list**.
BGP Support for Named Extended Community Lists	12.2(25)S 12.2(27)SBC 12.2(33)SRA 12.2(33)SXH 12.3(11)T 15.0(1)S	The BGP Support for Named Extended Community Lists feature introduces the ability to configure extended community lists using names in addition to the existing numbered format.
BGP Support for Sequenced Entries in Extended Community Lists	12.2(25)S 12.2(27)SBC 12.2(33)SRA 12.2(33)SXH 12.3(11)T 15.0(1)S	The BGP Support for Sequenced Entries in Extended Community Lists feature introduces automatic sequencing of individual entries in BGP extended community lists. This feature also introduces the ability to remove or resequence extended community list entries without deleting the entire existing extended community list.
BGP 4 Prefix Filter and Inbound Route Maps	Cisco IOS XE 3.1.0SG	

Configuring BGP Neighbor Session Options

This module describes configuration tasks to configure various options involving Border Gateway Protocol (BGP) neighbor peer sessions. BGP is an interdomain routing protocol designed to provide loop-free routing between organizations. This module contains tasks that use BGP neighbor session commands to configure:

- Fast session deactivation
- Bidirectional Forwarding Detection (BFD) for BGP IPv6 neighbors
- A router to automatically reestablish a BGP neighbor peering session when the peering session has been disabled or brought down
- Options to help an autonomous system migration
- TTL Security Check, a lightweight security mechanism to protect External BGP (eBGP) peering sessions from CPU-utilization-based attacks

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring BGP Neighbor Session Options

Before configuring advanced BGP features you should be familiar with the "Cisco BGP Overview" module and the "Configuring a Basic BGP Network" module.

Restrictions for Configuring BGP Neighbor Session Options

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

Information About Configuring BGP Neighbor Session Options

BGP Neighbor Sessions

BGP is mainly used to connect a local network to an external network to gain access to the Internet or to connect to other organizations. A BGP-speaking router does not discover another BGP-speaking device automatically. A network administrator usually manually configures the relationships between BGP-speaking routers.

A BGP neighbor device is a BGP-speaking router that has an active TCP connection to another BGP-speaking device. This relationship between BGP devices is often referred to as a peer instead of neighbor because a neighbor may imply the idea that the BGP devices are directly connected with no other router in between. Configuring BGP neighbor or peer sessions uses BGP neighbor session commands so this module will prefer the use of the term "neighbor" over "peer."

BGP Support for Fast Peering Session Deactivation

BGP Hold Timer

By default, the BGP hold timer is set to run every 180 seconds in Cisco IOS software. This timer value is set as the default to protect the BGP routing process from instability that can be caused by peering sessions with other routing protocols. BGP routers typically carry large routing tables, so frequent session resets are not desirable.

BGP Fast Peering Session Deactivation

BGP fast peering session deactivation improves BGP convergence and response time to adjacency changes with BGP neighbors. This feature is event driven and configured on a per-neighbor basis. When this feature is enabled, BGP will monitor the peering session with the specified neighbor. Adjacency changes are detected and terminated peering sessions are deactivated in between the default or configured BGP scanning interval.

Selective Address Tracking for BGP Fast Session Deactivation

In Cisco IOS Release 12.4(4)T, 12.2(31)SB, 12.2(33)SRB, and later releases, the BGP Selective Address Tracking feature introduced the use of a route map with BGP fast session deactivation. The **route-map** keyword and *map-name* argument are used with the **neighbor fall-over** BGP neighbor session command to determine if a peering session with a BGP neighbor should be reset when a route to the BGP peer changes. The route map is evaluated against the new route, and if a deny statement is returned, the peer session is reset. The route map is not used for session establishment.

Note Only **match ip address** and **match source-protocol** commands are supported in the route map. No **set** commands or other **match** commands are supported.

BFD Support of BGP IPv6 Neighbors

In Cisco IOS Release 15.1(2)S and later releases, Bidirectional Forwarding Detection (BFD) can be used to track fast forwarding path failure of BGP neighbors that have an IPv6 address. BFD is a detection protocol that is designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols. BFD provides faster reconvergence time for BGP after a forwarding path failure.

BGP Neighbor Session Restart After the Max-Prefix Limit Is Reached

Prefix Limits and BGP Peering Sessions

There is a configurable limit on the maximum number of prefixes that a router that is running BGP can receive from a peer router. This limit is configured with the **neighbor maximum-prefix** command. When the router receives too many prefixes from a peer router and the maximum-prefix limit is exceeded, the peering session is disabled or brought down. The session stays down until the network operator manually brings the session back up by entering the **clear ip bgp** command. Entering the **clear ip bgp** command clears stored prefixes.

BGP Neighbor Session Restart with the Maximum Prefix Limit

In Cisco IOS Release 12.0(22)S, 12.2(15)T, 12.2(18)S, and later releases, the **restart** keyword was added to enhance the capabilities of the **neighbor maximum-prefix** command. This enhancement allows the network operator to configure a router to automatically reestablish a BGP neighbor peering session when the peering session has been disabled or brought down. There is configurable time interval at which peering can be reestablished automatically. The configurable timer argument for the **restart** keyword is specified in minutes. The time range is from 1 to 65,535 minutes.

BGP Network Autonomous System Migration

Autonomous System Migration for BGP Networks

Autonomous-system migration can be necessary when a telecommunications or Internet service provider purchases another network. It is desirable for the provider to be able to integrate the second autonomous system without disrupting existing customer peering arrangements. The amount of configuration required in the customer networks can make this a cumbersome task that is difficult to complete without disrupting service.

Dual Autonomous System Support for BGP Network Autonomous System Migration

In Cisco IOS Release 12.0(29)S, 12.3(14)T, 12.2(33)SXH, and later releases, support was added for dual BGP autonomous system configuration to allow a secondary autonomous system to merge under a primary autonomous system, without disrupting customer peering sessions. The configuration of this feature is transparent to customer networks. Dual BGP autonomous system configuration allows a router to appear, to external peers, as a member of secondary autonomous system during the autonomous system migration. This feature allows the network operator to merge the autonomous systems and then later migrate customers to new configurations during normal service windows without disrupting existing peering arrangements.

The **neighbor local-as** command is used to customize the AS_PATH attribute by adding and removing autonomous system numbers for routes received from eBGP neighbors. This feature allows a router to appear to external peers as a member of another autonomous system for the purpose of autonomous system number migration. This feature simplifies this process of changing the autonomous system number in a BGP network by allowing the network operator to merge a secondary autonomous system into a primary autonomous system and then later update the customer configurations during normal service windows without disrupting existing peering arrangements.

BGP Autonomous System Migration Support for Confederations, Individual Peering Sessions, and Peer Groupings

This feature supports confederations, individual peering sessions, and configurations applied through peer groups and peer templates. If this feature is applied to a group peers, the individual peers cannot be customized.

Ingress Filtering During BGP Autonomous System Migration

Autonomous system path customization increases the possibility that routing loops can be created if such customization is misconfigured. The larger the number of customer peerings, the greater the risk. You can minimize this possibility by applying policies on the ingress interfaces to block the autonomous system number that is in transition or routes that have no **local-as** configuration.

> **Caution** BGP prepends the autonomous system number from each BGP network that a route traverses to maintain network reachability information and to prevent routing loops. This feature should be configured only for autonomous system migration and should be deconfigured after the transition has been completed. This procedure should be attempted only by an experienced network operator, as routing loops can be created with improper configuration.

BGP Network Migration to 4-Byte Autonomous System Numbers

The BGP Support for 4-Byte ASN feature introduced support for 4-byte autonomous system numbers. Because of increased demand for autonomous system numbers, in January 2009 the IANA will start to allocate 4-byte autonomous system numbers in the range from 65536 to 4294967295.

The Cisco implementation of 4-byte autonomous system numbers supports RFC 4893. RFC 4893 was developed to allow BGP to support a gradual transition from 2-byte autonomous system numbers to 4-byte autonomous system numbers. A new reserved (private) autonomous system number, 23456, was created by RFC 4893 and this number cannot be configured as an autonomous system number in the Cisco IOS CLI.

Migrating your BGP network to 4-byte autonomous system numbers requires some planning. If you are upgrading to an image that supports 4-byte autonomous system numbers, you can still use 2-byte autonomous system numbers. The **show** command output and regular expression match are not changed and remain in asplain (decimal value) format for 2-byte autonomous system numbers regardless of the format configured for 4-byte autonomous system numbers.

To ensure a smooth transition, we recommend that all BGP speakers within an autonomous system that is identified using a 4-byte autonomous system number be upgraded to support 4-byte autonomous system numbers.

For details about steps to perform to upgrade a BGP network to full 4-byte autonomous system support, see the Migration Guide for Explaining 4-Byte Autonomous System white paper.

TTL Security Check for BGP Neighbor Sessions

BGP Support for the TTL Security Check

When implemented for BGP, the TTL Security Check feature introduces a lightweight security mechanism to protect eBGP neighbor sessions from CPU utilization-based attacks. These types of attacks are typically brute force Denial of Service (DoS) attacks that attempt to disable the network by flooding the network with IP packets that contain forged source and destination IP addresses.

The TTL Security Check feature protects the eBGP neighbor session by comparing the value in the TTL field of received IP packets against a hop count that is configured locally for each eBGP neighbor session. If the value in the TTL field of the incoming IP packet is greater than or equal to the locally configured value, the IP packet is accepted and processed normally. If the TTL value in the IP packet is less than the locally configured value, the packet is silently discarded and no Internet Control Message Protocol (ICMP) message is generated. This is designed behavior; a response to a forged packet is unnecessary.

Although it is possible to forge the TTL field in an IP packet header, accurately forging the TTL count to match the TTL count from a trusted peer is impossible unless the network to which the trusted peer belongs has been compromised.

The TTL Security Check feature supports both directly connected neighbor sessions and multihop eBGP neighbor sessions. The BGP neighbor session is not affected by incoming packets that contain invalid TTL values. The BGP neighbor session will remain open, and the router will silently discard the invalid packet. The BGP session, however, can still expire if keepalive packets are not received before the session timer expires.

TTL Security Check for BGP Neighbor Sessions

The BGP Support for TTL Security Check feature is configured with the **neighbor ttl-security** command in router configuration mode or address family configuration mode. When this feature is enabled, BGP will establish or maintain a session only if the TTL value in the IP packet header is equal to or greater than the TTL value configured for the peering session. Enabling this feature secures the eBGP session in the incoming direction only and has no effect on outgoing IP packets or the remote router. The *hop-count* argument is used to configure the maximum number of hops that separate the two peers. The TTL value is determined by the router from the configured hop count. The value for this argument is a number from 1 to 254.

TTL Security Check Support for Multihop BGP Neighbor Sessions

The BGP Support for TTL Security Check feature supports both directly connected neighbor sessions and multihop neighbor sessions. When this feature is configured for a multihop neighbor session, the **neighbor ebgp-multihop** router configuration command cannot be configured and is not needed to establish the neighbor session. These commands are mutually exclusive, and only one command is required to establish a multihop neighbor session. If you attempt to configure both commands for the same peering session, an error message will be displayed in the console.

To configure this feature for an existing multihop session, you must first disable the existing neighbor session with the **no neighbor ebgp-multihop** command. The multihop neighbor session will be restored when you enable this feature with the **neighbor ttl-security** command.

This feature should be configured on each participating router. To maximize the effectiveness of this feature, the *hop-count* argument should be strictly configured to match the number of hops between the local and external network. However, you should also consider path variation when configuring this feature for a multihop neighbor session.

Benefits of the BGP Support for TTL Security Check

The BGP Support for TTL Security Check feature provides an effective and easy-to-deploy solution to protect eBGP neighbor sessions from CPU utilization-based attacks. When this feature is enabled, a host cannot attack a BGP session if the host is not a member of the local or remote BGP network or if the host is not directly connected to a network segment between the local and remote BGP networks. This solution greatly reduces the effectiveness of DoS attacks against a BGP autonomous system.

BGP Support for TCP Path MTU Discovery per Session

- Path MTU Discovery, page 223
- BGP Neighbor Session TCP PMTUD, page 223

Path MTU Discovery

The IP protocol family was designed to use a wide variety of transmission links. The maximum IP packet length is 65000 bytes. Most transmission links enforce a smaller maximum packet length limit, called the maximum transmission unit (MTU), which varies with the type of the transmission link. The design of IP accommodates link packet length limits by allowing intermediate routers to fragment IP packets as necessary for their outgoing links. The final destination of an IP packet is responsible for reassembling its fragments as necessary.

All TCP sessions are bounded by a limit on the number of bytes that can be transported in a single packet, and this limit is known as the maximum segment size (MSS). TCP breaks up packets into chunks in a transmit queue before passing packets down to the IP layer. A smaller MSS may not be fragmented at an IP device along the path to the destination device, but smaller packets increase the amount of bandwidth needed to transport the packets. The maximum TCP packet length is determined by both the MTU of the outbound interface on the source device and the MSS announced by the destination device during the TCP setup process.

Path MTU discovery (PMTUD) was developed as a solution to the problem of finding the optimal TCP packet length. PMTUD is an optimization (detailed in RFC 1191) wherein a TCP connection attempts to send the longest packets that will not be fragmented along the path from source to destination. It does this by using a flag, don’t fragment (DF), in the IP packet. This flag is supposed to alter the behavior of an intermediate router that cannot send the packet across a link because it is too long. Normally the flag is off, and the router should fragment the packet and send the fragments. If a router tries to forward an IP datagram, with the DF bit set, to a link that has a lower MTU than the size of the packet, the router will drop the packet and return an ICMP Destination Unreachable message to the source of this IP datagram, with the code indicating "fragmentation needed and DF set." When the source device receives the ICMP message, it will lower the send MSS, and when TCP retransmits the segment, it will use the smaller segment size.

BGP Neighbor Session TCP PMTUD

TCP path MTU discovery is enabled by default for all BGP neighbor sessions, but there are situations when you may want to disable TCP path MTU discovery for one or all BGP neighbor sessions. Although PMTUD works well for larger transmission links (for example, Packet over Sonet links), a badly configured TCP implementation or a firewall may slow or stop the TCP connections from forwarding any packets. In this type of situation, you may need to disable TCP path MTU discovery. In Cisco IOS Release 12.2(33)SRA, 12.2(31)SB, 12.2(33)SXH, 12.4(20)T, and later releases, configuration options were introduced to permit TCP path MTU discovery to be disabled, or subsequently reenabled, either for a single BGP neighbor session or for all BGP sessions. To disable the TCP path MTU discovery globally for all BGP neighbors, use the **no bgp transport path-mtu-discovery** command in router configuration mode. To disable the TCP path MTU discovery for a single neighbor, use the **no neighbor transport path-mtu-discovery** command in router or address family configuration modes. For more details, see the Disabling TCP Path MTU Discovery Globally for All BGP Sessions, page 241 or the Disabling TCP Path MTU Discovery for a Single BGP Neighbor, page 243.

BGP Dynamic Neighbors

Support for the BGP Dynamic Neighbors feature was introduced in Cisco IOS Release 12.2(33)SXH on the Cisco Catalyst 6500 series switches. BGP dynamic neighbor support allows BGP peering to a group of remote neighbors that are defined by a range of IP addresses. Each range can be configured as a subnet IP address. BGP dynamic neighbors are configured using a range of IP addresses and BGP peer groups.

After a subnet range is configured for a BGP peer group and a TCP session is initiated by another router for an IP address in the subnet range, a new BGP neighbor is dynamically created as a member of that group.

After the initial configuration of subnet ranges and activation of the peer group (referred to as a *listen range group*), dynamic BGP neighbor creation does not require any further CLI configuration on the initial router. Other routers can establish a BGP session with the initial router, but the initial router need not establish a BGP session to other routers if the IP address of the remote peer used for the BGP session is not within the configured range.

To support the BGP Dynamic Neighbors feature, the output for the **show ip bgp neighbors**, **show ip bgp peer-group**, and **show ip bgp summary** commands was updated to display information about dynamic neighbors.

A dynamic BGP neighbor will inherit any configuration for the peer group. In larger BGP networks, implementing BGP dynamic neighbors can reduce the amount and complexity of CLI configuration and save CPU and memory usage. Only IPv4 peering is supported.

How to Configure BGP Neighbor Session Options

Configuring Fast Session Deactivation

The tasks in this section show how to configure BGP next-hop address tracking. BGP next-hop address tracking significantly improves the response time of BGP to next-hop changes in the RIB. However, unstable Interior Gateway Protocol (IGP) peers can introduce instability to BGP neighbor sessions. We recommend that you aggressively dampen unstable IGP peering sessions to reduce the possible impact to BGP. For more details about route dampening, see the "Configuring Internal BGP Features" module.

Configuring Fast Session Deactivation for a BGP Neighbor

Perform this task to establish a peering session with a BGP neighbor and then configure the peering session for fast session deactivation to improve the network convergence time if the peering session is deactivated.

Enabling fast session deactivation for a BGP neighbor can significantly improve BGP convergence time. However, unstable IGP peers can still introduce instability to BGP neighbor sessions. We recommend that you aggressively dampen unstable IGP peering sessions to reduce the possible impact to BGP.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*]
5. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
6. **neighbor** *ip-address* **fall-over**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 50000`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address family-specific configurations. • The example creates an IPv4 unicast address family session.
Step 5	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 10.0.0.1 remote-as 50000`	Establishes a peering session with a BGP neighbor.

	Command or Action	Purpose
Step 6	**neighbor** *ip-address* **fall-over** **Example:** `Router(config-router-af)# neighbor 10.0.0.1 fall-over`	Configures the BGP peering to use fast session deactivation. • BGP will remove all routes learned through this peer if the session is deactivated.
Step 7	**end** **Example:** `Router(config-router-af)# end`	Exits configuration mode and enters privileged EXEC mode.

Configuring Selective Address Tracking for Fast Session Deactivation

Perform this task to configure selective address tracking for fast session deactivation. The optional **route-map** keyword and *map-name* argument of the **neighbor fall-over** command are used to determine if a peering session with a BGP neighbor should be deactivated (reset) when a route to the BGP peer changes. The route map is evaluated against the new route, and if a deny statement is returned, the peer session is reset.

Note Only **match ip address** and **match source-protocol** commands are supported in the route map. No **set** commands or other **match** commands are supported.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
5. **neighbor** *ip-address* **fall-over** [**route-map** *map-name*]
6. **exit**
7. **ip prefix-list** *list-name* [**seq** *seq-value* **]**{**deny** *network* / *length* | **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*]
8. **route-map** *map-name* [**permit** | **deny**][*sequence-number*]
9. **match ip address prefix-list** *prefix-list-name* [*prefix-list-name*...]
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 5	**neighbor** *ip-address* **fall-over** [**route-map** *map-name*] **Example:** `Router(config-router)# neighbor 192.168.1.2 fall-over route-map CHECK-NBR`	Applies a route map when a route to the BGP changes. • In this example, the route map named CHECK-NBR is applied when the route to neighbor 192.168.1.2 changes.
Step 6	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 7	**ip prefix-list** *list-name* [**seq** *seq-value* **]**{**deny** *network / length* | **permit** *network / length*}[**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list FILTER28 seq 5 permit 0.0.0.0/0 ge 28`	Creates a prefix list for BGP next-hop route filtering. • Selective next-hop route filtering supports prefix- length matching or source-protocol matching on a per-address family basis. • The example creates a prefix list named FILTER28 that permits routes only if the mask length is greater than or equal to 28.
Step 8	**route-map** *map-name* [**permit** | **deny**][*sequence-number*] **Example:** `Router(config)# route-map CHECK-NBR permit 10`	Configures a route map and enters route-map configuration mode. • In this example, a route map named CHECK-NBR is created. If there is an IP address match in the following **match** command, the IP address will be permitted.
Step 9	**match ip address prefix-list** *prefix-list-name* [*prefix-list-name...*] **Example:** `Router(config-route-map)# match ip address prefix-list FILTER28`	Matches the IP addresses in the specified prefix list. • Use the *prefix-list-name* argument to specify the name of a prefix list. The ellipsis means that more than one prefix list can be specified. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 10	**end** **Example:** `Router(config-route-map)# end`	Exits route-map configuration mode and enters privileged EXEC mode.

• What to Do Next, page 228

What to Do Next

The BGP Support for Next-Hop Address Tracking feature improves the response time of BGP to next-hop changes for routes installed in the RIB, which can also improve overall BGP convergence. For information about BGP next-hop address tracking, see the "Configuring Advanced BGP Features" module.

Configuring BFD for BGP IPv6 Neighbors

In Cisco IOS Release 15.1(2)S and later releases, Bidirectional Forwarding Detection (BFD) can be used for BGP neighbors that have an IPv6 address.

Once it has been verified that BFD neighbors are up, the **show bgp ipv6 unicast neighbors**command will indicate that BFD is being used to detect fast fallover on the specified neighbor.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 unicast-routing**
4. **ipv6 cef**
5. **interface** *type number*
6. **ipv6 address** *ipv6-address* / *prefix-length*
7. **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *multiplier-value*
8. **no shutdown**
9. **exit**
10. **router bgp** *autonomous-system-number*
11. **no bgp default ipv4-unicast**
12. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**]
13. **neighbor** *ipv6-address* **remote-as** *autonomous-system-number*
14. **neighbor** *ipv6-address* **fall-over bfd**
15. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Router(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 4	**ipv6 cef** **Example:** `Router(config)# ipv6 cef`	Enables Cisco Express Forwarding for IPv6.

	Command or Action	Purpose
Step 5	**interface** *type number* **Example:** `Router(config)# interface fastethernet 0/1`	Configures an interface type and number.
Step 6	**ipv6 address** *ipv6-address* / *prefix-length* **Example:** `Router(config-if)# ipv6 address 2001:DB8:1:1::1/64`	Configures an IPv6 address and enables IPv6 processing on an interface.
Step 7	**bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *multiplier-value* **Example:** `Router(config-if)# bfd interval 500 min_rx 500 multiplier 3`	Sets the baseline BFD session parameters on an interface.
Step 8	**no shutdown** **Example:** `Router(config-if)# no shutdown`	Restarts an interface.
Step 9	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 10	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.
Step 11	**no bgp default ipv4-unicast** **Example:** `Router(config-router)# no bgp default ipv4-unicast`	Disables the default IPv4 unicast address family for establishing peering sessions. • We recommend configuring this command in the global scope.

	Command or Action	Purpose
Step 12	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] **Example:** `Router(config-router)# address-family ipv6`	Enters address family configuration mode and enables IPv6 addressing.
Step 13	**neighbor** *ipv6-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 2001:DB8:2:1::4 remote-as 45000`	Adds the IP address of the neighbor in the specified autonomous system to the IPv6 BGP neighbor table of the local router.
Step 14	**neighbor** *ipv6-address* **fall-over bfd** **Example:** `Router(config-router)# neighbor 2001:DB8:2:1::4 fall-over bfd`	Enables BGP to monitor the peering session of an IPv6 neighbor using BFD.
Step 15	**end** **Example:** `Router(config-router)# end`	Exits configuration mode and enters privileged EXEC mode.

Configuring a Router to Reestablish a Neighbor Session After the Maximum Prefix Limit Has Been Exceeded

Perform this task to configure the time interval at which a BGP neighbor session is reestablished by a router when the number of prefixes that have been received from a BGP peer has exceeded the maximum prefix limit.

The network operator can configure a router that is running BGP to automatically reestablish a neighbor session that has been brought down because the configured maximum-prefix limit has been exceeded. No intervention from the network operator is required when this feature is enabled.

> **Note** This task attempts to reestablish a disabled BGP neighbor session at the configured time interval that is specified by the network operator. However, the configuration of the restart timer alone cannot change or correct a peer that is sending an excessive number of prefixes. The network operator will need to reconfigure the maximum-prefix limit or reduce the number of prefixes that are sent from the peer. A peer that is configured to send too many prefixes can cause instability in the network, where an excessive number of prefixes are rapidly advertised and withdrawn. In this case, the **warning-only** keyword of the **neighbor maximum-prefix** command can be configured to disable the restart capability, while the network operator corrects the underlying problem.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** {*ip-address* | *peer-group-name*} **maximum-prefix** *maximum* [*threshold*] [**restart** *minutes*] [**warning-only**]
5. **end**
6. **show ip bgp neighbors** *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **maximum-prefix** *maximum* [*threshold*] [**restart** *minutes*] [**warning-only**] **Example:** `Router(config-router)# neighbor 10.4.9.5 maximum-prefix 1000 90 restart 60`	Configures the maximum-prefix limit on a router that is running BGP. • Use the **restart** keyword and *minutes* argument to configure the router to automatically reestablish a neighbor session that has been disabled because the maximum-prefix limit has been exceeded. The configurable range of *minutes*is from 1 to 65535 minutes. • Use the **warning-only** keyword to configure the router to disable the restart capability to allow you to fix a peer that is sending too many prefixes. **Note** If the *minutes*argument is not configured, the disabled session will stay down after the maximum-prefix limit is exceeded. This is the default behavior.
Step 5	**end** **Example:** `Router(config-router)# end`	Exits configuration mode and enters privilaged EXEC mode.
Step 6	**show ip bgp neighbors** *ip-address* **Example:** `Router# show ip bgp neighbors 10.4.9.5`	(Optional) Displays information about the TCP and BGP connections to neighbors. • In this example, the output from this command will display the maximum prefix limit for the specified neighbor and the configured restart timer value. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following example output from the **show ip bgp neighbors** command verifies that a router has been configured to automatically reestablish disabled neighbor sessions. The output shows that the maximum prefix limit for neighbor 10.4.9.5 is set to 1000 prefixes, the restart threshold is set to 90 percent, and the restart interval is set at 60 minutes.

```
Router# show ip bgp neighbors 10.4.9.5

BGP neighbor is 10.4.9.5,  remote AS 101, internal link
  BGP version 4, remote router ID 10.4.9.5
  BGP state = Established, up for 2w2d
  Last read 00:00:14, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                0          0
    Keepalives:         23095      23095
    Route Refresh:          0          0
    Total:              23096      23096
```

```
  Default minimum time between advertisement runs is 5 seconds
 For address family: IPv4 Unicast
  BGP table version 1, neighbor versions 1/0 1/0
  Output queue sizes : 0 self, 0 replicated
  Index 2, Offset 0, Mask 0x4
  Member of update-group 2
                               Sent       Rcvd
  Prefix activity:             ----       ----
    Prefixes Current:             0          0
    Prefixes Total:               0          0
    Implicit Withdraw:            0          0
    Explicit Withdraw:            0          0
    Used as bestpath:           n/a          0
    Used as multipath:          n/a          0
                                 Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Total:                              0          0
!Configured maximum number of prefixes and restart interval information!
  Maximum prefixes allowed 1000
  Threshold for warning message 90%, restart interval 60 min
  Number of NLRIs in the update sent: max 0, min 0
  Connections established 1; dropped 0
  Last reset never
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 10.4.9.21, Local port: 179
Foreign host: 10.4.9.5, Foreign port: 11871
Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)
Event Timers (current time is 0x5296BD2C):
Timer          Starts    Wakeups            Next
Retrans         23098          0             0x0
TimeWait            0          0             0x0
AckHold         23096      22692             0x0
SendWnd             0          0             0x0
KeepAlive           0          0             0x0
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0
iss: 1900546793  snduna: 1900985663  sndnxt: 1900985663     sndwnd:  14959
irs: 2894590641  rcvnxt: 2895029492  rcvwnd:      14978  delrcvwnd:   1406
SRTT: 300 ms, RTTO: 607 ms, RTV: 3 ms, KRTT: 0 ms
minRTT: 0 ms, maxRTT: 316 ms, ACK hold: 200 ms
Flags: passive open, nagle, gen tcbs
Datagrams (max data segment is 1460 bytes):
Rcvd: 46021 (out of order: 0), with data: 23096, total data bytes: 438850
Sent: 46095 (retransmit: 0, fastretransmit: 0), with data: 23097, total data by9
```

- Troubleshooting Tips, page 234

Troubleshooting Tips

Use the **clear ip bgp** command to resets a BGP connection using BGP soft reconfiguration. This command can be used to clear stored prefixes to prevent a router that is running BGP from exceeding the maximum-prefix limit. For more details about using BGP soft reconfiguration, see the "Monitoring and Maintaining Basic BGP" task in the "Configuring a Basic BGP Network" module.

Display of the following error messages can indicate an underlying problem that is causing the neighbor session to become disabled. The network operator should check the values that are configured for the maximum-prefix limit and the configuration of any peers that are sending an excessive number of prefixes. The following sample error messages are similar to the error messages that may be displayed:

```
00:01:14:%BGP-5-ADJCHANGE:neighbor 10.10.10.2 Up
00:01:14:%BGP-4-MAXPFX:No. of unicast prefix received from 10.10.10.2 reaches 5, max 6
00:01:14:%BGP-3-MAXPFXEXCEED:No.of unicast prefix received from 10.10.10.2:7 exceed limit6
00:01:14:%BGP-5-ADJCHANGE:neighbor 10.10.10.2 Down - BGP Notification sent
00:01:14:%BGP-3-NOTIFICATION:sent to neighbor 10.10.10.2 3/1 (update malformed) 0 byte
```

The **bgp dampening** command can be used to configure the dampening of a flapping route or interface when a peer is sending too many prefixes and causing network instability. Use this command only when troubleshooting or tuning a router that is sending an excessive number of prefixes. For more details about BGP route dampening, see the "Configuring Advanced BGP Features" module.

Configuring Dual-AS Peering for Network Migration

Perform this task to configure a BGP peer router to appear to external peers as a member of another autonomous system for the purpose of autonomous system number migration. When the BGP peer is configured with dual autonomous system numbers then the network operator can merge a secondary autonomous system into a primary autonomous system and update the customer configuration during a future service window without disrupting existing peering arrangements.

The **show ip bgp** and **show ip bgp neighbors** commands can be used to verify autonomous system number for entries in the routing table and the status of this feature.

Note

- The BGP Support for Dual AS Configuration for Network AS Migrations feature can be configured for only true eBGP peering sessions. This feature cannot be configured for two peers in different subautonomous systems of a confederation.
- The BGP Support for Dual AS Configuration for Network AS Migrations feature can be configured for individual peering sessions and configurations applied through peer groups and peer templates. If this command is applied to a peer group, the peers cannot be individually customized.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **neighbor** *ip-address* **local-as** [*autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]]
6. **neighbor** *ip-address* **remove-private-as**
7. **end**
8. **show ip bgp** [*network*] [*network-mask*] [**longer-prefixes**] [**prefix-list** *prefix-list-name* | **route-map** *route-map-name*] [**shorter-prefixes** *mask-length*]
9. **show ip bgp neighbors** [*neighbor-address*] [**received-routes** | **routes**| **advertised-routes** | **paths** *regexp* | **dampened-routes** | **received** *prefix-filter*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode, and creates a BGP routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 10.0.0.1 remote-as 45000`	Establishes a peering session with a BGP neighbor.
Step 5	**neighbor** *ip-address* **local-as** [*autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]] **Example:** `Router(config-router)# neighbor 10.0.0.1 local-as 50000 no-prepend replace-as dual-as`	Customizes the AS_PATH attribute for routes received from an eBGP neighbor. • The **replace-as** keyword is used to prepend only the local autonomous system number (as configured with the *ip-address* argument) to the AS_PATH attribute. The autonomous system number from the local BGP routing process is not prepended. • The **dual-as** keyword is used to configure the eBGP neighbor to establish a peering session using the real autonomous-system number (from the local BGP routing process) or by using the autonomous system number configured with the *ip-address* argument (local-as). • The example configures the peering session with the 10.0.0.1 neighbor to accept the real autonomous system number and the local-as number.
Step 6	**neighbor** *ip-address* **remove-private-as** **Example:** `Router(config-router)# neighbor 10.0.0.1 remove-private-as`	(Optional) Removes private autonomous system numbers from outbound routing updates. • This command can be used with the **replace-as** functionality to remove the private autonomous system number and replace it with an external autonomous system number. • Private autonomous system numbers (64512 to 65535) are automatically removed from the AS_PATH attribute when this command is configured.

	Command or Action	Purpose
Step 7	**end** **Example:** `Router(config-router)# end`	Exits configuration mode and enters privileged EXEC mode.
Step 8	**show ip bgp** [*network*] [*network-mask*] [**longer-prefixes**] [**prefix-list** *prefix-list-name* | **route-map** *route-map-name*] [**shorter-prefixes** *mask-length*] **Example:** `Router# show ip bgp`	Displays entries in the BGP routing table. • The output can be used to verify if the real autonomous system number or local-as number is configured.
Step 9	**show ip bgp neighbors** [*neighbor-address*] [**received-routes** | **routes**| **advertised-routes** | **paths** *regexp* | **dampened-routes** | **received** *prefix-filter*] **Example:** `Router# show ip bgp neighbors`	Displays information about TCP and BGP connections to neighbors. • The output will display **local AS**, **no-prepend**, **replace-as**, and **dual-as** with the corresponding autonomous system number when these options are configured.

Configuring the TTL Security Check for BGP Neighbor Sessions

Perform this task to allow BGP to establish or maintain a session only if the TTL value in the IP packet header is equal to or greater than the TTL value configured for the BGP neighbor session.

- To maximize the effectiveness of the BGP Support for TTL Security Check feature, we recommend that you configure it on each participating router. Enabling this feature secures the eBGP session in the incoming direction only and has no effect on outgoing IP packets or the remote router.

Note

- The **neighbor ebgp-multihop** command is not needed when the BGP Support for TTL Security Check feature is configured for a multihop neighbor session and should be disabled before configuring this feature.
- The effectiveness of the BGP Support for TTL Security Check feature is reduced in large-diameter multihop peerings. In the event of a CPU utilization-based attack against a BGP router that is configured for large-diameter peering, you may still need to shut down the affected neighbor sessions to handle the attack.
- This feature is not effective against attacks from a peer that has been compromised inside of the local and remote network. This restriction also includes peers that are on the network segment between the local and remote network.

SUMMARY STEPS

1. **enable**
2. **trace** [*protocol*] *destination*
3. **configure terminal**
4. **router bgp** *autonomous-system-number*
5. **neighbor** *ip-address* **ttl-security hops** *hop-count*
6. **end**
7. **show running-config**
8. **show ip bgp neighbors** [*ip-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**trace** [*protocol*] *destination* **Example:** `Router# trace ip 10.1.1.1`	Discovers the routes of the specified protocol that packets will actually take when traveling to their destination. • Enter the **trace** command to determine the number of hops to the specified peer.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 65000`	Enters router configuration mode, and creates a BGP routing process.

	Command or Action	Purpose
Step 5	**neighbor** *ip-address* **ttl-security hops** *hop-count* **Example:** `Router(config-router)# neighbor 10.1.1.1 ttl-security hops 2`	Configures the maximum number of hops that separate two peers. • The *hop-count* argument is set to the number of hops that separate the local and remote peer. If the expected TTL value in the IP packet header is 254, then the number 1 should be configured for the *hop-count* argument. The range of values is a number from 1 to 254. • When the BGP Support for TTL Security Check feature is enabled, BGP will accept incoming IP packets with a TTL value that is equal to or greater than the expected TTL value. Packets that are not accepted are discarded. • The example configuration sets the expected incoming TTL value to at least 253, which is 255 minus the TTL value of 2, and this is the minimum TTL value expected from the BGP peer. The local router will accept the peering session from the 10.1.1.1 neighbor only if it is one or two hops away.
Step 6	**end** **Example:** `Router(config-router)# end`	Exits configuration mode and enters privileged EXEC mode.
Step 7	**show running-config** **Example:** `Router# show running-config | begin bgp`	(Optional) Displays the contents of the currently running configuration file. • The output of this command displays the configuration of the **neighbor ttl-security** command for each peer under the BGP configuration section of output. That section includes the neighbor address and the configured hop count. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 8	**show ip bgp neighbors** [*ip-address*] **Example:** `Router# show ip bgp neighbors 10.4.9.5`	(Optional) Displays information about the TCP and BGP connections to neighbors. • This command displays "External BGP neighbor may be up to *number* hops away" when the BGP Support for TTL Security Check feature is enabled. The *number* value represents the hop count. It is a number from 1 to 254. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The configuration of the BGP Support for TTL Security Check feature can be verified with the **show running-config** and **show ip bgp neighbors**commands. This feature is configured locally on each peer, so there is no remote configuration to verify.

The following is sample output from the **show running-config** command. The output shows that neighbor 10.1.1.1 is configured to establish or maintain the neighbor session only if the expected TTL count in the incoming IP packet is 253 or 254.

```
Router# show running-config
| begin bgp
```

```
router bgp 65000
 no synchronization
 bgp log-neighbor-changes
 neighbor 10.1.1.1 remote-as 55000
 neighbor 10.1.1.1 ttl-security hops 2
 no auto-summary
.
.
.
```

The following is sample output from the **show ip bgp neighbors** command. The output shows that the local router will accept packets from the 10.1.1.1 neighbor if it is no more than 2 hops away. The configuration of this feature is displayed in the address family section of the output. The relevant line is shown in bold in the output.

```
Router# show ip bgp neighbors 10.1.1.1
BGP neighbor is 10.1.1.1,  remote AS 55000, external link
  BGP version 4, remote router ID 10.2.2.22
  BGP state = Established, up for 00:59:21
  Last read 00:00:21, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  2          2
    Notifications:          0          0
    Updates:                0          0
    Keepalives:           226        227
    Route Refresh:          0          0
    Total:                228        229
  Default minimum time between advertisement runs is 5 seconds
 For address family: IPv4 Unicast
  BGP table version 1, neighbor version 1/0
  Output queue sizes : 0 self, 0 replicated
  Index 1, Offset 0, Mask 0x2
  Member of update-group 1
                                 Sent       Rcvd
  Prefix activity:               ----       ----
    Prefixes Current:               0          0
    Prefixes Total:                 0          0
    Implicit Withdraw:              0          0
    Explicit Withdraw:              0          0
    Used as bestpath:             n/a          0
    Used as multipath:            n/a          0
                                   Outbound    Inbound
  Local Policy Denied Prefixes:    --------    -------
    Total:                                0          0
  Number of NLRIs in the update sent: max 0, min 0
  Connections established 2; dropped 1
  Last reset 00:59:50, due to User reset
  External BGP neighbor may be up to 2 hops away.
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 10.2.2.22, Local port: 179
Foreign host: 10.1.1.1, Foreign port: 11001
Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)
Event Timers (current time is 0xCC28EC):
Timer          Starts    Wakeups            Next
Retrans            63          0             0x0
TimeWait            0          0             0x0
AckHold            62         50             0x0
SendWnd             0          0             0x0
KeepAlive           0          0             0x0
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0
iss:  712702676  snduna:  712703881  sndnxt:  712703881     sndwnd:  15180
irs: 2255946817  rcvnxt: 2255948041  rcvwnd:      15161  delrcvwnd:   1223
SRTT: 300 ms, RTTO: 607 ms, RTV: 3 ms, KRTT: 0 ms
```

```
minRTT: 0 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: passive open, nagle, gen tcbs

Datagrams (max data segment is 1460 bytes):
Rcvd: 76 (out of order: 0), with data: 63, total data bytes: 1223
Sent: 113 (retransmit: 0, fastretransmit: 0), with data: 62, total data bytes: 4
```

Configuring BGP Support for TCP Path MTU Discovery per Session

This section contains the following tasks:

Disabling TCP Path MTU Discovery Globally for All BGP Sessions

Perform this task to disable TCP path MTU discovery for all BGP sessions. TCP path MTU discovery is enabled by default when you configure BGP sessions, but we recommend that you enter the **show ip bgp neighbors** command to ensure that TCP path MTU discovery is enabled.

This task assumes that you have previously configured BGP neighbors with active TCP connections.

SUMMARY STEPS

1. **enable**
2. **show ip bgp neighbors** [*ip-address*]
3. **configure terminal**
4. **router bgp** *autonomous-system-number*
5. **no bgp transport path-mtu-discovery**
6. **end**
7. **show ip bgp neighbors** [*ip-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**show ip bgp neighbors** [*ip-address*] **Example:** `Router# show ip bgp neighbors`	(Optional) Displays information about the TCP and BGP connections to neighbors. • Use this command to determine whether BGP neighbors have TCP path MTU discovery enabled. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 50000`	Enters router configuration mode to create or configure a BGP routing process.
Step 5	**no bgp transport path-mtu-discovery** **Example:** `Router(config-router)# no bgp transport path-mtu-discovery`	Disables TCP path MTU discovery for all BGP sessions.
Step 6	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.
Step 7	**show ip bgp neighbors** [*ip-address*] **Example:** `Router# show ip bgp neighbors`	(Optional) Displays information about the TCP and BGP connections to neighbors. • In this example, the output from this command will not display that any neighbors have TCP path MTU enabled. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following sample output from the **show ip bgp neighbors** command shows that TCP path MTU discovery is enabled for BGP neighbors. Two entries in the output--Transport(tcp) path-mtu-discovery is enabled and path mtu capable--show that TCP path MTU discovery is enabled.

```
Router# show ip bgp neighbors
BGP neighbor is 172.16.1.2,  remote AS 45000, internal link
  BGP version 4, remote router ID 172.16.1.99
.
.
.
 For address family: IPv4 Unicast
  BGP table version 5, neighbor version 5/0
.
.
.
  Address tracking is enabled, the RIB does have a route to 172.16.1.2
  Address tracking requires at least a /24 route to the peer
  Connections established 3; dropped 2
  Last reset 00:00:35, due to Router ID changed
  Transport(tcp) path-mtu-discovery is enabled
.
.
.
SRTT: 146 ms, RTTO: 1283 ms, RTV: 1137 ms, KRTT: 0 ms
minRTT: 8 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: higher precedence, retransmission timeout, nagle, path mtu capable
```

The following is sample output from the **show ip bgp neighbors** command after the **no bgp transport path-mtu-discovery** command has been entered. Note that the path mtu entries are missing.

```
Router# show ip bgp neighbors
BGP neighbor is 172.16.1.2,  remote AS 45000, internal link
  BGP version 4, remote router ID 172.16.1.99
.
.
.
 For address family: IPv4 Unicast
  BGP table version 5, neighbor version 5/0
.
.
.
  Address tracking is enabled, the RIB does have a route to 172.16.1.2
  Address tracking requires at least a /24 route to the peer
  Connections established 3; dropped 2
  Last reset 00:00:35, due to Router ID changed
.
.
.
SRTT: 146 ms, RTTO: 1283 ms, RTV: 1137 ms, KRTT: 0 ms
minRTT: 8 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: higher precedence, retransmission timeout, nagle
```

Disabling TCP Path MTU Discovery for a Single BGP Neighbor

Perform this task to establish a peering session with an internal BGP (iBGP) neighbor and then disable TCP path MTU discovery for the BGP neighbor session. The **neighbor transport** command can be used in router configuration or address family configuration mode.

This task assumes that you know that TCP path MTU discovery is enabled by default for all your BGP neighbors.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
5. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
6. **neighbor** {*ip-address*| *peer-group-name*} **activate**
7. **no neighbor** {*ip-address*| *peer-group-name*} **transport**{**connection-mode** | **path-mtu-discovery**}
8. **end**
9. **show ip bgp neighbors**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example creates an IPv4 unicast address family session.

	Command or Action	Purpose
Step 5	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 6	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 172.16.1.1 activate`	Activates the neighbor under the IPv4 address family.
Step 7	**no neighbor** {*ip-address*| *peer-group-name*} **transport**{**connection-mode** | **path-mtu-discovery**} **Example:** `Router(config-router-af)# no neighbor 172.16.1.1 transport path-mtu-discovery`	Disables TCP path MTU discovery for a single BGP neighbor. • In this example, TCP path MTU discovery is disabled for the neighbor at 172.16.1.1.
Step 8	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 9	**show ip bgp neighbors** **Example:** `Router# show ip bgp neighbors`	(Optional) Displays information about the TCP and BGP connections to neighbors. • In this example, the output from this command will not display that the neighbor has TCP path MTU discovery enabled. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following sample output shows that TCP path MTU discovery has been disabled for BGP neighbor 172.16.1.1 but that it is still enabled for BGP neighbor 192.168.2.2. Two entries in the output--Transport(tcp) path-mtu-discovery is enabled and path mtu capable--show that TCP path MTU discovery is enabled.

```
Router# show ip bgp neighbors
BGP neighbor is 172.16.1.1,  remote AS 45000, internal link
  BGP version 4, remote router ID 172.17.1.99
```

```
.
.
.
  Address tracking is enabled, the RIB does have a route to 172.16.1.1
  Address tracking requires at least a /24 route to the peer
  Connections established 1; dropped 0
  Last reset never
.
.
.
SRTT: 165 ms, RTTO: 1172 ms, RTV: 1007 ms, KRTT: 0 ms
minRTT: 20 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: higher precedence, retransmission timeout, nagle
.
.
.
BGP neighbor is 192.168.2.2,  remote AS 50000, external link
  BGP version 4, remote router ID 10.2.2.99
.
.
.
 For address family: IPv4 Unicast
  BGP table version 4, neighbor version 4/0
.
.
.
  Address tracking is enabled, the RIB does have a route to 192.168.2.2
  Address tracking requires at least a /24 route to the peer
  Connections established 2; dropped 1
  Last reset 00:05:11, due to User reset
  Transport(tcp) path-mtu-discovery is enabled
.
.
.
SRTT: 210 ms, RTTO: 904 ms, RTV: 694 ms, KRTT: 0 ms
minRTT: 20 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: higher precedence, retransmission timeout, nagle, path mtu capable
```

Enabling TCP Path MTU Discovery Globally for All BGP Sessions

Perform this task to enable TCP path MTU discovery for all BGP sessions. TCP path MTU discovery is enabled by default when you configure BGP sessions, but if the BGP Support for TCP Path MTU Discovery per Session feature has been disabled, you can use this task to reenable it. To verify that TCP path MTU discovery is enabled, use the **show ip bgp neighbors** command.

This task assumes that you have previously configured BGP neighbors with active TCP connections.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp transport path-mtu-discovery**
5. **end**
6. **show ip bgp neighbors**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**bgp transport path-mtu-discovery** **Example:** `Router(config-router)# bgp transport path-mtu-discovery`	Enables TCP path MTU discovery for all BGP sessions.
Step 5	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.
Step 6	**show ip bgp neighbors** **Example:** `Router# show ip bgp neighbors`	(Optional) Displays information about the TCP and BGP connections to neighbors. • In this example, the output from this command will show that all neighbors have TCP path MTU discovery enabled. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following sample output from the **show ip bgp neighbors** command shows that TCP path MTU discovery is enabled for BGP neighbors. Two entries in the output--Transport(tcp) path-mtu-discovery is enabled and path mtu capable--show that TCP path MTU discovery is enabled.

```
Router# show ip bgp neighbors
```

```
BGP neighbor is 172.16.1.2,  remote AS 45000, internal link
  BGP version 4, remote router ID 172.16.1.99
.
.
.
 For address family: IPv4 Unicast
  BGP table version 5, neighbor version 5/0
.
.
.
  Address tracking is enabled, the RIB does have a route to 172.16.1.2
  Address tracking requires at least a /24 route to the peer
  Connections established 3; dropped 2
  Last reset 00:00:35, due to Router ID changed
  Transport(tcp) path-mtu-discovery is enabled
.
.
.
SRTT: 146 ms, RTTO: 1283 ms, RTV: 1137 ms, KRTT: 0 ms
minRTT: 8 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: higher precedence, retransmission timeout, nagle, path mtu capable
```

Enabling TCP Path MTU Discovery for a Single BGP Neighbor

Perform this task to establish a peering session with an eBGP neighbor and then enable TCP path MTU discovery for the BGP neighbor session. The **neighbor transport** command can be used in router configuration or address family configuration mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
5. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
6. **neighbor** {*ip-address*| *peer-group-name*} **activate**
7. **neighbor** {*ip-address*| *peer-group-name*} **transport**{**connection-mode** | **path-mtu-discovery**}
8. **end**
9. **show ip bgp neighbors** [*ip-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example creates an IPv4 unicast address family session.
Step 5	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.2.2 remote-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 6	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.2.2 activate`	Activates the neighbor under the IPv4 address family.
Step 7	**neighbor** {*ip-address*| *peer-group-name*} **transport**{**connection-mode** | **path-mtu-discovery**} **Example:** `Router(config-router-af)# neighbor 192.168.2.2 transport path-mtu-discovery`	Enables TCP path MTU discovery for a single BGP neighbor.
Step 8	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 9	**show ip bgp neighbors** [*ip-address*] **Example:** `Router# show ip bgp neighbors 192.168.2.2`	(Optional) Displays information about the TCP and BGP connections to neighbors. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following sample output from the **show ip bgp neighbors** command shows that TCP path MTU discovery is enabled for the BGP neighbor at 192.168.2.2. Two entries in the output--Transport(tcp) path-mtu-discovery is enabled and path-mtu capable--show that TCP path MTU discovery is enabled.

```
Router# show ip bgp neighbors 192.168.2.2
BGP neighbor is 192.168.2.2,  remote AS 50000, external link
  BGP version 4, remote router ID 10.2.2.99
.
.
.
 For address family: IPv4 Unicast
  BGP table version 4, neighbor version 4/0
.
.
.
  Address tracking is enabled, the RIB does have a route to 192.168.2.2
  Address tracking requires at least a /24 route to the peer
  Connections established 2; dropped 1
  Last reset 00:05:11, due to User reset
  Transport(tcp) path-mtu-discovery is enabled
.
.
.
SRTT: 210 ms, RTTO: 904 ms, RTV: 694 ms, KRTT: 0 ms
minRTT: 20 ms, maxRTT: 300 ms, ACK hold: 200 ms
Flags: higher precedence, retransmission timeout, nagle, path mtu capable
```

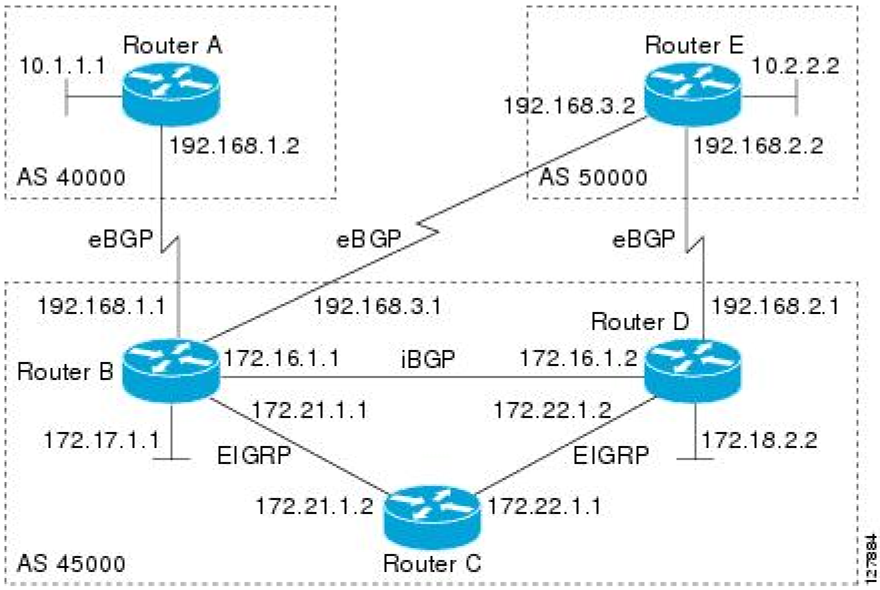
Implementing BGP Dynamic Neighbors Using Subnet Ranges

In Cisco IOS Release 12.2(33)SXH, support for BGP dynamic neighbors was introduced. Perform this task to implement the dynamic creation of BGP neighbors using subnet ranges.

In this task, a BGP peer group is created on Router B in the figure below, a global limit is set on the number of dynamic BGP neighbors, and a subnet range is associated with a peer group. Configuring the subnet range enables the dynamic BGP neighbor process. The peer group is added to the BGP neighbor table of the local router, and an alternate autonomous system number is also configured. The peer group is activated under the IPv4 address family.

The next step is to move to another router--Router E in the figure below--where a BGP session is started and the neighbor router, Router B, is configured as a remote BGP peer. The peering configuration opens a TCP session and triggers Router B to create a dynamic BGP neighbor because the IP address that starts the TCP session (192.168.3.2) is within the configured subnet range for dynamic BGP peers. The task moves back to the first router, Router B, to run three **show** commands that have been modified to display dynamic BGP peer information.

Figure 23 BGP Dynamic Neighbor Topology



This task requires Cisco IOS Release 12.2(33)SXH, or a later release, to be running.

Note This task supports only IPv4 BGP peering.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp log-neighbor-changes**
5. **neighbor** *peer-group-name* **peer-group**
6. **bgp listen** [**limit** *max-number*]
7. **bgp listen** [**limit** *max-number* | **range** *network* / *length* **peer-group** *peer-group-name*]
8. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **ebgp-multihop [** *ttl*]
9. **neighbor** *peer-group-name* **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number...*]
10. **address-family ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*]]
11. **neighbor** {*ip-address*| *peer-group-name*} **activate**
12. **end**
13. Move to another router that has an interface within the subnet range for the BGP peer group configured in this task.
14. **enable**
15. **configure terminal**
16. **router bgp** *autonomous-system-number*
17. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*[**alternate-as** *autonomous-system-number...*]
18. Return to the first router.
19. **show ip bgp summary**
20. **show ip bgp peer-group** [*peer-group-name*] [**summary**]
21. **show ip bgp neighbors** [*ip-address*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `RouterB> enable`	Enables privileged EXEC mode. • Enter your password if prompted. • The configuration is entered on router B.
Step 2	**configure terminal** **Example:** `RouterB# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `RouterB(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**bgp log-neighbor-changes** **Example:** `RouterB(config-router)# bgp log-neighbor-changes`	(Optional) Enables logging of BGP neighbor status changes (up or down) and neighbor resets. • Use this command for troubleshooting network connectivity problems and measuring network stability. Unexpected neighbor resets might indicate high error rates or high packet loss in the network and should be investigated.
Step 5	**neighbor** *peer-group-name* **peer-group** **Example:** `RouterB(config-router)# neighbor group192 peer-group`	Creates a BGP peer group. • In this example, a peer group named group192 is created. This group will be used as a listen range group.
Step 6	**bgp listen** [**limit** *max-number*] **Example:** `RouterB(config-router)# bgp listen limit 200`	Sets a global limit of BGP dynamic subnet range neighbors. • Use the optional **limit** keyword and *max-number* argument to define the maximum number of BGP dynamic subnet range neighbors that can be created. **Note** Only the syntax applicable to this task is used in this example. For the complete syntax, see Step 7.
Step 7	**bgp listen** [**limit** *max-number* | **range** *network* / *length* **peer-group** *peer-group-name*] **Example:** `RouterB(config-router)# bgp listen range 192.168.0.0/16 peer-group group192`	Associates a subnet range with a BGP peer group and activates the BGP dynamic neighbors feature. • Use the optional **limit** keyword and *max-number* argument to define the maximum number of BGP dynamic neighbors that can be created. • Use the optional **range** keyword and *network* / *length* argument to define a prefix range to be associated with the specified peer group. • In this example, the prefix range 192.168.0.0/16 is associated with the listen range group named group192.
Step 8	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **ebgp-multihop [** *ttl*] **Example:** `RouterB(config-router)# neighbor group192 ebgp-multihop 255`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected.

	Command or Action	Purpose
Step 9	**neighbor** *peer-group-name* **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number*...] **Example:** `RouterB(config-router)# neighbor group192 remote-as 40000 alternate-as 50000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • Use the optional **alternate-as** keyword and *autonomous-system-number* argument to identify up to five alternate autonomous system numbers for listen range neighbors. • In this example, the peer group named group192 is configured with two possible autonomous system numbers. **Note** The **alternate-as** keyword is used only with the listen range peer groups, not with individual BGP neighbors.
Step 10	**address-family ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*]] **Example:** `RouterB(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations.
Step 11	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `RouterB(config-router-af)# neighbor group192 activate`	Activates the neighbor or listen range peer group for the configured address family. • In this example, the neighbor 172.16.1.1 is activated for the IPv4 address family. **Note** Usually BGP peer groups cannot be activated using this command, but the listen range peer groups are a special case.
Step 12	**end** **Example:** `RouterB(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 13	Move to another router that has an interface within the subnet range for the BGP peer group configured in this task.	--
Step 14	**enable** **Example:** `RouterE> enable`	Enables privileged EXEC mode. • Enter your password if prompted. • The configuration is entered on Router E.

	Command or Action	Purpose
Step 15	**configure terminal** **Example:** `RouterE# configure terminal`	Enters global configuration mode.
Step 16	**router bgp** *autonomous-system-number* **Example:** `RouterE(config)# router bgp 50000`	Enters router configuration mode for the specified routing process.
Step 17	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*[**alternate-as** *autonomous-system-number...*] **Example:** `RouterE(config-router)# neighbor 192.168.3.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • In this example, the interface (192.168.3.2 in the figure above) at Router E is with the subnet range set for the BGP listen range group, group192. When TCP opens a session to peer to Router B, Router B creates this peer dynamically.
Step 18	Return to the first router.	--
Step 19	**show ip bgp summary** **Example:** `RouterB# show ip bgp summary`	(Optional) Displays the BGP path, prefix, and attribute information for all connections to BGP neighbors. • In this step, the configuration has returned to Router B.
Step 20	**show ip bgp peer-group** [*peer-group-name*] [**summary**] **Example:** `RouterB# show ip bgp peer-group group192`	(Optional) Displays information about BGP peer groups.

	Command or Action	Purpose
Step 21	**show ip bgp neighbors** [*ip-address*] **Example:** `RouterB# show ip bgp neighbors 192.168.3.2`	(Optional) Displays information about BGP and TCP connections to neighbors. • In this example, information is displayed about the dynamically created neighbor at 192.168.3.2. The IP address of this BGP neighbor can be found in the output of either the **show ip bgp summary** or the **show ip bgp peer-group** command. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following output examples were taken from Router B in the figure above after the appropriate configuration steps in this task were completed on both Router B and Router E.

The following output from the **show ip bgp summary** command shows that the BGP neighbor 192.168.3.2 was dynamically created and is a member of the listen range group, group192. The output also shows that the IP prefix range of 192.168.0.0/16 is defined for the listen range named group192.

```
Router# show ip bgp summary
BGP router identifier 192.168.3.1, local AS number 45000
BGP table version is 1, main routing table version 1
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
*192.168.3.2    4 50000       2       2        0    0    0 00:00:37        0
* Dynamically created based on a listen range command
Dynamically created neighbors: 1/(200 max), Subnet ranges: 1
BGP peergroup group192 listen range group members:
  192.168.0.0/16
```

The following output from the **show ip bgp peer-group** command shows information about the listen range group, group192 that was configured in this task:

```
Router# show ip bgp peer-group group192
BGP peer-group is group192,  remote AS 40000
  BGP peergroup group192 listen range group members:
  192.168.0.0/16
  BGP version 4
  Default minimum time between advertisement runs is 30 seconds
 For address family: IPv4 Unicast
  BGP neighbor is group192, peer-group external, members:
  *192.168.3.2
  Index 0, Offset 0, Mask 0x0
  Update messages formatted 0, replicated 0
  Number of NLRIs in the update sent: max 0, min 0
```

The following sample output from the **show ip bgp neighbors** command shows that the neighbor 192.168.3.2 is a member of the peer group, group192, and belongs to the subnet range group 192.168.0.0/16, which shows that this peer was dynamically created:

```
Router# show ip bgp neighbors 192.168.3.2
BGP neighbor is *192.168.3.2,  remote AS 50000, external link
 Member of peer-group group192 for session parameters
 Belongs to the subnet range group: 192.168.0.0/16
  BGP version 4, remote router ID 192.168.3.2
  BGP state = Established, up for 00:06:35
  Last read 00:00:33, last write 00:00:25, hold time is 180, keepalive intervals
  Neighbor capabilities:
    Route refresh: advertised and received(new)
```

```
      Address family IPv4 Unicast: advertised and received
    Message statistics:
      InQ depth is 0
      OutQ depth is 0

                           Sent       Rcvd
      Opens:                  1          1
      Notifications:          0          0
      Updates:                0          0
      Keepalives:             7          7
      Route Refresh:          0          0
      Total:                  8          8
    Default minimum time between advertisement runs is 30 seconds
  For address family: IPv4 Unicast
    BGP table version 1, neighbor version 1/0
    Output queue size : 0
    Index 1, Offset 0, Mask 0x2
    1 update-group member
    group192 peer-group member
.
.
.
```

Configuration Examples for BGP Neighbor Session Options

Example Configuring Fast Session Deactivation for a BGP Neighbor

In the following example, the BGP routing process is configured on Router A and Router B to monitor and use fast peering session deactivation for the neighbor session between the two routers. Although fast peering session deactivation is not required at both routers in the neighbor session, it will help the BGP networks in both autonomous systems to converge faster if the neighbor session is deactivated.

Router A

```
router bgp 40000
 neighbor 192.168.1.1 remote-as 45000
 neighbor 192.168.1.1 fall-over
 end
```

Router B

```
router bgp 45000
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.1.2 fall-over
 end
```

Example Configuring Selective Address Tracking for Fast Session Deactivation

The following example shows how to configure the BGP peering session to be reset if a route with a prefix of /28 or a more specific route to a peer destination is no longer available:

```
router bgp 45000
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.1.2 fall-over route-map CHECK-NBR
 exit
ip prefix-list FILTER28 seq 5 permit 0.0.0.0/0 ge 28
route-map CHECK-NBR permit 10
 match ip address prefix-list FILTER28
 end
```

Example Configuring BFD for a BGP IPv6 Neighbor

The following example configures FastEthernet interface 0/1 with the IPv6 address 2001:DB8:4:1::1. Bidirectional Forwarding Detection (BFD) is configured for the BGP neighbor at 2001:DB8:5:1::2. BFD will track forwarding path failure of the BGP neighbor and provide faster reconvergence time for BGP after a forwarding path failure.

```
ipv6 unicast-routing
ipv6 cef
interface fastethernet 0/1
 ipv6 address 2001:DB8:4:1::1/64
 bfd interval 500 min_rx 500 multiplier 3
 no shutdown
 exit
router bgp 65000
 no bgp default ipv4-unicast
 address-family ipv6 unicast
 neighbor 2001:DB8:5:1::2 remote-as 65001
 neighbor 2001:DB8:5:1::2 fall-over bfd
 end
```

Example Restart Session After Maximum Number of Prefixes From Neighbor Reached

The following example sets the maximum number of prefixes allowed from the neighbor at 192.168.6.6 to 2000 and configures the router to reestablish a peering session after 30 minutes if one has been disabled:

```
router bgp 101
 network 172.16.0.0
 neighbor 192.168.6.6 maximum-prefix 2000 restart 30
```

Examples Configuring Dual-AS Peering for Network Migration

The following examples show how to configure and verify this feature:

- Example Dual-AS Configuration, page 259
- Example Dual-AS Confederation Configuration, page 259
- Example Replace-AS Configuration, page 260

Example Dual-AS Configuration

The following examples shows how this feature is used to merge two autonomous systems without interrupting peering arrangements with the customer network. The **neighbor local-as** command is configured to allow Router 1 to maintain peering sessions through autonomous system 40000 and autonomous system 45000. Router 2 is a customer router that runs a BGP routing process in autonomous system 50000 and is configured to peer with autonomous-system 45000.

Router 1 in Autonomous System 40000 (Provider Network)

```
interface Serial3/0
 ip address 10.3.3.11 255.255.255.0
!
router bgp 40000
 no synchronization
 bgp router-id 10.0.0.11
 neighbor 10.3.3.33 remote-as 50000
 neighbor 10.3.3.33 local-as 45000 no-prepend replace-as dual-as
```

Router 1 in Autonomous System 45000 (Provider Network)

```
interface Serial3/0
 ip address 10.3.3.11 255.255.255.0
!
router bgp 45000
 bgp router-id 10.0.0.11
 neighbor 10.3.3.33 remote-as 50000
```

Router 2 in Autonomous System 50000 (Customer Network)

```
interface Serial3/0
 ip address 10.3.3.33 255.255.255.0
!
router bgp 50000
 bgp router-id 10.0.0.3
 neighbor 10.3.3.11 remote-as 45000
```

After the transition is complete, the configuration on router 50000 can be updated to peer with autonomous system 40000 during a normal maintenance window or during other scheduled downtime:

```
 neighbor 10.3.3.11 remote-as 100
```

Example Dual-AS Confederation Configuration

The following example can be used in place of the Router 1 configuration in the "Example: Dual-AS Configuration" example . The only difference between these configurations is that Router 1 is configured to be part of a confederation.

```
interface Serial3/0/0
 ip address 10.3.3.11 255.255.255.0
!
router bgp 65534
 no synchronization
 bgp confederation identifier 100
 bgp router-id 10.0.0.11
 neighbor 10.3.3.33 remote-as 50000
 neighbor 10.3.3.33 local-as 45000 no-prepend replace-as dual-as
```

Example Replace-AS Configuration

The following example strips private autonomous system 64512 from outbound routing updates for the 10.3.3.33 neighbor and replaces it with autonomous system 50000:

```
router bgp 64512
 neighbor 10.3.3.33 local-as 50000 no-prepend replace-as
```

Example Configuring the TTL-Security Check

The example configurations in this section show how to configure the BGP Support for TTL Security Check feature.

The following example uses the **trace** command to determine the hop count to an eBGP peer. The hop count number is displayed in the output for each networking device that IP packets traverse to reach the specified neighbor. In the following example, the hop count for the 10.1.1.1 neighbor is 1.

```
Router# trace ip 10.1.1.1
Type escape sequence to abort.
Tracing the route to 10.1.1.1
  1 10.1.1.1 0 msec *  0 msec
```

The following example sets the hop count to 2 for the 10.1.1.1 neighbor. Because the hop-count argument is set to 2, BGP will accept only IP packets with a TTL count in the header that is equal to or greater than 253.

```
Router(config-router)# neighbor 10.1.1.1 ttl-security hops 2
```

Examples Configuring BGP Support for TCP Path MTU Discovery per Session

This section contains the following configuration examples:

- Example Disabling TCP Path MTU Discovery Globally for All BGP Sessions, page 260
- Example Disabling TCP Path MTU Discovery for a Single BGP Neighbor, page 260
- Example Enabling TCP Path MTU Discovery Globally for All BGP Sessions, page 261
- Example Enabling TCP Path MTU Discovery for a Single BGP Neighbor, page 261

Example Disabling TCP Path MTU Discovery Globally for All BGP Sessions

The following example shows how to disable TCP path MTU discovery for all BGP neighbor sessions. Use the **show ip bgp neighbors** command to verify that TCP path MTU discovery has been disabled.

```
enable
configure terminal
 router bgp 45000
  no bgp transport path-mtu-discovery
  end
show ip bgp neighbors
```

Example Disabling TCP Path MTU Discovery for a Single BGP Neighbor

The following example shows how to disable TCP path MTU discovery for an eBGP neighbor at 192.168.2.2:

```
enable
configure terminal
 router bgp 45000
  neighbor 192.168.2.2 remote-as 50000
  neighbor 192.168.2.2 activate
  no neighbor 192.168.2.2 transport path-mtu-discovery
  end
show ip bgp neighbors 192.168.2.2
```

Example Enabling TCP Path MTU Discovery Globally for All BGP Sessions

The following example shows how to enable TCP path MTU discovery for all BGP neighbor sessions. Use the **show ip bgp neighbors** command to verify that TCP path MTU discovery has been enabled.

```
enable
configure terminal
 router bgp 45000
  bgp transport path-mtu-discovery
  end
show ip bgp neighbors
```

Example Enabling TCP Path MTU Discovery for a Single BGP Neighbor

The following example shows how to enable TCP path MTU discovery for an eBGP neighbor at 192.168.2.2. Use the **show ip bgp neighbors** command to verify that TCP path MTU discovery has been enabled.

```
enable
configure terminal
 router bgp 45000
  neighbor 192.168.2.2 remote-as 50000
  neighbor 192.168.2.2 activate
  neighbor 192.168.2.2 transport path-mtu-discovery
  end
show ip bgp neighbors 192.168.2.2
```

Example Implementing BGP Dynamic Neighbors Using Subnet Ranges

In Cisco IOS Release 12.2(33)SXH, support for BGP dynamic neighbors was introduced. The following example configurations show how to implement BGP dynamic neighbors using subnet ranges.

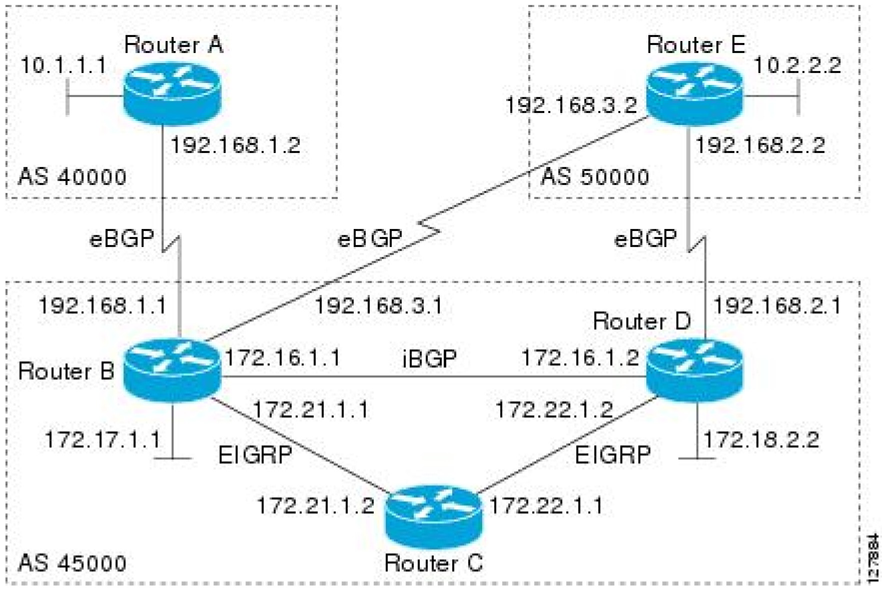
In the following example, two BGP peer groups are created on Router B in the figure below, a global limit is set on the number of dynamic BGP neighbors, and a subnet range is associated with a peer group. Configuring the subnet range enables the dynamic BGP neighbor process. The peer groups are added to the BGP neighbor table of the local router, and an alternate autonomous system number is also configured for one of the peer groups, group192. The subnet range peer groups and a standard BGP peer are then activated under the IPv4 address family.

The configuration moves to another router--Router A in the figure below--where a BGP session is started and the neighbor router, Router B, is configured as a remote BGP peer. The peering configuration opens a TCP session and triggers Router B to create a dynamic BGP neighbor because the IP address that starts the TCP session (192.168.1.2) is within the configured subnet range for dynamic BGP peers.

A third router--Router E in the figure below--also starts a BGP peering session with Router B. Router E is in the autonomous system 50000, which is the configured alternate autonomous system. Router B responds to the resulting TCP session by creating another dynamic BGP peer.

This example concludes with the output of the **show ip bgp summary** command entered on Router B.

Figure 24 BGP Dynamic Neighbor Topology



Router B

```
enable
configure terminal
router bgp 45000
 bgp log-neighbor-changes
 bgp listen limit 200
 bgp listen range 172.21.0.0/16 peer-group group172
 bgp listen range 192.168.0.0/16 peer-group group192
 neighbor group172 peer-group
 neighbor group172 remote-as 45000
 neighbor group192 peer-group
 neighbor group192 remote-as 40000 alternate-as 50000
 neighbor 172.16.1.2 remote-as 45000
 address-family ipv4 unicast
 neighbor group172 activate
 neighbor group192 activate
 neighbor 172.16.1.2 activate
 end
```

Router A

```
enable
configure terminal
router bgp 40000
 neighbor 192.168.1.1 remote-as 45000
 exit
```

Router E

```
enable
configure terminal
router bgp 50000
 neighbor 192.168.3.1 remote-as 45000
 exit
```

After both Router A and Router E are configured, the **show ip bgp summary** command is run on Router B. The output displays the regular BGP neighbor, 172.16.1.2, and the two BGP neighbors that were created dynamically when Router A and Router E initiated TCP sessions for BGP peering to Router B. The output also shows information about the configured listen range subnet groups.

```
BGP router identifier 192.168.3.1, local AS number 45000
BGP table version is 1, main routing table version 1
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.16.1.2      4 45000      15      15        1    0    0 00:12:20        0
*192.168.1.2    4 40000       3       3        1    0    0 00:00:37        0
*192.168.3.2    4 50000       6       6        1    0    0 00:04:36        0
* Dynamically created based on a listen range command
Dynamically created neighbors: 2/(200 max), Subnet ranges: 2
BGP peergroup group172 listen range group members:
  172.21.0.0/16
BGP peergroup group192 listen range group members:
  192.168.0.0/16
```

Where to Go Next

- If you want to connect to an external service provider and use other external BGP features, see the "Connecting to a Service Provider Using External BGP" module.
- If you want to configure some internal BGP features, see the "Configuring Internal BGP Features" module.
- If you want to configure some advanced BGP features including BGP next-hop address tracking and route dampening, see the "Configuring Advanced BGP Features" module.

Additional References

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
Overview of Cisco BGP conceptual information with links to all the individual BGP modules	"Cisco BGP Overview" module
Conceptual and configuration details for basic BGP tasks	"Configuring a Basic BGP Network" module
Conceptual and configuration details for advanced BGP tasks	"Configuring Advanced BGP Features" module
Cisco IOS master command list, all releases	Cisco IOS Master Command List, All Releases
Bidirectional Forwarding Detection configuration tasks	*Cisco IOS XE IP Routing: BFD Configuration Guide*

Standards

Standard	Title
MDT SAFI	MDT SAFI

MIBs

MIB	MIBs Link
CISCO-BGP4-MIB	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1191	Path MTU Discovery
RFC 1771	*A Border Gateway Protocol 4 (BGP-4)*
RFC 1772	*Application of the Border Gateway Protocol in the Internet*
RFC 1773	*Experience with the BGP Protocol*
RFC 1774	*BGP-4 Protocol Analysis*
RFC 1930	*Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 2918	*Route Refresh Capability for BGP-4*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring BGP Neighbor Session Options

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 17 Feature Information for Configuring BGP Neighbor Session Options Features

Feature Name	Releases	Feature Information
BGP Dynamic Neighbors	12.2(33)SXH 15.1(2)T 15.0(1)S	BGP dynamic neighbor support allows BGP peering to a group of remote neighbors that are defined by a range of IP addresses. Each range can be configured as a subnet IP address. BGP dynamic neighbors are configured using a range of IP addresses and BGP peer groups. After a subnet range is configured for a BGP peer group and a TCP session is initiated for an IP address in the subnet range, a new BGP neighbor is dynamically created as a member of that group. The new BGP neighbor will inherit any configuration for the peer group. The output for three **show** commands has been updated to display information about dynamic neighbors. The following commands were introduced or modified by this feature: **bgp listen**, **debug ip bgp range**, **neighbor remote-as**, **show ip bgp neighbors**, **show ip bgp peer-group**, **show ip bgp summary**.

Feature Name	Releases	Feature Information
BGP Restart Session After Max-Prefix Limit	12.0(22)S 12.2(15)T 12.2(18)S 15.0(1)S	The BGP Restart Session After Max-Prefix Limit feature enhanced the capabilities of the **neighbor maximum-prefix** command with the introduction of the **restart** keyword. This enhancement allows the network operator to configure the time interval at which a peering session is reestablished by a router when the number of prefixes that have been received from a peer has exceeded the maximum prefix limit. The following commands were modified by this release: **neighbor maximum-prefix**, **show ip bgp neighbors**.
BGP Selective Address Tracking	12.4(4)T 12.2(31)SB 12.2(33)SRB	The BGP Selective Address Tracking feature introduced the use of a route map for next-hop route filtering and fast session deactivation. Selective next-hop filtering uses a route map to selectively define routes to help resolve the BGP next hop, or a route map can be used to determine if a peering session with a BGP neighbor should be reset when a route to the BGP peer changes. The following commands were modified by this feature: **bgp nexthop**, **neighbor fall-over**.

Feature Name	Releases	Feature Information
BGP Support for 4-Byte ASN	12.0(32)S12 12.0(32)SY8 12.0(33)S3 12.2(33)SRE 12.2(33)XNE 12.2(33)SXI1 12.4(24)T 15.0(1)S	The BGP Support for 4-Byte ASN feature introduced support for 4-byte autonomous system numbers. In Cisco IOS Release 12.0(32)SY8, 12.0(33)S3, 12.2(33)SRE, 12.2(33)XNE, and 12.2(33)SXI1, the Cisco implementation of 4-byte autonomous system numbers uses asplain as the default regular expression match and output display format for autonomous system numbers, but you can configure 4-byte autonomous system numbers in both the asplain format and the asdot format as described in RFC 5396. To change the default regular expression match and output display of 4-byte autonomous system numbers to asdot format, use the **bgp asnotation dot** command. In Cisco IOS Release 12.0(32)S12, and 12.4(24)T, the Cisco implementation of 4-byte autonomous system numbers uses asdot as the only configuration format, regular expression match, and output display, with no asplain support. The following commands were introduced or modified by this feature: **bgp asnotation dot**, **bgp confederation identifier**, **bgp confederation peers**, all **clear ip bgp** commands that configure an autonomous system number, **ip as-path access-list**, **ip extcommunity-list**, **match source-protocol**, **neighbor local-as**, **neighbor remote-as**, **neighbor soo**, **redistribute** (IP), **router bgp**, **route-target**, **set as-path**, **set extcommunity**, **set origin**, **soo**, all **show ip bgp** commands that display an

Feature Name	Releases	Feature Information
		autonomous system number, and **show ip extcommunity-list**.
BGP Support for Dual AS Configuration for Network AS Migrations	12.0(27)S 12.2(25)S 12.3(11)T 12.2(33)SRA 12.2(33)SXH 15.0(1)S	The BGP Support for Dual AS Configuration for Network AS Migrations feature extended the functionality of the BGP Local-AS feature by providing additional autonomous system path customization configuration options. The configuration of this feature is transparent to customer peering sessions, allowing the provider to merge two autonomous systems without interrupting customer peering arrangements. Customer peering sessions can later be updated during a maintenance window or during other scheduled downtime. The following command was modified by this feature: **neighbor local-as**.
BGP Support for Fast Peering Session Deactivation	12.0(29)S 12.3(14)T 12.2(33)SRA 12.2(31)SB 12.2(33)SXH 15.0(1)S	The BGP Support for Fast Peering Session Deactivation feature introduced an event-driven notification system that allows a Border Gateway Protocol (BGP) process to monitor BGP peering sessions on a per-neighbor basis. This feature improves the response time of BGP to adjacency changes by allowing BGP to detect an adjacency change and deactivate the terminated session in between standard BGP scanning intervals. Enabling this feature improves overall BGP convergence. The following command was modified by this feature: **neighbor fall-over**.

Feature Name	Releases	Feature Information
BGP Support for TCP Path MTU Discovery per Session	12.2(33)SRA 12.2(31)SB 12.2(33)SXH 12.4(20)T 15.0(1)S	BGP support for TCP path maximum transmission unit (MTU) discovery introduced the ability for BGP to automatically discover the best TCP path MTU for each BGP session. The TCP path MTU is enabled by default for all BGP neighbor sessions, but you can disable, and subsequently enable, the TCP path MTU globally for all BGP sessions or for an individual BGP neighbor session. The following commands were introduced or modified by this feature: **bgp transport**, **neighbor transport**, **show ip bgp neighbors**.
BGP Support for TTL Security Check	12.0(27)S 12.3(7)T 12.2(25)S 12.2(18)SXE 15.0(1)S	The BGP Support for TTL Security Check feature introduced a lightweight security mechanism to protect external Border Gateway Protocol (eBGP) peering sessions from CPU utilization-based attacks using forged IP packets. Enabling this feature prevents attempts to hijack the eBGP peering session by a host on a network segment that is not part of either BGP network or by a host on a network segment that is not between the eBGP peers. The following commands were introduced or modified by this feature: **neighbor ttl-security**, **show ip bgp neighbors**.
BGP IPv6 Client for Single-Hop BFD	15.1(2)S	Bidirectional Forwarding Detection (BFD) can be used to track fast forwarding path failure of BGP neighbors that use an IPv6 address. The following command was modified by this feature: **neighbor fall-over**.

Configuring Internal BGP Features

This module describes how to configure internal Border Gateway Protocol (BGP) features. Internal BGP (iBGP) refers to running Border Gateway Protocol (BGP) on networking devices within one autonomous system. BGP is an interdomain routing protocol designed to provide loop-free routing between separate routing domains (autonomous systems) that contain independent routing policies. Many companies now have large internal networks and there are many issues involved in scaling the existing internal routing protocols to match the increasing traffic demands while maintaining network efficiency.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About Internal BGP Features

BGP Routing Domain Confederation

One way to reduce the internal BGP (iBGP) mesh is to divide an autonomous system into multiple subautonomous systems and group them into a single confederation. To the outside world, the confederation looks like a single autonomous system. Each autonomous system is fully meshed within itself, and has a few connections to other autonomous systems in the same confederation. Even though the peers in different autonomous systems have external BGP (eBGP) sessions, they exchange routing information as if they were iBGP peers. Specifically, the next hop, Multi_Exit_Discriminator (MED) attribute, and local preference information is preserved. This feature allows the you to retain a single Interior Gateway Protocol (IGP) for all of the autonomous systems.

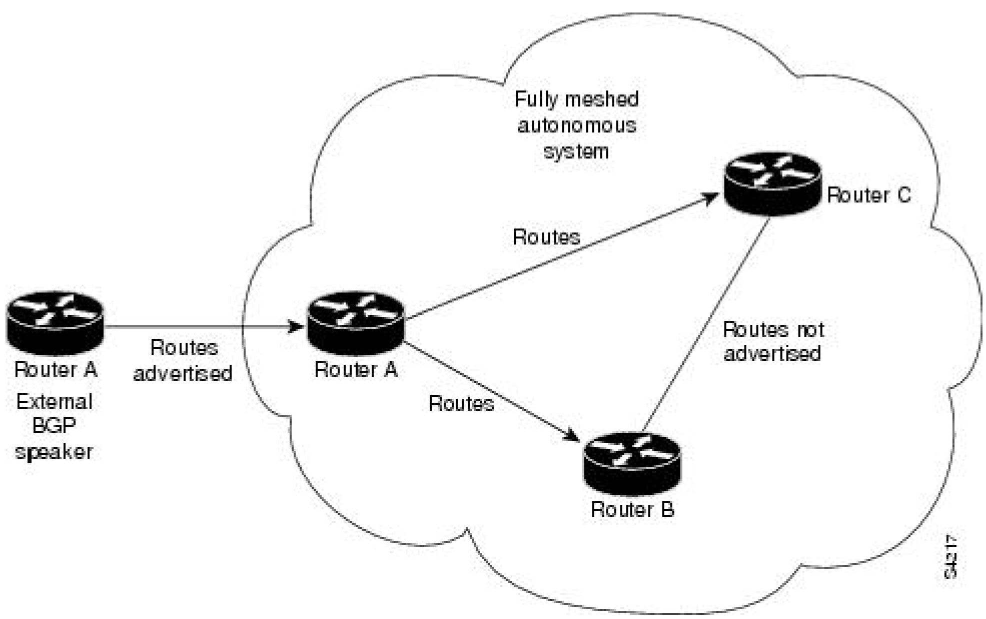
To configure a BGP confederation, you must specify a confederation identifier. To the outside world, the group of autonomous systems will look like a single autonomous system with the confederation identifier as the autonomous system number.

BGP Route Reflector

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

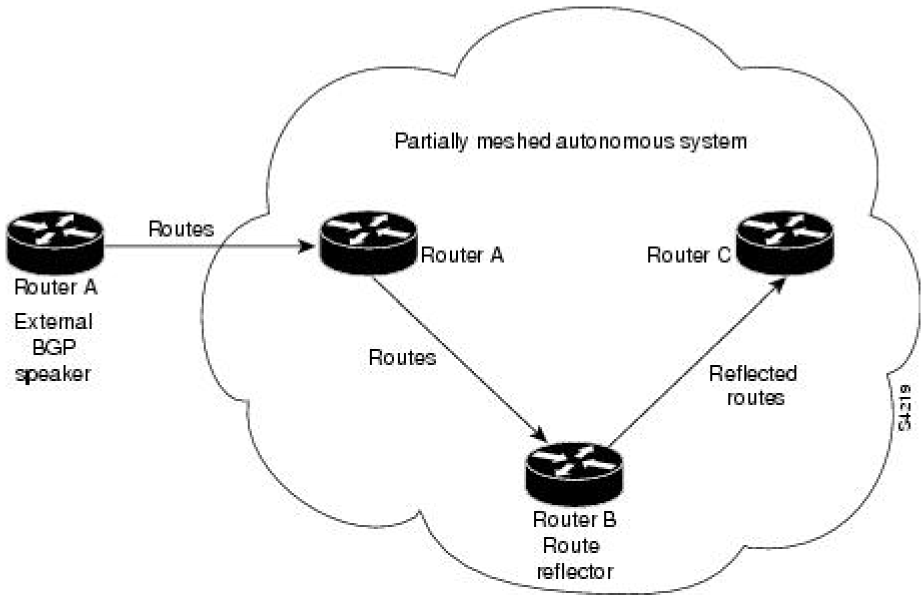
The figure below illustrates a simple iBGP configuration with three iBGP speakers (Routers A, B, and C). Without route reflectors, when Router A receives a route from an external neighbor, it must advertise it to both routers B and C. Routers B and C do not readvertise the iBGP learned route to other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

Figure 25 Three Fully Meshed iBGP Speakers



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

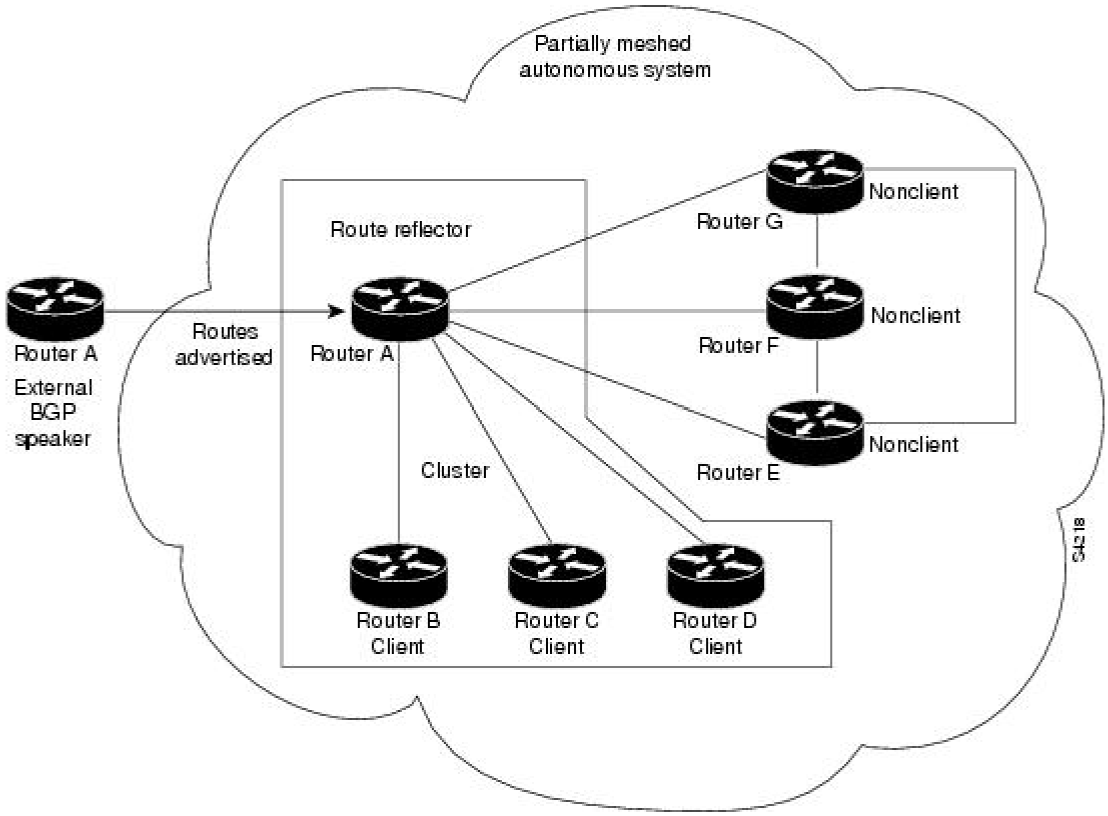
Figure 26 Simple BGP Model with a Route Reflector



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

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

Figure 27 More Complex BGP Route Reflector Model



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

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

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

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

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

• Route Reflector Mechanisms to Avoid Routing Loops, page 275

Route Reflector Mechanisms to Avoid Routing Loops

As the iBGP learned routes are reflected, routing information may loop. The route reflector model has the following mechanisms to avoid routing loops:

- Originator ID is an optional, nontransitive BGP attribute. It is a 4-byte attribute created by a route reflector. The attribute carries the router ID of the originator of the route in the local autonomous system. Therefore, if a misconfiguration causes routing information to come back to the originator, the information is ignored.
- Cluster-list is an optional, nontransitive BGP attribute. It is a sequence of cluster IDs that the route has passed. When a route reflector reflects a route from its clients to nonclient peers, and vice versa, it appends the local cluster ID to the cluster list. If the cluster list is empty, a new cluster list is created. Using this attribute, a route reflector can identify if routing information is looped back to the same cluster due to misconfiguration. If the local cluster ID is found in the cluster list, the advertisement is ignored.
- The use of **set** clauses in outbound route maps can modify attributes and possibly create routing loops. To avoid this behavior, most **set** clauses of outbound route maps are ignored for routes reflected to iBGP peers. The only **set** clause of an outbound route map that is acted upon is the **set ip next-hop** clause.

BGP Outbound Route Map on Route Reflector to Set IP Next Hop for iBGP Peer

The BGP Outbound Route Map on Route Reflector to Set IP Next Hop feature allows a route reflector to modify the next hop attribute for a reflected route.

The use of **set** clauses in outbound route maps can modify attributes and possibly create routing loops. To avoid this behavior, most **set** clauses of outbound route maps are ignored for routes reflected to iBGP peers. The only **set** clause of an outbound route map on a route reflector (RR) that is acted upon is the **set ip next-hop** clause. The **set ip next-hop** clause is applied to reflected routes.

Configuring an RR with an outbound route map allows a network administrator to modify the next hop attribute for a reflected route. By configuring a route map with the **set ip next-hop** clause, the administrator puts the RR into the forwarding path, and can configure iBGP multipath load sharing to achieve load balancing. That is, the RR can distribute outgoing packets among multiple egress points. See the "Configuring iBGP Multipath Load Sharing" module.

Caution Incorrectly setting BGP attributes for reflected routes can cause inconsistent routing, routing loops, or a loss of connectivity. Setting BGP attributes for reflected routes should only be attempted by someone who has a good understanding of the design implications.

BGP VPLS Autodiscovery Support on Route Reflector

In Cisco IOS Release 12.2(33)SRE, BGP VPLS Autodiscovery Support on Route Reflector was introduced. On the Cisco 7600 and Cisco 7200 series routers, BGP Route Reflector was enhanced to be able to reflect BGP VPLS prefixes without having VPLS explicitly configured on the route reflector. The route reflector reflects the VPLS prefixes to other provider edge (PE) routers so that the PEs do not need to have a full mesh of BGP sessions. The network administrator configures only the BGP VPLS address family on the route reflector.

For an example of a route reflector configuration that can reflect VPLS prefixes, see the Example BGP VPLS Autodiscovery Support on Route Reflector, page 288. For more information about VPLS Autodiscovery, see the VPLS Autodiscovery:BGP Based chapter in the *Cisco IOS MPLS Configuration Guide* .

BGP Route Dampening

Route dampening is a BGP feature designed to minimize the propagation of flapping routes across an internetwork. A route is considered to be flapping when its availability alternates repeatedly.

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

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

Route Dampening Minimizes Route Flapping

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

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

BGP Route Dampening Terms

The following terms are used when describing route dampening:

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

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

How to Configure Internal BGP Features

Configuring a Routing Domain Confederation

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

Command	Purpose
`Router(config-router)#` **bgp confederation identifier** *as-number*	Configures a BGP confederation.

In order to treat the neighbors from other autonomous systems within the confederation as special eBGP peers, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **bgp confederation peers** *as-number* `[as-number]`	Specifies the autonomous systems that belong to the confederation.

For an alternative way to reduce the iBGP mesh, see "Configuring a Route Reflector, page 278."

Configuring a Route Reflector

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

Command	Purpose
`Router(config-router)#` **neighbor** {*ip-address* | *peer-group-name*} **route-reflector-client**	Configures the local router as a BGP route reflector and the specified neighbor as a client.

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

Command	Purpose
`Router(config-router)#` **bgp cluster-id** *cluster-id*	Configures the cluster ID.

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

By default, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients. However, if the clients are fully meshed, the route reflector need not reflect routes to clients.

To disable client-to-client route reflection, use the **no bgp client-to-client reflection** command in router configuration mode:

Command	Purpose
`Router(config-router)#` **no bgp client-to-client reflection**	Disables client-to-client route reflection.

Configuring a Route Reflector Using a Route Map to Set Next Hop for iBGP Peer

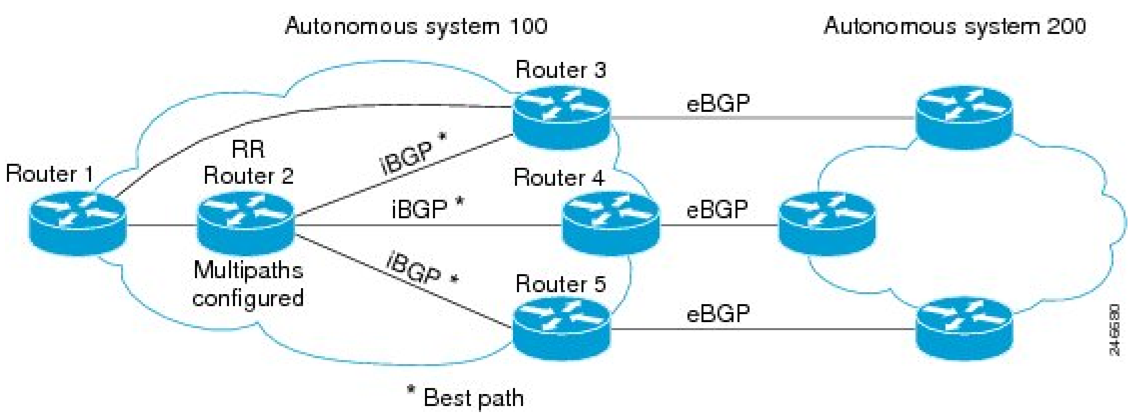
Perform this task on an RR to set a next hop for an iBGP peer. One reason to perform this task is when you want to make the RR the next hop for routes, so that you can configure iBGP load sharing. Create a route map that sets the next hop to be the RR's address, which will be advertised to the RR clients. The route map is applied only to outbound routes from the router to which the route map is applied.

Caution Incorrectly setting BGP attributes for reflected routes can cause inconsistent routing, routing loops, or a loss of connectivity. Setting BGP attributes for reflected routes should only be attempted by someone who has a good understanding of the design implications.

Note Do not use the **neighbor next-hop-self** command to modify the next hop attribute for an RR. Using the **neighbor next-hop-self** command on the RR will modify next hop attributes only for non-reflected routes and not the intended routes that are being reflected from the RR clients. To modify the next hop attribute when reflecting a route, use an outbound route map.

This task configures the RR (Router 2) in the scenario illustrated in the figure below. In this case, Router 1 is the iBGP peer whose routes' next hop is being set. Without a route map, outbound routes from Router 1 would go to next hop Router 3. Instead, setting the next hop to the RR's address will cause routes from Router 1 to go to the RR, and thus allow the RR to perform load balancing among Routers 3, 4, and 5.

Figure 28 Route Reflector Using a Route Map to Set Next Hop for an iBGP Peer



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **route-map** *map-tag*
4. **set ip next-hop** *ip-address*
5. **exit**
6. **router bgp** *as-number*
7. **address-family ipv4**
8. **maximum-paths ibgp** *number*
9. **neighbor** *ip-address* **remote-as** *as-number*
10. **neighbor** *ip-address* **activate**
11. **neighbor** *ip-address* **route-reflector-client**
12. **neighbor** *ip-address* **route-map** *map-name* **out**
13. Repeat Steps 12 through 14 for the other RR clients.
14. **end**
15. **show ip bgp neighbors**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**route-map** *map-tag* **Example:** `Router(config)# route-map rr-out`	Enters route map configuration mode to configure a route map. • The route map is created to set the next hop for the route reflector client.
Step 4	**set ip next-hop** *ip-address* **Example:** `Router(config-route-map)# set ip next-hop 10.2.0.1`	Specifies that for routes that are advertised where this route map is applied, the next-hop attribute is set to this IPv4 address. • For this task, we want to set the next hop to be the address of the RR.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Router(config-route-map)# exit`	Exits route-map configuration mode and enters global configuration mode.
Step 6	**router bgp** *as-number* **Example:** `Router(config)# router bgp 100`	Enters router configuration mode and creates a BGP routing process.
Step 7	**address-family ipv4** **Example:** `Router(config-router-af)# address-family ipv4`	Enters address family configuration mode to configure BGP peers to accept address family specific configurations.
Step 8	**maximum-paths ibgp** *number* **Example:** `Router(config-router)# maximum-paths ibgp 5`	Controls the maximum number of parallel iBGP routes that can be installed in the routing table.
Step 9	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Router(config-router-af)# neighbor 10.1.0.1 remote-as 100`	Adds an entry to the BGP neighbor table.
Step 10	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 10.1.0.1 activate`	Enables the exchange of information with the peer.
Step 11	**neighbor** *ip-address* **route-reflector-client** **Example:** `Router(config-router-af)# neighbor 10.1.0.1 route-reflector-client`	Configures the local router as a BGP route reflector, and configures the specified neighbor as a route-reflector client.

	Command or Action	Purpose
Step 12	**neighbor** *ip-address* **route-map** *map-name* **out** **Example:** `Router(config-router-af)# neighbor 10.1.0.1 route-map rr-out out`	Applies the route map to outgoing routes from this neighbor. • Reference the route map you created in Step 3.
Step 13	Repeat Steps 12 through 14 for the other RR clients.	You will not be applying a route map to the other RR clients.
Step 14	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.
Step 15	**show ip bgp neighbors** **Example:** `Router# show ip bgp neighbors`	(Optional) Displays information about the BGP neighbors, including their status as RR clients, and information about the route map configured.

Adjusting BGP Timers

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

To adjust BGP timers for all neighbors, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **timers bgp** *keepalive holdtime*	Adjusts BGP timers for all neighbors.

To adjust BGP keepalive and hold-time timers for a specific neighbor, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **neighbor** [*ip-address* | *peer-group-name*] **timers** *keepalive holdtime*	Sets the keepalive and hold-time timers (in seconds) for the specified peer or peer group.

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

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

Configuring the Router to Consider a Missing MED as Worst Path

To configure the router to consider a path with a missing MED attribute as the worst path, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **bgp bestpath med missing-as-worst**	Configures the router to consider a missing MED as having a value of infinity, making the path without a MED value the least desirable path.

Configuring the Router to Consider the MED to Choose a Path from Subautonomous System Paths

To configure the router to consider the MED value in choosing a path, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **bgp bestpath med confed**	Configures the router to consider the MED in choosing a path from among those advertised by different subautonomous systems within a confederation.

The comparison between MEDs is made only if there are no external autonomous systems in the path (an external autonomous system is an autonomous system that is not within the confederation). If there is an external autonomous system in the path, then the external MED is passed transparently through the confederation, and the comparison is not made.

The following example compares route A with these paths:

```
path= 65000 65004, med=2
path= 65001 65004, med=3
path= 65002 65004, med=4
path= 65003 1, med=1
```

In this case, path 1 would be chosen if the **bgp bestpath med confed router configuration**command is enabled. The fourth path has a lower MED, but it is not involved in the MED comparison because there is an external autonomous system is in this path.

Configuring the Router to Use the MED to Choose a Path in a Confederation

To configure the router to use the MED to choose the best path from among paths advertised by a single subautonomous system within a confederation, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **bgp deterministic med**	Configures the router to compare the MED variable when choosing among routes advertised by different peers in the same autonomous system.

Note If the **bgp always-compare-med** router configuration command is enabled, all paths are fully comparable, including those from other autonomous systems in the confederation, even if the **bgp deterministic med** command is also enabled.

Enabling BGP Route Dampening

To enable BGP route dampening, use the following command in address family or router configuration mode:

Command	Purpose
`Router(config-router)#` **bgp dampening**	Enables BGP route dampening.

To change the default values of various dampening factors, use the following command in address family or router configuration mode:

Command	Purpose
`Router(config-router)#` **bgp dampening** *half-life reuse suppress max-suppress* [**route-map** *map-name*]	Changes the default values of route dampening factors.

Monitoring and Maintaining BGP Route Dampening

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

Command	Purpose
`Router#` **show ip bgp flap-statistics**	Displays BGP flap statistics for all paths.
`Router#` **show ip bgp flap-statistics regexp** *regexp*	Displays BGP flap statistics for all paths that match the regular expression.
`Router#` **show ip bgp flap-statistics filter-list access-** *list*	Displays BGP flap statistics for all paths that pass the filter.

Command	Purpose
Router# **show ip bgp flap-statistics** *ip-address mask*	Displays BGP flap statistics for a single entry.
Router# **show ip bgp flap-statistics** *ip-address mask* **longer-prefix**	Displays BGP flap statistics for more specific entries.

To clear BGP flap statistics (thus making it less likely that the route will be dampened), use the following commands as needed:

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

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

Once a route is dampened, you can display BGP route dampening information, including the time remaining before the dampened routes will be unsuppressed. To display the information, use the following command:

Command	Purpose
Router# **show ip bgp dampened-paths**	Displays the dampened routes, including the time remaining before they will be unsuppressed.

You can clear BGP route dampening information and unsuppress any suppressed routes by using the following command:

Command	Purpose
Router# **clear ip bgp dampening** [*ip-address network-mask*]	Clears route dampening information and unsuppresses the suppressed routes.

Internal BGP Feature Configuration Examples

Example BGP Confederation Configurations with Route Maps

This section contains an example of the use of a BGP confederation configuration that includes BGP communities and route maps. For more examples of how to configure a BGP confederation, see the section Examples BGP Confederation, page 286 in this chapter.

This example shows how BGP community attributes are used with a BGP confederation configuration to filter routes.

In this example, the route map named *set-community* is applied to the outbound updates to neighbor 172.16.232.50 and the local-as community attribute is used to filter the routes. The routes that pass access list 1 have the special community attribute value local-as. The remaining routes are advertised normally. This special community value automatically prevents the advertisement of those routes by the BGP speakers outside autonomous system 200.

```
router bgp 65000
 network 10.0.1.0 route-map set-community
 bgp confederation identifier 200
 bgp confederation peers 65001
 neighbor 172.16.232.50 remote-as 100
 neighbor 172.16.233.2 remote-as 65001
!
route-map set-community permit 10
 match ip address 1
 set community local-as
!
```

Examples BGP Confederation

The following is a sample configuration that shows several peers in a confederation. The confederation consists of three internal autonomous systems with autonomous system numbers 6001, 6002, and 6003. To the BGP speakers outside the confederation, the confederation looks like a normal autonomous system with autonomous system number 500 (specified via the **bgp confederation identifier** router configuration command).

In a BGP speaker in autonomous system 6001, the **bgp confederation peers** router configuration command marks the peers from autonomous systems 6002 and 6003 as special eBGP peers. Hence peers 172.16.232.55 and 172.16.232.56 will get the local preference, next hop, and MED unmodified in the updates. The router at 10.16.69.1 is a normal eBGP speaker and the updates received by it from this peer will be just like a normal eBGP update from a peer in autonomous system 6001.

```
router bgp 6001
 bgp confederation identifier 500
 bgp confederation peers 6002 6003
 neighbor 172.16.232.55 remote-as 6002
 neighbor 172.16.232.56 remote-as 6003
 neighbor 10.16.69.1 remote-as 777
```

In a BGP speaker in autonomous system 6002, the peers from autonomous systems 6001 and 6003 are configured as special eBGP peers. 10.70.70.1 is a normal iBGP peer and 10.99.99.2 is a normal eBGP peer from autonomous system 700.

```
router bgp 6002
 bgp confederation identifier 500
 bgp confederation peers 6001 6003
 neighbor 10.70.70.1 remote-as 6002
 neighbor 172.16.232.57 remote-as 6001
 neighbor 172.16.232.56 remote-as 6003
 neighbor 10.99.99.2 remote-as 700
```

In a BGP speaker in autonomous system 6003, the peers from autonomous systems 6001 and 6002 are configured as special eBGP peers. 10.200.200.200 is a normal eBGP peer from autonomous system 701.

```
router bgp 6003
 bgp confederation identifier 500
 bgp confederation peers 6001 6002
 neighbor 172.16.232.57 remote-as 6001
 neighbor 172.16.232.55 remote-as 6002
 neighbor 10.200.200.200 remote-as 701
```

The following is a part of the configuration from the BGP speaker 10.200.200.205 from autonomous system 701 in the same example. Neighbor 172.16.232.56 is configured as a normal eBGP speaker from autonomous system 500. The internal division of the autonomous system into multiple autonomous systems is not known to the peers external to the confederation.

```
router bgp 701
 neighbor 172.16.232.56 remote-as 500
 neighbor 10.200.200.205 remote-as 701
```

Example Route Reflector Using a Route Map to Set Next Hop for iBGP Peer

The following example is based on the figure above. Router 2 is the route reflector for the clients: Routers 1, 3, 4, and 5. Router 1 is connected to Router 3, but you don't want Router 1 to forward traffic destined to AS 200 to use Router 3 as the next hop (and therefore use the direct link with Router 3); you want to direct the traffic to the RR, which can load share among Routers 3, 4, and 5.

This example configures the RR, Router 2. A route map named rr-out is applied to Router 1; the route map sets the next hop to be the RR at 10.2.0.1. When Router 1 sees that the next hop is the RR address, Router 1 forwards the routes to the RR. When the RR receives packets, it will automatically load share among the iBGP paths. A maximum of five iBGP paths are allowed.

Router 2

```
route-map rr-out
 set ip next-hop 10.2.0.1
!
interface gigabitethernet 0/0
 ip address 10.2.0.1 255.255.0.0
router bgp 100
 address-family ipv4 unicast
 maximum-paths ibgp 5
 neighbor 10.1.0.1 remote-as 100
 neighbor 10.1.0.1 activate
 neighbor 10.1.0.1 route-reflector-client
 neighbor 10.1.0.1 route-map rr-out out
!
 neighbor 10.3.0.1 remote-as 100
 neighbor 10.3.0.1 activate
 neighbor 10.3.0.1 route-reflector-client
!
 neighbor 10.4.0.1 remote-as 100
```

```
 neighbor 10.4.0.1 activate
 neighbor 10.4.0.1 route-reflector-client
!
 neighbor 10.5.0.1 remote-as 100
 neighbor 10.5.0.1 activate
 neighbor 10.5.0.1 route-reflector-client
end
```

Example BGP VPLS Autodiscovery Support on Route Reflector

In the following example, a host named PE-RR (indicating Provider Edge Route Reflector) is configured as a route reflector capable of reflecting VPLS prefixes. The VPLS address family is configured by **address-family l2vpn vpls** below.

```
hostname PE-RR
!
router bgp 1
 bgp router-id 1.1.1.3
 no bgp default route-target filter
 bgp log-neighbor-changes
neighbor iBGP_PEERS peer-group
neighbor iBGP_PEERS remote-as 1
neighbor iBGP_PEERS update-source Loopback1
neighbor 1.1.1.1 peer-group iBGP_PEERS
neighbor 1.1.1.2 peer-group iBGP_PEERS
!
address-family l2vpn vpls
  neighbor iBGP_PEERS send-community extended
  neighbor iBGP_PEERS route-reflector-client
  neighbor 1.1.1.1 peer-group iBGP_PEERS
  neighbor 1.1.1.2 peer-group iBGP_PEERS
exit-address-family
!
```

Additional References

The following sections provide references related to configuring internal BGP features.

Related Documents

Related Topic	Document Title
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*
BGP overview	"Cisco BGP Overview" module
Basic BGP configuration tasks	"Configuring a Basic BGP Network" module
iBGP multipath load sharing	"iBGP Multipath Load Sharing" module
Connecting to a service provider	"Connecting to a Service Provider Using External BGP" module
Configuring features that apply to multiple IP routing protocols	*Cisco IOS IP Routing: Protocol-Independent Configuration Guide*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
•	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1772	*Application of the Border Gateway Protocol in the Internet*
RFC 1773	*Experience with the BGP Protocol*
RFC 1774	*BGP-4 Protocol Analysis*
RFC 1930	*Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)*
RFC 2519	*A Framework for Inter-Domain Route Aggregation*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 2918	*Route Refresh Capability for BGP-4*
RFC 3392	*Capabilities Advertisement with BGP-4*
RFC 4271	A Border Gateway Protocol 4 (BGP-4)
RFC 4893	BGP Support for Four-octet AS Number Space
RFC 5396	Textual Representation of Autonomous system (AS) Numbers
RFC 5398	Autonomous System (AS) Number Reservation for Documentation Use

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Configuring Internal BGP Features

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 18 Feature Information for Configuring Internal BGP Features

Feature Name	Releases	Feature Configuration Information
Configuring internal BGP features	10.3 12.0(32)S12 12.0(7)T 12.2(33)SRA 12.2(33)SXH	All the features contained in this module are considered to be legacy features and will work in all trains release images. The following commands were introduced or modified by these features: • **bgp always-compare-med** • **bgp bestpath med confed** • **bgp bestpath med missing-as-worst** • **bgp client-to-client reflection** • **bgp cluster-id** • **bgp confederation identifier** • **bgp confederation peers** • **bgp dampening** • **bgp deterministic med** • **clear ip bgp dampening** • **clear ip bgp flap-statistics** • **neighbor route-reflector-client** • **neighbor timers** • **show ip bgp** • **show ip bgp dampened-paths** • **show ip bgp flap-statistics** • **timers bgp**
BGP Outbound Route Map on Route Reflector to Set IP Next Hop	12.0(22)S 12.0(16)ST 12.2 12.2(14)S 15.0(1)S	The BGP Outbound Route Map on Route Reflector to Set IP Next Hop feature allows a route reflector to modify the next hop attribute for a reflected route.
BGP VPLS Autodiscovery Support on Route Reflector	12.2(33)SRE	This feature was introduced on the Cisco 7600 and Cisco 7200 series routers. This feature is documented in the following sections:

Configuring Advanced BGP Features

This module describes configuration tasks for various advanced Border Gateway Protocol (BGP) features. BGP is an interdomain routing protocol designed to provide loop-free routing between organizations. This module contains tasks to configure BGP next-hop address tracking, BGP Nonstop Forwarding (NSF) awareness using the BGP graceful restart capability, route dampening, Bidirectional Forwarding Detection (BFD) support for BGP, BGP MIB support and BGP support for Multi-Topology Routing (MTR).

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Configuring Advanced BGP Features

Before configuring advanced BGP features you should be familiar with the "Cisco BGP Overview" module and the "Configuring a Basic BGP Network" module.

Restrictions for Configuring Advanced BGP Features

- A router that runs Cisco IOS software can be configured to run only one BGP routing process and to be a member of only one BGP autonomous system. However, a BGP routing process and autonomous system can support multiple address family configurations.
- Multicast BGP peer support is not available in Cisco IOS software after Release 12.2(33)SRA.

Information About Configuring Advanced BGP Features

BGP Version 4

Border Gateway Protocol (BGP) is an interdomain routing protocol designed to provide loop-free routing between separate routing domains that contain independent routing policies (autonomous systems). The Cisco IOS software implementation of BGP version 4 includes multiprotocol extensions to allow BGP to carry routing information for IP multicast routes and multiple Layer 3 protocol address families including IP Version 4 (IPv4), IP Version 6 (IPv6), Virtual Private Networks version 4 (VPNv4), and Connectionless Network Services (CLNS). For more details about configuring a basic BGP network, see the "Configuring a Basic BGP Network" module.

BGP is mainly used to connect a local network to an external network to gain access to the Internet or to connect to other organizations. When connecting to an external organization, external BGP (eBGP) peering sessions are created. For more details about connecting to external BGP peers, see the "Connecting to a Service Provider Using External BGP" module.

Although BGP is referred to as an exterior gateway protocol (EGP) many networks within an organization are becoming so complex that BGP can be used to simplify the internal network used within the organization. BGP peers within the same organization exchange routing information through internal BGP (iBGP) peering sessions. For more details about internal BGP peers, see the "Configuring Internal BGP Features" chapter of the Cisco IOS IP Routing Configuration Guide.

Note BGP requires more configuration than other routing protocols and the effects of any configuration changes must be fully understood. Incorrect configuration can create routing loops and negatively impact normal network operation.

BGP Support for Next-Hop Address Tracking

To configure BGP next-hop address tracking you should understand the following concepts:

BGP Next-Hop Address Tracking

The BGP next-hop address tracking feature is enabled by default when a supporting Cisco software image is installed. BGP next-hop address tracking is event driven. BGP prefixes are automatically tracked as peering sessions are established. Next-hop changes are rapidly reported to the BGP routing process as they are updated in the RIB. This optimization improves overall BGP convergence by reducing the response time to next-hop changes for routes installed in the RIB. When a best-path calculation is run in between BGP scanner cycles, only next-hop changes are tracked and processed.

Default BGP Scanner Behavior

BGP monitors the next hop of installed routes to verify next-hop reachability and to select, install, and validate the BGP best path. By default, the BGP scanner is used to poll the RIB for this information every 60 seconds. During the 60 second time period between scan cycles, Interior Gateway Protocol (IGP) instability or other network failures can cause black holes and routing loops to temporarily form.

Selective BGP Next-Hop Route Filtering

In Cisco IOS Release 12.4(4)T, 12.2(33)SRB, and later releases, BGP selective next-hop route filtering was implemented as part of the BGP Selective Address Tracking feature to support BGP next-hop address tracking. Selective next-hop route filtering uses a route map to selectively define routes to help resolve the BGP next hop.

The ability to use a route map with the **bgp nexthop**command allows the configuration of the length of a prefix that applies to the BGP Next_Hop attribute. The route map is used during the BGP bestpath calculation and is applied to the route in the routing table that covers the next-hop attribute for BGP prefixes. If the next-hop route fails the route map evaluation, the next-hop route is marked as unreachable. This command is per address family, so different route maps can be applied for next-hop routes in different address families.

Note Only **match ip address** and **match source-protocol** commands are supported in the route map. No **set** commands or other **match** commands are supported.

BGP Next_Hop Attribute

The Next_Hop attribute identifies the next-hop IP address to be used as the BGP next hop to the destination. The router makes a recursive lookup to find the BGP next hop in the routing table. In external BGP (eBGP), the next hop is the IP address of the peer that sent the update. Internal BGP (iBGP) sets the next-hop address to the IP address of the peer that advertised the prefix for routes that originate internally. When any routes to iBGP that are learned from eBGP are advertised, the Next_Hop attribute is unchanged.

A BGP next-hop IP address must be reachable in order for the router to use a BGP route. Reachability information is usually provided by the IGP, and changes in the IGP can influence the forwarding of the next-hop address over a network backbone.

BGP Nonstop Forwarding Awareness

To configure BGP Nonstop Forwarding (NSF) awareness you should understand the following concepts:

- Cisco NSF Routing and Forwarding Operation, page 296
- Cisco Express Forwarding for NSF, page 296
- BGP Graceful Restart for NSF, page 297

Cisco NSF Routing and Forwarding Operation

Cisco NSF is supported by the BGP, EIGRP, OSPF, and IS-IS protocols for routing and by Cisco Express Forwarding (CEF) for forwarding. Of the routing protocols, BGP, EIGRP, OSPF, and IS-IS have been enhanced with NSF-capability and awareness, which means that routers running these protocols can detect a switchover and take the necessary actions to continue forwarding network traffic and to recover route information from the peer devices.

In this document, a networking device is said to be NSF-aware if it is running NSF-compatible software. A device is said to be NSF-capable if it has been configured to support NSF; therefore, it would rebuild routing information from NSF-aware or NSF-capable neighbors.

Each protocol depends on CEF to continue forwarding packets during switchover while the routing protocols rebuild the Routing Information Base (RIB) tables. Once the routing protocols have converged, CEF updates the FIB table and removes stale route entries. CEF then updates the line cards with the new FIB information.

Note Currently, EIGRP supports only NSF awareness.

Cisco Express Forwarding for NSF

A key element of NSF is packet forwarding. In a Cisco networking device, packet forwarding is provided by CEF. CEF maintains the FIB and uses the FIB information that was current at the time of the switchover to continue forwarding packets during a switchover. This feature reduces traffic interruption during the switchover.

During normal NSF operation, CEF on the active RP synchronizes its current FIB and adjacency databases with the FIB and adjacency databases on the standby RP. Upon switchover of the active RP, the standby RP initially has FIB and adjacency databases that are mirror images of those that were current on the active RP. For platforms with intelligent line cards, the line cards will maintain the current forwarding information over a switchover; for platforms with forwarding engines, CEF will keep the forwarding engine on the standby RP current with changes that are sent to it by CEF on the active RP. In this way, the line cards or forwarding engines will be able to continue forwarding after a switchover as soon as the interfaces and a data path are available.

As the routing protocols start to repopulate the RIB on a prefix-by-prefix basis, the updates in turn cause prefix-by-prefix updates for CEF, which it uses to update the FIB and adjacency databases. Existing and new entries will receive the new version ("epoch") number, indicating that they have been refreshed. The forwarding information is updated on the line cards or forwarding engine during convergence. The RP signals when the RIB has converged. The software removes all FIB and adjacency entries that have an epoch older than the current switchover epoch. The FIB now represents the newest routing protocol forwarding information

The routing protocols run only on the active RP, and they receive routing updates from their neighbor routers. Routing protocols do not run on the standby RP. Following a switchover, the routing protocols request that the NSF-aware neighbor devices send state information to help rebuild the routing tables.

Note For NSF operation, the routing protocols depend on CEF to continue forwarding packets while the routing protocols rebuild the routing information.

BGP Graceful Restart for NSF

When an NSF-capable router begins a BGP session with a BGP peer, it sends an OPEN message to the peer. Included in the message is a declaration that the NSF-capable or NSF-aware router has "graceful restart capability." Graceful restart is the mechanism by which BGP routing peers avoid a routing flap following a switchover. If the BGP peer has received this capability, it is aware that the device sending the message is NSF-capable. Both the NSF-capable router and its BGP peer(s) (NSF-aware peers) need to exchange the graceful restart capability in their OPEN messages, at the time of session establishment. If both the peers do not exchange the graceful restart capability, the session will not be graceful restart capable.

If the BGP session is lost during the RP switchover, the NSF-aware BGP peer marks all the routes associated with the NSF-capable router as stale; however, it continues to use these routes to make forwarding decisions for a set period of time. This functionality means that no packets are lost while the newly active RP is waiting for convergence of the routing information with the BGP peers.

After an RP switchover occurs, the NSF-capable router reestablishes the session with the BGP peer. In establishing the new session, it sends a new graceful restart message that identifies the NSF-capable router as having restarted.

At this point, the routing information is exchanged between the two BGP peers. Once this exchange is complete, the NSF-capable device uses the routing information to update the RIB and the FIB with the new forwarding information. The NSF-aware device uses the network information to remove stale routes from its BGP table. Following that, the BGP protocol is fully converged.

If a BGP peer does not support the graceful restart capability, it will ignore the graceful restart capability in an OPEN message but will establish a BGP session with the NSF-capable device. This functionality will allow interoperability with non-NSF-aware BGP peers (and without NSF functionality), but the BGP session with non-NSF-aware BGP peers will not be graceful restart capable.

BGP NSF Awareness

BGP support for NSF requires that neighbor routers are NSF-aware or NSF-capable. NSF awareness in BGP is also enabled by the graceful restart mechanism. A router that is NSF-aware functions like a router that is NSF-capable with one exception: an NSF-aware router is incapable of performing an SSO operation. However, a router that is NSF-aware is capable of maintaining a peering relationship with a NSF-capable neighbor during a NSF SSO operation, as well as holding routes for this neighbor during the SSO operation.

The BGP Nonstop Forwarding Awareness feature provides an NSF-aware router with the capability to detect a neighbor that is undergoing an SSO operation, maintain the peering session with this neighbor, retain known routes, and continue to forward packets for these routes. The deployment of BGP NSF awareness can minimize the affects of route-processor (RP) failure conditions and improve the overall network stability by reducing the amount of resources that are normally required for reestablishing peering with a failed router.

NSF awareness for BGP is not enabled by default. The **bgp graceful-restart** command is used to globally enable NSF awareness on a router that is running BGP. NSF-aware operations are also transparent to the network operator and BGP peers that do not support NSF capabilities.

Note NSF awareness is enabled automatically in supported software images for Interior Gateway Protocols, such as EIGRP, IS-IS, and OSPF. In BGP, global NSF awareness is not enabled automatically and must be started by issuing the **bgp graceful-restart** command in router configuration mode.

BGP Graceful Restart per Neighbor

In Cisco IOS Releases 12.2(33)SRC, 12.2(33)SB (on platforms including the Cisco 10000 series routers), 15.0(1)M, and later releases, the ability to enable or disable BGP graceful restart for every individual BGP neighbor was introduced. Three new methods of configuring BGP graceful restart for BGP peers, in addition to the existing global BGP graceful restart configuration, are now available. Graceful restart can be enabled or disabled for a BGP peer or a BGP peer group using the **neighbor ha-mode graceful-restart** command, or a BGP peer can inherit a graceful restart configuration from a BGP peer-session template using the **ha-mode graceful-restart**command.

Although BGP graceful restart is disabled by default, the existing global command enables graceful restart for all BGP neighbors regardless of their capabilities. The ability to enable or disable BGP graceful restart for individual BGP neighbors provides a greater level of control for a network administrator.

When the BGP graceful restart capability is configured for an individual neighbor, each method of configuring graceful restart has the same priority, and the last configuration instance is applied to the neighbor. For example, if global graceful restart is enabled for all BGP neighbors but an individual neighbor is subsequently configured as a member of a peer group for which the graceful restart is disabled, graceful restart is disabled for that neighbor.

The configuration of the restart and stale-path timers is available only with the global **bgp graceful-restart** command, but the default values are set when the **neighbor ha-mode graceful-restart**or **ha-mode graceful-restart** commands are configured. The default values are optimal for most network deployments, and these values should be adjusted only by an experienced network operator.

BGP Peer Session Templates

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

General session commands can be configured once in a peer session template and then applied to many neighbors through the direct application of a peer session template or through indirect inheritance from a peer session template. The configuration of peer session templates simplifies the configuration of general session commands that are commonly applied to all neighbors within an autonomous system.

Peer session templates support direct and indirect inheritance. A BGP neighbor can be configured with only one peer session template at a time, and that peer session template can contain only one indirectly inherited peer session template. A BGP neighbor can directly inherit only one session template and can indirectly inherit up to seven additional peer session templates.

Peer session templates support inheritance. A directly applied peer session template can directly or indirectly inherit configurations from up to seven peer session templates. So, a total of eight peer session templates can be applied to a neighbor or neighbor group.

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

For more details about BGP peer session templates, see the "Configuring a Basic BGP Network" module.

To use a BGP peer session template to enable or disable BGP graceful restart, see the section "Enabling and Disabling BGP Graceful Restart Using BGP Peer Session Templates".

BGP Route Dampening

Route dampening is a BGP feature designed to minimize the propagation of flapping routes across an internetwork. A route is considered to be flapping when its availability alternates repeatedly.

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

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

Minimizing Flapping

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

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

Understanding Route Dampening Terms

The following terms are used when describing route dampening:

- Flap--A route whose availability alternates repeatedly.
- History state--After a route flaps once, it is assigned a penalty and put into history state, meaning the router does not have the best path, based on historical information.
- Penalty--Each time a route flaps, the router configured for route dampening in another autonomous system assigns the route a penalty of 1000. Penalties are cumulative. The penalty for the route is stored in the BGP routing table until the penalty exceeds the suppress limit. At that point, the route state changes from history to damp.
- Damp state--In this state, the route has flapped so often that the router will not advertise this route to BGP neighbors.

- Suppress limit--A route is suppressed when its penalty exceeds this limit. The default value is 2000.
- Half-life--Once the route has been assigned a penalty, the penalty is decreased by half after the half-life period (which is 15 minutes by default). The process of reducing the penalty happens every 5 seconds.
- Reuse limit--As the penalty for a flapping route decreases and falls below this reuse limit, the route is unsuppressed. That is, the route is added back to the BGP table and once again used for forwarding. The default reuse limit is 750. The process of unsuppressing routes occurs at 10-second increments. Every 10 seconds, the router finds out which routes are now unsuppressed and advertises them to the world.
- Maximum suppress limit--This value is the maximum amount of time a route can be suppressed. The default value is four times the half-life.

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

BFD for BGP

Bidirectional Forwarding Detection (BFD) support for BGP was introduced in Cisco IOS Releases 12.0(31)S, 12.4(4)T, 12.0(32)S, 12.2(33)SRA,12.2(33)SXH, 12.2(33)SB, and later releases. BFD is a detection protocol designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols. In addition to fast forwarding path failure detection, BFD provides a consistent failure detection method for network administrators. Because the network administrator can use BFD to detect forwarding path failures at a uniform rate, rather than the variable rates for different routing protocol hello mechanisms, network profiling and planning will be easier, and reconvergence time will be consistent and predictable. The main benefit of implementing BFD for BGP is a marked decrease in reconvergence time.

One caveat exists for BFD; BFD and BGP graceful restart capability cannot both be configured on a router running BGP. If an interface goes down, BFD detects the failure and indicates that the interface cannot be used for traffic forwarding and the BGP session goes down, but graceful restart still allows traffic forwarding on platforms that support NSF even though the BGP session is down, allowing traffic forwarding using the interface that is down. Configuring both BFD and BGP graceful restart for NSF on a router running BGP may result in suboptimal routing.

For more details about BFD, see the "Bidirectional Forwarding Detection" module of the *Cisco IOS IP Routing: BFD Configuration Guide*.

BGP MIB Support

The Management Information Base (MIB) to support BGP is the CISCO-BGP4-MIB. In Cisco IOS Release 12.0(26)S, 12.3(7)T, 12.2(25)S, 12.2(33)SRA, 12.2(33)SXH, and later releases, the BGP MIB Support Enhancements feature introduced support in the CISCO-BGP4-MIB for new SNMP notifications. The following sections describe the objects and notifications (traps) that are supported:

BGP FSM Transition Change Support

The *cbgpRouteTable* supports BGP Finite State Machine (FSM) transition state changes.

The *cbgpFsmStateChange* object allows you to configure SNMP notifications (traps) for all FSM transition state changes. This notification contains the following MIB objects:

- *bgpPeerLastError*
- *bgpPeerState*
- *cbgpPeerLastErrorTxt*

- *cbgpPeerPrevState*

The *cbgpBackwardTransition* object supports all BGP FSM transition state changes. This object is sent each time the FSM moves to either a higher or lower numbered state. This notification contains the following MIB objects:

- *bgpPeerLastError*
- *bgpPeerState*
- *cbgpPeerLastErrorTxt*
- *cbgpPeerPrevState*

The **snmp-server enable bgp traps** command allows you to enable the traps individually or together with the existing FSM backward transition and established state traps as defined in RFC 1657.

BGP Route Received Route Support

The *cbgpRouteTable* object supports the total number of routes received by a BGP neighbor. The following MIB object is used to query the CISCO-BGP4-MIB for routes that are learned from individual BGP peers:

- *cbgpPeerAddrFamilyPrefixTable*

Routes are indexed by the address-family identifier (AFI) or subaddress-family identifier (SAFI). The prefix information displayed in this table can also viewed in the output of the **show ip bgp** command.

BGP Prefix Threshold Notification Support

The *cbgpPrefixMaxThresholdExceed* and *cbgpPrfefixMaxThresholdClear* objects were introduced to allow you to poll for the total number of routes received by a BGP peer.

The *cbgpPrefixMaxThresholdExceed* object allows you to configure SNMP notifications to be sent when the prefix count for a BGP session has exceeded the configured value. This notification is configured on a per address family basis. The prefix threshold is configured with the **neighbor maximum-prefix** command. This notification contains the following MIB objects:

- *cbgpPeerPrefixAdminLimit*
- *cbgpPeerPrefixThreshold*

The cbgpPrfefixMaxThresholdClear object allows you to configure SNMP notifications to be sent when the prefix count drops below the clear trap limit. This notification is configured on a per address family basis. This notification contains the following objects:

- *cbgpPeerPrefixAdminLimit*
- *cbgpPeerPrefixClearThreshold*

Notifications are sent when the prefix count drops below the clear trap limit for an address family under a BGP session after the cbgpPrefixMaxThresholdExceed notification is generated. The clear trap limit is calculated by subtracting 5 percent from the maximum prefix limit value configured with the **neighbor maximum-prefix** command. This notification will not be generated if the session goes down for any other reason after the cbgpPrefixMaxThresholdExceed is generated.

VPNv4 Unicast Address Family Route Support

The *cbgpRouteTable* object allows you to configure SNMP GET operations for VPNv4 unicast address-family routes.

The following MIB object allows you to query for multiple BGP capabilities (for example, route refresh, multiprotocol BGP extensions, and graceful restart):

- *cbgpPeerCapsTable*

The following MIB object allows you to query for IPv4 and VPNv4 address family routes:

- *cbgpPeerAddrFamilyTable*

Each route is indexed by peer address, prefix, and prefix length. This object indexes BGP routes by the AFI and then by the SAFI. The AFI table is the primary index, and the SAFI table is the secondary index. Each BGP speaker maintains a local Routing Information Base (RIB) for each supported AFI and SAFI combination.

cbgpPeerTable Support

The *cbgpPeerTable* has been modified to support the enhancements described in this document. The following new table objects are supported in the CISCO-BGP-MIB.my:

- *cbgpPeerLastErrorTxt*
- *cbgpPeerPrevState*

The following table objects are not supported. The status of theses objects is listed as deprecated, and these objects are not operational:

- *cbgpPeerPrefixAccepted*
- *cbgpPeerPrefixDenied*
- *cbgpPeerPrefixLimit*
- *cbgpPeerPrefixAdvertised*
- *cbgpPeerPrefixSuppressed*
- *cbgpPeerPrefixWithdrawn*

BGP Support for MTR

BGP support for MTR was introduced in Cisco IOS Release 12.2(33)SRB. For more details, see the "Multi-Topology Routing" documentation. Before using BGP to support MTR, you should be familiar with the following concepts:

- BGP Network Scope, page 302
- MTR CLI Hierarchy Under BGP, page 303
- BGP Sessions for Class-Specific Topologies, page 303
- Topology Translation Using BGP, page 304
- Topology Import Using BGP, page 304

BGP Network Scope

A new configuration hierarchy, named scope, has been introduced into the BGP protocol. To implement MTR for BGP, the scope hierarchy is required, but the scope hierarchy is not limited to MTR use. The scope hierarchy introduces some new configuration modes such as router scope configuration mode. Router scope configuration mode is entered by configuring the **scope** command in router configuration mode, and a collection of routing tables is created when this command is entered. BGP commands configured under the scope hierarchy are configured for a single network (globally), or on a per-VRF basis, and are referred to as scoped commands. The scope hierarchy can contain one or more address families.

MTR CLI Hierarchy Under BGP

The BGP CLI has been modified to provide backwards compatibility for pre-MTR BGP configuration and to provide a hierarchical implementation of MTR. Router configuration mode is backwards compatible with the pre-address family and pre-MTR configuration CLI. Global commands that affect all networks are configured in this configuration mode. For address-family and topology configuration, general session commands and peer templates can be configured to be used in the address-family or topology configuration modes.

After any global commands are configured, the scope is defined either globally or for a specific VRF. Address family configuration mode is entered by configuring the **address-family** command in router scope configuration mode or router configuration mode. Unicast is the default address family if no subaddress family (SAFI) is specified. MTR supports only the IPv4 address family with a SAFI of unicast or multicast. Entering address family configuration mode from router configuration mode configures BGP to use pre-MTR-based CLI. This configuration mode is backwards compatible with pre-existing address family configurations. Entering address family configuration mode from router scope configuration mode configures the router to use the hierarchical CLI that supports MTR. Address family configuration parameters that are not specific to a topology are entered in this address family configuration mode.

BGP topology configuration mode is entered by configuring the **topology**(BGP) command in address family configuration mode. Up to 32 topologies (including the base topology) can be configured on a router. The topology ID is configured by entering the **bgp tid** command. All address family and subaddress family configuration parameters for the topology are configured here.

Note Configuring a scope for a BGP routing process removes CLI support for pre-MTR-based configuration.

The following shows the hierarchy levels that are used when configuring BGP for MTR implementation:

```
router bgp <
autonomous-system-number
>
 ! global commands

 scope {global | vrf <
vrf-name
>}
  ! scoped commands

  address-family {<
afi
>} [<
safi
>]
   ! address family specific commands

   topology {<
topology-name
> | base}
    ! topology specific commands
```

BGP Sessions for Class-Specific Topologies

MTR is configured under BGP on a per-session basis. The base unicast and multicast topologies are carried in the global (default) session. A separate session is created for each class-specific topology that is configured under a BGP routing process. Each session is identified by its topology ID. BGP performs a best-path calculation individually for each class-specific topology. A separate RIB and FIB are maintained for each session.

Topology Translation Using BGP

Depending on the design and policy requirements for your network, you may need to install routes from a class-specific topology on one router in a class-specific topology on a neighboring router. Topology translation functionality using BGP provides support for this operation. Topology translation is BGP neighbor-session based. The **neighbor translate-topology** command is configured using the IP address and topology ID from the neighbor.

The topology ID identifies the class-specific topology of the neighbor. The routes in the class-specific topology of the neighbor are installed in the local class-specific RIB. BGP performs a best-path calculation on all installed routes and installs these routes into the local class-specific RIB. If a duplicate route is translated, BGP will select and install only one instance of the route per standard BGP best-path calculation behavior.

Topology Import Using BGP

Topology import functionality using BGP is similar to topology translation. The difference is that routes are moved between class-specific topologies on the same router using BGP. This function is configured by entering the **import topology** command. The name of the class-specific topology or base topology is specified when entering this command. Best-path calculations are run on the imported routes before they are installed into the topology RIB. This command also includes a **route-map** keyword to allow you to filter routes that are moved between class-specific topologies.

How to Configure Advanced BGP Features

Configuring BGP Next-Hop Address Tracking

The tasks in this section show how configure BGP next-hop address tracking. BGP next-hop address tracking significantly improves the response time of BGP to next-hop changes in the RIB. However, unstable Interior Gateway Protocol (IGP) peers can introduce instability to BGP neighbor sessions. We recommend that you aggressively dampen unstable IGP peering sessions to reduce the possible impact to BGP. For more details about configuring route dampening, see the Configuring BGP Route Dampening, page 326.

Disabling BGP Next-Hop Address Tracking

Perform this task to disable BGP next-hop address tracking. BGP next-hop address tracking is enabled by default under the IPv4 and VPNv4 address families. Beginning with Cisco IOS Release 12.2(33)SB6, BGP next-hop address tracking is also enabled by default under the VPNv6 address family whenever the next hop is an IPv4 address mapped to an IPv6 next-hop address.

Disabling next hop address tracking may be useful if you the network has unstable IGP peers and route dampening is not resolving the stability issues. To reenable BGP next-hop address tracking, use the **bgp nexthop**command with the **trigger** and **enable** keywords.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4** [[**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**] | **vpnv6** [**unicast**]]
5. no **bgp nexthop trigger enable**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 64512`	Enters router configuration mod to create or configure a BGP routing process.
Step 4	**address-family ipv4** [[**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**] | **vpnv6** [**unicast**]] **Example:** `Router(config-router)# address-family ipv4 unicast`	Enter address family configuration mode to configure BGP peers to accept address family-specific configurations. • The example creates an IPv4 unicast address family session.

	Command or Action	Purpose
Step 5	no **bgp nexthop trigger enable** **Example:** `Router(config-router-af)# no bgp nexthop trigger enable`	Disables BGP next-hop address tracking. • Next-hop address tracking is enabled by default for IPv4 and VPNv4 address family sessions. • The example disables next-hop address tracking.
Step 6	**end** **Example:** `Router(config-router-af)# end`	Exits address-family configuration mode, and enters Privileged EXEC mode.

Adjusting the Delay Interval for BGP Next-Hop Address Tracking

Perform this task to adjust the delay interval between routing table walks for BGP next-hop address tracking.

You can increase the performance of this feature by tuning the delay interval between full routing table walks to match the tuning parameters for the Interior Gateway protocol (IGP). The default delay interval is 5 seconds. This value is optimal for a fast-tuned IGP. In the case of an IGP that converges more slowly, you can change the delay interval to 20 seconds or more, depending on the IGP convergence time.

BGP next-hop address tracking significantly improves the response time of BGP to next-hop changes in the RIB. However, unstable Interior Gateway Protocol (IGP) peers can introduce instability to BGP neighbor sessions. We recommend that you aggressively dampen unstable IGP peering sessions to reduce the possible impact to BGP.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4** [[**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]]
5. **bgp nexthop trigger delay delay-timer**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 64512`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**address-family ipv4** [[**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]] **Example:** `Router(config-router)# address-family ipv4 unicast`	Enter address family configuration mode to configure BGP peers to accept address family-specific configurations. • The example creates an IPv4 unicast address family session.
Step 5	**bgp nexthop trigger delay delay-timer** **Example:** `Router(config-router-af)# bgp nexthop trigger delay 20`	Configures the delay interval between routing table walks for next-hop address tracking. • The time period determines how long BGP will wait before starting a full routing table walk after notification is received. • The value for the *delay-timer* argument is a number from 1 to 100 seconds. The default value is 5 seconds. • The example configures a delay interval of 20 seconds.
Step 6	**end** **Example:** `Router(config-router-af)# end`	Exits address-family configuration mode, and enters privileged EXEC mode.

Configuring BGP Selective Next-Hop Route Filtering

Perform this task to configure selective next-hop route filtering using a route map to filter potential next-hop routes. This task uses prefix lists and route maps to match IP addresses or source protocols and can be used to avoid aggregate addresses and BGP prefixes being considered as next-hop routes. Only **match ip address** and **match source-protocol** commands are supported in the route map. No **set** commands or other **match** commands are supported.

For more examples of how to use the **bgp nexthop** command, see the Configuring BGP Selective Next-Hop Route Filtering Examples.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
5. **bgp nexthop route-map** *map-name*
6. **exit**
7. **exit**
8. **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*]
9. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
10. **match ip address prefix-list** *prefix-list-name* [*prefix-list-name*...]
11. **exit**
12. **route-map** *map-name* [**permit**| **deny**][*sequence-number*]
13. **end**
14. **show ip bgp** [*network*] [*network-mask*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 5	**bgp nexthop route-map** *map-name* **Example:** `Router(config-router-af)# bgp nexthop route-map CHECK-NEXTHOP`	Permits a route map to selectively define routes to help resolve the BGP next hop. • In this example the route map named CHECK-NEXTHOP is created.
Step 6	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 7	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 8	**ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*}[**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list FILTER25 seq 5 permit 0.0.0.0/0 le 25`	Creates a prefix list for BGP next-hop route filtering. • Selective next-hop route filtering supports prefix length matching or source protocol matching on a per address-family basis. • The example creates a prefix list named FILTER25 that permits routes only if the mask length is more than 25; this will avoid aggregate routes being considered as the next-hop route.
Step 9	**route-map** *map-name* [**permit**| **deny**] [*sequence-number*] **Example:** `Router(config)# route-map CHECK-NEXTHOP deny 10`	Configures a route map and enters route map configuration mode. • In this example, a route map named CHECK-NEXTHOP is created. If there is an IP address match in the following **match** command, the IP address will be denied.

	Command or Action	Purpose
Step 10	**match ip address prefix-list** *prefix-list-name* [*prefix-list-name*...] **Example:** `Router(config-route-map)# match ip address prefix-list FILTER25`	Matches the IP addresses in the specified prefix list. • Use the *prefix-list-name* argument to specify the name of a prefix list. The ellipsis means that more than one prefix list can be specified. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 11	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 12	**route-map** *map-name* [**permit**| **deny**] [*sequence-number*] **Example:** `Router(config)# route-map CHECK-NEXTHOP permit 20`	Configures a route map and enters route map configuration mode. • In this example, all other IP addresses are permitted by route map CHECK-NEXTHOP.
Step 13	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.
Step 14	**show ip bgp** [*network*] [*network-mask*] **Example:** `Router# show ip bgp`	Displays the entries in the BGP routing table. • Enter this command to view the next-hop addresses for each route. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Example

The following example from the **show ip bgp** command shows the next-hop addresses for each route:

```
BGP table version is 7, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*  10.1.1.0/24      192.168.1.2              0             0 40000 i
*  10.2.2.0/24      192.168.3.2              0             0 50000 i
*> 172.16.1.0/24    0.0.0.0                  0         32768 i
*> 172.17.1.0/24    0.0.0.0                  0         32768
```

Configuring BGP Nonstop Forwarding Awareness Using BGP Graceful Restart

The tasks in this section show how configure BGP Nonstop Forwarding (NSF) awareness using the BGP graceful restart capability. The first task enables BGP NSF globally for all BGP neighbors and suggests a few troubleshooting options. The second task describes how to adjust the BGP graceful restart timers although the default settings are optimal for most network deployments. The next three tasks demonstrate how to enable or disable BGP graceful restart for individual BGP neighbors including peer session templates and peer groups. The final task verifies the local and peer router configuration of BGP NSF.

Enabling BGP Global NSF Awareness Using BGP Graceful Restart

Perform this task to enable BGP NSF awareness globally for all BGP neighbors. BGP NSF awareness is part of the graceful restart mechanism and BGP NSF awareness is enabled by issuing the **bgp graceful-restart** command in router configuration mode. BGP NSF awareness allows NSF-aware routers to support NSF-capable routers during an SSO operation. NSF-awareness is not enabled by default and should be configured on all neighbors that participate in BGP NSF.

Note The configuration of the restart and stale-path timers is not required to enable the BGP graceful restart capability. The default values are optimal for most network deployments, and these values should be adjusted only by an experienced network operator.

Note Configuring both BFD and BGP graceful restart for NSF on a router running BGP may result in suboptimal routing. For more details, see "BFD for BGP".

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp graceful-restart** [**restart-time** *seconds*] [**stalepath-time** *seconds*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**bgp graceful-restart** [**restart-time** *seconds*] [**stalepath-time** *seconds*] **Example:** `Router(config-router)# bgp graceful-restart`	Enables the BGP graceful restart capability and BGP NSF awareness. • If you enter this command after the BGP session has been established, you must restart the session for the capability to be exchanged with the BGP neighbor. • Use this command on the restarting router and all of its peers (NSF-capable and NSF-aware).
Step 5	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

- Troubleshooting Tips, page 312
- What to Do Next, page 313

Troubleshooting Tips

To troubleshoot the NSF feature, use the following commands in privileged EXEC mode, as needed:

- **debug ip bgp** --Displays open messages that advertise the graceful restart capability.
- **debug ip bgp event** --Displays graceful restart timer events, such as the restart timer and the stalepath timer.
- **debug ip bgp updates** --Displays sent and received EOR messages. The EOR message is used by the NSF-aware router to start the stalepath timer, if configured.

- **show ip bgp** --Displays entries in the BGP routing table. The output from this command will display routes that are marked as stale by displaying the letter "S" next to each stale route.
- **show ip bgp neighbor** --Displays information about the TCP and BGP connections to neighbor devices. When enabled, the graceful restart capability is displayed in the output of this command.

What to Do Next

If the **bgp graceful-restart** command has been issued after the BGP session has been established, you must reset by issuing the **clear ip bgp *** command or by reloading the router before graceful restart capabilities will be exchanged. For more information about resetting BGP sessions and using the **clear ip bgp** command, see the "Configuring a Basic BGP Network" module.

Configuring BGP NSF Awareness Timers

Perform this task to adjust the BGP graceful restart timers. There are two BGP graceful restart timers that can be configured. The optional **restart-time** keyword and *seconds* argument determine how long peer routers will wait to delete stale routes before a BGP open message is received. The default value is 120 seconds. The optional **stalepath-time** keyword and *seconds* argument determine how long a router will wait before deleting stale routes after an end of record (EOR) message is received from the restarting router. The default value is 360 seconds.

Note The configuration of the restart and stale-path timers is not required to enable the BGP graceful restart capability. The default values are optimal for most network deployments, and these values should be adjusted only by an experienced network operator.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp graceful-restart** [**restart-time**seconds]
5. **bgp graceful-restart** [**stalepath-time** *seconds*]
6. Router(config-router)# **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**bgp graceful-restart** [**restart-time**seconds] **Example:** `Router(config-router)# bgp graceful-restart restart-time 130`	Enables the BGP graceful restart capability and BGP NSF awareness. • The **restart-time** argument determines how long peer routers will wait to delete stale routes before a BGP open message is received. • The default value is 120 seconds. The configurable range is from 1 to 3600 seconds. **Note** Only the syntax applicable to this step is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 5	**bgp graceful-restart** [**stalepath-time** *seconds*] **Example:** `Router(config-router)# bgp graceful-restart stalepath-time 350`	Enables the BGP graceful restart capability and BGP NSF awareness. • The **stalepath-time** argument determines how long a router will wait before deleting stale routes after an end of record (EOR) message is received from the restarting router. • The default value is 360 seconds. The configurable range is from 1 to 3600 seconds. **Note** Only the syntax applicable to this step is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 6	Router(config-router)# **end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

• What to Do Next, page 314

What to Do Next

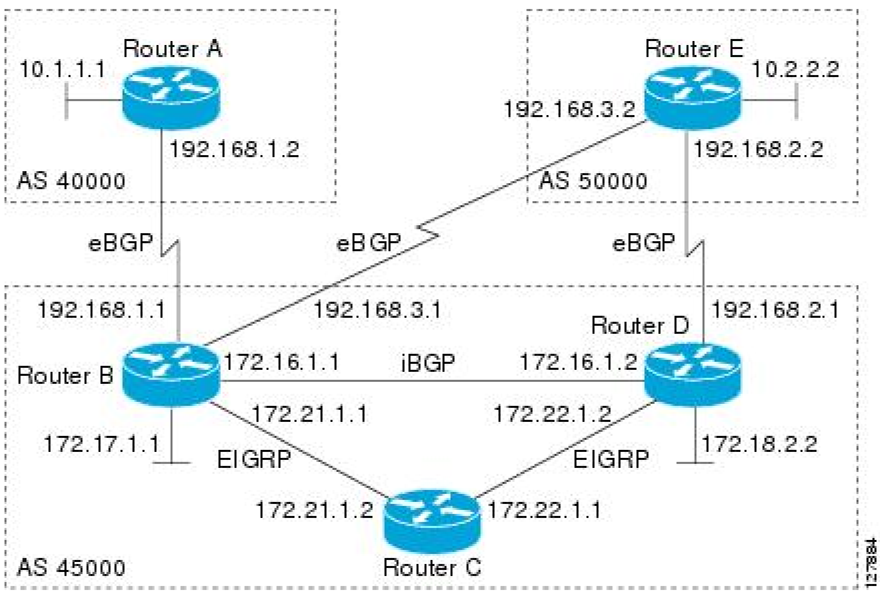
If the **bgp graceful-restart** command has been issued after the BGP session has been established, you must reset the peer sessions by issuing the **clear ip bgp *** command or by reloading the router before graceful restart capabilities will be exchanged. For more information about resetting BGP sessions and using the **clear ip bgp**command, see the "Configuring a Basic BGP Network" module.

Enabling and Disabling BGP Graceful Restart Using BGP Peer Session Templates

Perform this task to enable and disable BGP graceful restart for BGP neighbors using peer session templates. In this task, a BGP peer session template is created, and BGP graceful restart is enabled. A second peer session template is created, and this template is configured to disable BGP graceful restart.

In this example, the configuration is performed at Router B in the figure below and two external BGP neighbors--at Router A and Router E in the figure below--are identified. The first BGP peer at Router A is configured to inherit the first peer session template that enables BGP graceful restart, whereas the second BGP peer at Router E inherits the second template that disables BGP graceful restart. Using the optional **show ip bgp neighbors** command, the status of the BGP graceful restart capability is verified for each BGP neighbor configured in this task.

Figure 29 Network Topology Showing BGP Neighbors



The restart and stale-path timers can be modified only using the global **bgp graceful-restart** command as shown in the Configuring BGP NSF Awareness Timers, page 313. The restart and stale-path timers are set to the default values when BGP graceful restart is enabled for BGP neighbors using peer session templates.

This task requires a Cisco IOS Release 12.2(33)SRC, or 12.2(33)SB.

Note A BGP peer cannot inherit from a peer policy or session template and be configured as a peer group member at the same. BGP templates and BGP peer groups are mutually exclusive.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-session** *session-template-name*
5. **ha-mode graceful-restart** [**disable**]
6. **exit-peer-session**
7. **template peer-session** *session-template-name*
8. **ha-mode graceful-restart** [**disable**]
9. **exit-peer-session**
10. **bgp log-neighbor-changes**
11. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
12. **neighbor** *ip-address* **inherit peer-session** *session-template-number*
13. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
14. **neighbor** *ip-address* **inherit peer-session** *session-template-number*
15. **end**
16. **show ip bgp template peer-session** [*session-template-number*]
17. **show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**template peer-session** *session-template-name* **Example:** `Router(config-router)# template peer-session S1`	Enters session-template configuration mode and creates a peer session template. • In this example, a peer session template named S1 is created.
Step 5	**ha-mode graceful-restart** [**disable**] **Example:** `Router(config-router-stmp)# ha-mode graceful-restart`	Enables the BGP graceful restart capability and BGP NSF awareness. • Use the **disable** keyword to disable BGP graceful restart capability. • If you enter this command after the BGP session has been established, you must restart the session in order for the capability to be exchanged with the BGP neighbor. • In this example, the BGP graceful restart capability is enabled for the peer session template named S1.
Step 6	**exit-peer-session** **Example:** `Router(config-router-stmp)# exit-peer-session`	Exits session-template configuration mode and returns to router configuration mode.
Step 7	**template peer-session** *session-template-name* **Example:** `Router(config-router)# template peer-session S2`	Enters session-template configuration mode and creates a peer session template. • In this example, a peer session template named S2 is created.
Step 8	**ha-mode graceful-restart** [**disable**] **Example:** `Router(config-router-stmp)# ha-mode graceful-restart disable`	Enables the BGP graceful restart capability and BGP NSF awareness. • Use the **disable** keyword to disable BGP graceful restart capability. • If you enter this command after the BGP session has been established, you must restart the session in order for the capability to be exchanged with the BGP neighbor. • In this example, the BGP graceful restart capability is disabled for the peer session template named S2.
Step 9	**exit-peer-session** **Example:** `Router(config-router-stmp)# exit-peer-session`	Exits session-template configuration mode and returns to router configuration mode.

	Command or Action	Purpose
Step 10	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor status changes (up or down) and neighbor resets. • Use this command for troubleshooting network connectivity problems and measuring network stability. Unexpected neighbor resets might indicate high error rates or high packet loss in the network and should be investigated.
Step 11	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 remote-as 40000`	Configures peering with a BGP neighbor in the specified autonomous system. • In this example, the BGP peer at 192.168.1.2 is an external BGP peer because it has a different autonomous system number from the router where the BGP configuration is being entered (see Step 3).
Step 12	**neighbor** *ip-address* **inherit peer-session** *session-template-number* **Example:** `Router(config-router)# neighbor 192.168.1.2 inherit peer-session S1`	Inherits a peer session template. • In this example, the peer session template named S1 is inherited, and the neighbor inherits the enabling of BGP graceful restart.
Step 13	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 remote-as 50000`	Configures peering with a BGP neighbor in the specified autonomous system. • In this example, the BGP peer at 192.168.3.2 is an external BGP peer because it has a different autonomous system number from the router where the BGP configuration is being entered (see Step 3).
Step 14	**neighbor** *ip-address* **inherit peer-session** *session-template-number* **Example:** `Router(config-router)# neighbor 192.168.3.2 inherit peer-session S2`	Inherits a peer session-template. • In this example, the peer session template named S2 is inherited, and the neighbor inherits the disabling of BGP graceful restart.
Step 15	**end** **Example:** `Router(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 16	**show ip bgp template peer-session** [*session-template-number*] **Example:** `Router# show ip bgp template peer-session`	(Optional) Displays locally configured peer session templates. • The output can be filtered to display a single peer policy template with the *session-template-name* argument. This command also supports all standard output modifiers.
Step 17	**show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]] **Example:** `Router# show ip bgp neighbors 192.168.1.2`	(Optional) Displays information about TCP and BGP connections to neighbors. • "Graceful Restart Capability: advertised" will be displayed for each neighbor that has exchanged graceful restart capabilities with this router. • In this example, the output is filtered to display information about the BGP peer at 192.168.1.2.

Examples

The following example shows partial output from the show ip bgp neighbors command for the BGP peer at 192.168.1.2 (Router A in the figure above). Graceful restart is shown as enabled. Note the default values for the restart and stale-path timers. These timers can only be set using the global bgp graceful-restart command.

```
Router# show ip bgp neighbors 192.168.1.2
BGP neighbor is 192.168.1.2,  remote AS 40000, external link
 Inherits from template S1 for session parameters
  BGP version 4, remote router ID 192.168.1.2
  BGP state = Established, up for 00:02:11
  Last read 00:00:23, last write 00:00:27, hold time is 180, keepalive intervals
  Neighbor sessions:
    1 active, is multisession capable
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
    Graceful Restart Capability: advertised
    Multisession Capability: advertised and received
!
Address tracking is enabled, the RIB does have a route to 192.168.1.2
  Connections established 1; dropped 0
  Last reset never
  Transport(tcp) path-mtu-discovery is enabled
  Graceful-Restart is enabled, restart-time 120 seconds, stalepath-time 360 secs
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
```

The following example shows partial output from the **show ip bgp neighbors** command for the BGP peer at 192.168.3.2 (Router E in the figure above). Graceful restart is shown as disabled.

```
Router# show ip bgp neighbors 192.168.3.2
BGP neighbor is 192.168.3.2,  remote AS 50000, external link
 Inherits from template S2 for session parameters
  BGP version 4, remote router ID 192.168.3.2
  BGP state = Established, up for 00:01:41
  Last read 00:00:45, last write 00:00:45, hold time is 180, keepalive intervals
  Neighbor sessions:
```

```
    1 active, is multisession capable
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
!
Address tracking is enabled, the RIB does have a route to 192.168.3.2
  Connections established 1; dropped 0
  Last reset never
  Transport(tcp) path-mtu-discovery is enabled
  Graceful-Restart is disabled
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
```

Enabling BGP Graceful Restart for an Individual BGP Neighbor

Perform this task on Router B in the figure above to enable BGP graceful restart on the internal BGP peer at Router C in the figure above. Under address family IPv4, the neighbor at Router C is identified, and BGP graceful restart is enabled for the neighbor at Router C with the IP address 172.21.1.2. To verify that BGP graceful restart is enabled, the optional **show ip bgp neighbors** command is used.

This task requires a Cisco IOS Release 12.2(33)SRC, 12.2(33)SB, or 15.0(1)M.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
5. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
6. **neighbor** *ip-address* **activate**
7. **neighbor** *ip-address* **ha-mode graceful-restart** [**disable**]
8. **end**
9. **show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.
Step 5	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 172.21.1.2 remote-as 45000`	Configures peering with a BGP neighbor in the specified autonomous system. • In this example, the BGP peer at 172.21.1.2 is an internal BGP peer because it has the same autonomous system number as the router where the BGP configuration is being entered (see Step 3).
Step 6	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 172.21.1.2 activate`	Enables the neighbor to exchange prefixes for the IPv4 address family with the local router. • In this example, the internal BGP peer at 172.21.1.2 is activated.
Step 7	**neighbor** *ip-address* **ha-mode graceful-restart** [**disable**] **Example:** `Router(config-router-af)# neighbor 172.21.1.2 ha-mode graceful-restart`	Enables the BGP graceful restart capability for a BGP neighbor. • Use the **disable** keyword to disable BGP graceful restart capability. • If you enter this command after the BGP session has been established, you must restart the session in order for the capability to be exchanged with the BGP neighbor. • In this example, the BGP graceful restart capability is enabled for the neighbor at 172.21.1.2.
Step 8	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 9	**show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]] **Example:** `Router# show ip bgp neighbors 172.21.1.2`	(Optional) Displays information about TCP and BGP connections to neighbors. • "Graceful Restart Capability: advertised" will be displayed for each neighbor that has exchanged graceful restart capabilities with this router. • In this example, the output is filtered to display information about the BGP peer at 172.21.1.2.

Examples

The following example shows partial output from the **show ip bgp neighbors** command for the BGP peer at 172.21.1.2. Graceful restart is shown as enabled. Note the default values for the restart and stale-path timers. These timers can be set using only the global **bgp graceful-restart** command.

```
Router# show ip bgp neighbors 172.21.1.2
BGP neighbor is 172.21.1.2,  remote AS 45000, internal link
  BGP version 4, remote router ID 172.22.1.1
  BGP state = Established, up for 00:01:01
  Last read 00:00:02, last write 00:00:07, hold time is 180, keepalive intervals
  Neighbor sessions:
    1 active, is multisession capable
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
    Graceful Restart Capability: advertised
    Multisession Capability: advertised and received
!
  Address tracking is enabled, the RIB does have a route to 172.21.1.2
  Connections established 1; dropped 0
  Last reset never
  Transport(tcp) path-mtu-discovery is enabled
  Graceful-Restart is enabled, restart-time 120 seconds, stalepath-time 360 secs
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
```

Disabling BGP Graceful Restart for a BGP Peer Group

Perform this task to disable BGP graceful restart for a BGP peer group. In this task, a BGP peer group is created and graceful restart is disabled for the peer group. A BGP neighbor, 172.16.1.2 at Router D in the figure above, is then identified and added as a peer group member and inherits the configuration associated with the peer group, which, in this example, disables BGP graceful restart.

This task requires a Cisco IOS Release 12.2(33)SRC, 12.2(33)SB, or 15.0(1)M.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
5. **neighbor** *peer-group-name* **peer-group**
6. **neighbor** *peer-group-name* **remote-as** *autonomous-system-number*
7. **neighbor** *peer-group-name* **ha-mode graceful-restart** [**disable**]
8. **neighbor** *ip-address* **peer-group** *peer-group-name*
9. **end**
10. **show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.

	Command or Action	Purpose
Step 5	**neighbor** *peer-group-name* **peer-group** **Example:** `Router(config-router-af)# neighbor PG1 peer-group`	Creates a BGP peer group. • In this example, the peer group named PG1 is created.
Step 6	**neighbor** *peer-group-name* **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor PG1 remote-as 45000`	Configures peering with a BGP peer group in the specified autonomous system. • In this example, the BGP peer group named PG1 is added to the IPv4 multiprotocol BGP neighbor table of the local router.
Step 7	**neighbor** *peer-group-name* **ha-mode graceful-restart** [**disable**] **Example:** `Router(config-router-af)# neighbor PG1 ha-mode graceful-restart disable`	Enables the BGP graceful restart capability for a BGP neighbor. • Use the **disable** keyword to disable BGP graceful restart capability. • If you enter this command after the BGP session has been established, you must restart the session for the capability to be exchanged with the BGP neighbor. • In this example, the BGP graceful restart capability is disabled for the BGP peer group named PG1.
Step 8	**neighbor** *ip-address* **peer-group** *peer-group-name* **Example:** `Router(config-router-af)# neighbor 172.16.1.2 peer-group PG1`	Assigns the IP address of a BGP neighbor to a peer group. • In this example, the BGP neighbor peer at 172.16.1.2 is configured as a member of the peer group named PG1.
Step 9	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 10	**show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]] **Example:** `Router# show ip bgp neighbors 172.16.1.2`	(Optional) Displays information about TCP and BGP connections to neighbors. • In this example, the output is filtered to display information about the BGP peer at 172.16.1.2 and the "Graceful-Restart is disabled" line shows that the graceful restart capability is disabled for this neighbor.

Examples

The following example shows partial output from the **show ip bgp neighbors** command for the BGP peer at 172.16.1.2. Graceful restart is shown as disabled. Note the default values for the restart and stale-path timers. These timers can be set using only the global **bgp graceful-restart** command.

```
Router# show ip bgp neighbors 172.16.1.2
BGP neighbor is 172.16.1.2,  remote AS 45000, internal link
 Member of peer-group PG1 for session parameters
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Idle
  Neighbor sessions:
    0 active, is multisession capable
!
Address tracking is enabled, the RIB does have a route to 172.16.1.2
  Connections established 0; dropped 0
  Last reset never
  Transport(tcp) path-mtu-discovery is enabled
  Graceful-Restart is disabled
```

Verifying the Configuration of BGP Nonstop Forwarding Awareness

Use the following steps to verify the local configuration of BGP NSF awareness on a router and to verify the configuration of NSF awareness on peer routers in a BGP network.

SUMMARY STEPS

1. **enable**
2. **show running-config** [*options*]
3. **show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]]

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Router> enable
```

Step 2 **show running-config** [*options*]

Displays the running configuration on the local router. The output will display the configuration of the **bgp graceful-restart** command in the BGP section. Repeat this command on all BGP neighbor routers to verify that all BGP peers are configured for BGP NSF awareness. In this example, BGP graceful restart is enabled globally and the external neighbor at 192.168.1.2 is configured to be a BGP peer and will have the BGP graceful restart capability enabled.

Example:

```
Router# show running-config
.
.
.
router bgp 45000
 bgp router-id 172.17.1.99
 bgp log-neighbor-changes
```

```
 bgp graceful-restart restart-time 130
 bgp graceful-restart stalepath-time 350
 bgp graceful-restart
 timers bgp 70 120
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.1.2 activate
.
.
.
```

Step 3 **show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]]

Displays information about TCP and BGP connections to neighbors. "Graceful Restart Capability: advertised" will be displayed for each neighbor that has exchanged graceful restart capabilities with this router. In Cisco IOS Releases 12.2(33)SRC, 12.2(33)SB, or later releases, the ability to enable or disable the BGP graceful restart capability for an individual BGP neighbor, peer group or peer session template was introduced and output was added to this command to show the BGP graceful restart status.

The following partial output example using a Cisco IOS Release 12.2(33)SRC image, displays the graceful restart information for internal BGP neighbor 172.21.1.2 at Router C in the figure above. Note the "Graceful-Restart is enabled" message.

Example:

```
Router# show ip bgp neighbors 172.21.1.2
BGP neighbor is 172.21.1.2,  remote AS 45000, internal link
  BGP version 4, remote router ID 172.22.1.1
  BGP state = Established, up for 00:01:01
  Last read 00:00:02, last write 00:00:07, hold time is 180, keepalive intervals
  Neighbor sessions:
    1 active, is multisession capable
  Neighbor capabilities:
    Route refresh: advertised and received(new)
    Address family IPv4 Unicast: advertised and received
    Graceful Restart Capability: advertised
    Multisession Capability: advertised and received
!
  Address tracking is enabled, the RIB does have a route to 172.21.1.2
  Connections established 1; dropped 0
  Last reset never
  Transport(tcp) path-mtu-discovery is enabled
  Graceful-Restart is enabled, restart-time 120 seconds, stalepath-time 360 secs
```

Configuring BGP Route Dampening

The tasks in this section configure and monitor BGP route dampening. Route dampening is designed to minimize the propagation of flapping routes across an internetwork. A route is considered to be flapping when its availability alternates repeatedly.

- Enabling and Configuring BGP Route Dampening, page 326
- Monitoring and Maintaining BGP Route Dampening, page 328

Enabling and Configuring BGP Route Dampening

Perform this task to enable and configure BGP route dampening.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** as-*number*
4. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
5. **bgp dampening** [*half-life reuse suppress max-suppress-time*] [**route-map** *map-name*]
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** as-*number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the VRF instance to associate with subsequent IPv4 address family configuration mode commands.

	Command or Action	Purpose
Step 5	**bgp dampening** [*half-life reuse suppress max-suppress-time*] [**route-map** *map-name*] **Example:** `Router(config-router-af)# bgp dampening 30 1500 10000 120`	Enables BGP route dampening and changes the default values of route dampening factors. • The *half-life*, *reuse*, *suppress*, and *max-suppress-time* arguments are all position dependent; if one argument is entered then all the arguments must be entered. • Use the **route-map** keyword and *map-name* argument to control where BGP route dampening is enabled.
Step 6	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and enters privileged EXEC mode.

Monitoring and Maintaining BGP Route Dampening

Perform the steps in this task as required to monitor and maintain BGP route dampening.

SUMMARY STEPS

1. **enable**
2. **show ip bgp flap-statistics** [**regexp** *regexp* | **filter-list** *access-list* | *ip-address mask* [**longer-prefix**]]
3. **clear ip bgp flap-statistics** [*neighbor-address* [*ipv4-mask*]] [**regexp** *regexp* | **filter-list** *extcom-number*]
4. **show ip bgp dampened-paths**
5. **clear ip bgp** [**ipv4** {**multicast** | **unicast**} | **ipv6**{**multicast** | **unicast**} | **vpnv4 unicast**] **dampening** [*neighbor-address*] [*ipv4-mask*]

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Router> enable
```

Step 2 **show ip bgp flap-statistics** [**regexp** *regexp* | **filter-list** *access-list* | *ip-address mask* [**longer-prefix**]]

Use this command to monitor the flaps of all the paths that are flapping. The statistics will be deleted once the route is not suppressed and is stable for at least one half-life.

Example:

```
Router# show ip bgp flap-statistics
BGP table version is 10, local router ID is 172.17.232.182
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
```

```
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          From            Flaps Duration Reuse    Path
*d 10.0.0.0         172.17.232.177  4     00:13:31 00:18:10 100
*d 10.2.0.0         172.17.232.177  4     00:02:45 00:28:20 100
```

Step 3 **clear ip bgp flap-statistics** [*neighbor-address* [*ipv4-mask*]] [**regexp** *regexp* | **filter-list** *extcom-number*]

Use this command to clear the accumulated penalty for routes that are received on a router that has BGP dampening enabled. If no arguments or keywords are specified, flap statistics are cleared for all routes. Flap statistics are also cleared when the peer is stable for the half-life time period. After the BGP flap statistics are cleared, the route is less likely to be dampened.

Example:

```
Router# clear ip bgp flap-statistics 172.17.232.177
```

Step 4 **show ip bgp dampened-paths**

Use this command to monitor the flaps of all the paths that are flapping. The statistics will be deleted once the route is not suppressed and is stable for at least one half-life.

Example:

```
Router# show ip bgp dampened-paths
BGP table version is 10, local router ID is 172.29.232.182
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          From              Reuse   Path
*d 10.0.0.0         172.16.232.177    00:18:4 100 ?
*d 10.2.0.0         172.16.232.177    00:28:5 100 ?
```

Step 5 **clear ip bgp** [**ipv4** {**multicast** | **unicast**} | **ipv6**{**multicast** | **unicast**} | **vpnv4 unicast**] **dampening** [*neighbor-address*] [*ipv4-mask*]

Use this command to clear stored route dampening information. If no keywords or arguments are entered, route dampening information for the entire routing table is cleared. The following example clears route dampening information for VPNv4 address family prefixes from network 192.168.10.0/24, and unsuppresses its suppressed routes.

Example:

```
Router# clear ip bgp vpnv4 unicast dampening 192.168.10.0 255.255.255.0
```

Decreasing BGP Convergence Time Using BFD

BFD support for BGP was introduced in Cisco IOS Releases 12.0(31)S, 12.4(4)T, 12.2(33)SRA, 12.2(33)SXH, 12.2(33)SB, and later releases. You start a BFD process by configuring BFD on the interface. When the BFD process is started, no entries are created in the adjacency database, in other words, no BFD control packets are sent or received. The adjacency creation takes places once you have configured BFD support for the applicable routing protocols. The first two tasks must be configured to implement BFD support for BGP to reduce the BGP convergence time. The third task is an optional task to help monitor or troubleshoot BFD.

See also the "Configuring BGP Neighbor Session Options" chapter, the section "Configuring BFD for BGP IPv6 Neighbors."

Prerequisites

- Cisco Express Forwarding (CEF) and IP routing must be enabled on all participating routers.
- BGP must be configured on the routers before BFD is deployed. You should implement fast convergence for the routing protocol that you are using. See the IP routing documentation for your version of Cisco IOS software for information on configuring fast convergence.

Restrictions

- For the current Cisco implementation of BFD support for BGP in Cisco IOS Releases 12.0(31)S, 12.4(4)T, 12.2(33)SRA, 12.2(33)SXH, and 12.2(33)SB, BFD is supported only for IPv4 networks, and only asynchronous mode is supported. In asynchronous mode, either BFD peer can initiate a BFD session.
- BFD works only for directly-connected neighbors. BFD neighbors must be no more than one IP hop away. Multihop configurations are not supported.
- Configuring both BFD and BGP graceful restart for NSF on a router running BGP may result in suboptimal routing. For more details, see the BFD for BGP, page 300.

Configuring BFD Session Parameters on the Interface

The steps in this procedure show how to configure BFD on the interface by setting the baseline BFD session parameters on an interface. Repeat the steps in this procedure for each interface over which you want to run BFD sessions to BFD neighbors.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface FastEthernet 6/0`	Enters interface configuration mode.
Step 4	**bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* **Example:** `Router(config-if)# bfd interval 50 min_rx 50 multiplier 5`	Enables BFD on the interface.
Step 5	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode.

Configuring BFD Support for BGP

Perform this task to configure BFD support for BGP, so that BGP is a registered protocol with BFD and will receive forwarding path detection failure messages from BFD.

- BGP must be running on all participating routers.
- The baseline parameters for BFD sessions on the interfaces over which you want to run BFD sessions to BFD neighbors must be configured. See "Configuring BFD Session Parameters on the Interface" for more information.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **fall-over bfd**
5. **end**
6. **show bfd neighbors** [**details**]
7. **show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp tag1`	Specifies a BGP process and enters router configuration mode.
Step 4	**neighbor** *ip-address* **fall-over bfd** **Example:** `Router(config-router)# neighbor 172.16.10.2 fall-over bfd`	Enables BFD support for fallover.
Step 5	**end** **Example:** `Router(config-router)# end`	Returns the router to privileged EXEC mode.
Step 6	**show bfd neighbors** [**details**] **Example:** `Router# show bfd neighbors detail`	Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 7	**show ip bgp neighbors** [*ip-address* [**received-routes** | **routes** | **advertised-routes** | **paths** [*regexp*] | **dampened-routes** | **flap-statistics**| **received prefix-filter**| **policy**[**detail**]]] **Example:** `Router# show ip bgp neighbors`	Displays information about BGP and TCP connections to neighbors.

Monitoring and Troubleshooting BFD for Cisco 7600 Series Routers

To monitor or troubleshoot BFD on Cisco 7600 series routers, perform one or more of the steps in this section.

SUMMARY STEPS

1. **enable**
2. **show bfd neighbors** [**details**]
3. **debug bfd** [**event** | **packet** | **ipc-error** | **ipc-event** | **oir-error** | **oir-event**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show bfd neighbors** [**details**] **Example:** `Router# show bfd neighbors details`	(Optional) Displays the BFD adjacency database. • The **details** keyword shows all BFD protocol parameters and timers per neighbor.
Step 3	**debug bfd** [**event** | **packet** | **ipc-error** | **ipc-event** | **oir-error** | **oir-event**] **Example:** `Router# debug bfd packet`	(Optional) Displays debugging information about BFD packets.

• What to Do Next, page 333

What to Do Next

For more information about configuring BFD support for another routing protocol see the "Bidirectional Forwarding Detection" configuration guide.

Enabling BGP MIB Support

SNMP notifications can be configured on the router and GET operations can be performed from an external management station only after BGP SNMP support is enabled. Perform this task on a router to configure SNMP notifications for the BGP MIB.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server enable traps bgp** [[**state-changes** [**all**] [**backward-trans**] [**limited**]] | [**threshold prefix**]]
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server enable traps bgp** [[**state-changes** [**all**] [**backward-trans**] [**limited**]] | [**threshold prefix**]] **Example:** `Router# snmp-server enable traps bgp`	Enables BGP support for SNMP operations. Entering this command with no keywords or arguments enables support for all BGP events. • The **state-changes** keyword is used to enable support for FSM transition events. • The **all** keyword enables support for FSM transitions events. • The **backward-trans** keyword enables support only for backward transition state change events. • The **limited** keyword enables support for backward transition state changes and established state events. • The **threshold**and **prefix** keywords are used to enable notifications when the configured maximum prefix limit is reached on the specified peer.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode, and enters privileged EXEC mode.

Configuring BGP Support for MTR

Before performing the following tasks, you must have configured MTR topologies. For more details, see the "Multi-Topology Routing" feature in Cisco IOS Release 12.2(33)SRB.

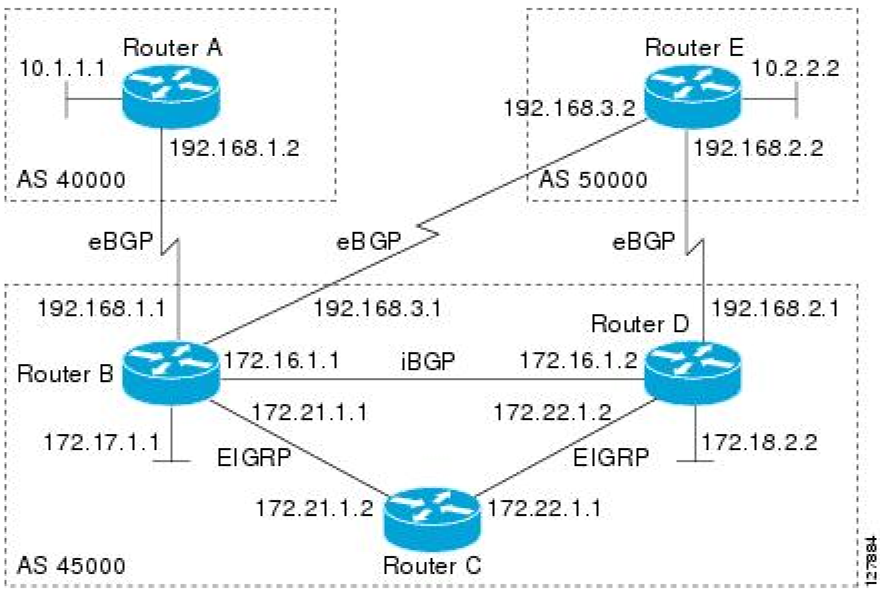
• Activating an MTR Topology Using BGP, page 335

- Importing Routes from an MTR Topology Using BGP, page 340

Activating an MTR Topology Using BGP

Perform this task to activate an MTR topology inside an address family using BGP. This task is configured on Router B in the figure below and must also be configured on Router D and Router E. In this task, a scope hierarchy is configured to apply globally and a neighbor is configured under router scope configuration mode. Under the IPv4 unicast address family, an MTR topology that applies to video traffic is activated for the specified neighbor. There is no interface configuration mode for BGP topologies.

Figure 30 BGP Network Diagram



The BGP CLI has been modified to provide backwards compatibility for pre-MTR BGP configuration and to provide a hierarchical implementation of MTR. A new configuration hierarchy, named scope, has been introduced into the BGP protocol. To implement MTR for BGP, the scope hierarchy is required, but the scope hierarchy is not limited to MTR use. The scope hierarchy introduces some new configuration modes such as router scope configuration mode. Router scope configuration mode is entered by configuring the **scope** command in router configuration mode, and a collection of routing tables is created when this command is entered. The following shows the hierarchy levels that are used when configuring BGP for MTR implementation:

```
router bgp <
autonomous-system-number
>
 ! global commands

 scope {global | vrf <
vrf-name
>}
  ! scoped commands

  address-family {<
afi
>} [<
safi
>]
   ! address family specific commands
```

```
   topology {<
topology-name
> | base}
    ! topology specific commands
```

Before using BGP to support MTR, you should be familiar with all the concepts documented in the BGP Support for MTR, page 302.

- You must be running a Cisco IOS Release 12.2(33)SRB, or later release, on any routers configured for MTR.
- A global MTR topology configuration has been configured and activated.
- IP routing and CEF are enabled.

Note

- Redistribution within a topology is permitted. Redistribution from one topology to another is not permitted. This restriction is designed to prevent routing loops. You can use topology translation or topology import functionality to move routes from one topology to another.
- Only the IPv4 address family (multicast and unicast) is supported.
- Only a single multicast topology can be configured, and only the base topology can be specified if a multicast topology is created.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **scope** {**global** | **vrf** *vrf-name*}
5. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
6. **neighbor** {*ip-address*| *peer-group-name*} **transport**{**connection-mode** {**active** | **passive**} | **path-mtu-discovery** | **multi-session** | **single-session**}
7. **address-family ipv4** [**mdt** | **multicast** | **unicast**]
8. **topology** {**base**| *topology-name*}
9. **bgp tid** *number*
10. **neighbor** *ip-address* **activate**
11. **neighbor** {*ip-address*| *peer-group-name*} **translate-topology** *number*
12. **end**
13. **clear ip bgp topology** {***** | *topology-name*} {*as-number* | **dampening** [*network-address* [*network-mask*]] | **flap-statistics** [*network-address* [*network-mask*]] | **peer-group** *peer-group-name* | **table-map** | **update-group** [*number* | *ip-address*]} [**in** [**prefix-filter**] | **out**| **soft** [**in** [**prefix-filter**] | **out**]]
14. **show ip bgp topology** {***** | *topology*} **summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**scope** {**global** | **vrf** *vrf-name*} **Example:** `Router(config-router)# scope global`	Defines the scope to the BGP routing process and enters router scope configuration mode. • BGP general session commands that apply to a single network, or a specified VRF, are entered in this configuration mode. • Use the **global** keyword to specify that BGP uses the global routing table. • Use the **vrf** keyword and *vrf-name* argument to specify that BGP uses a specific VRF routing table. The VRF must already exist.
Step 5	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-scope)# neighbor 172.16.1.2 remote-as 45000`	Adds the IP address of the neighbor in the specified autonomous system to the multiprotocol BGP neighbor table of the local router.

	Command or Action	Purpose
Step 6	**neighbor** {*ip-address*| *peer-group-name*} **transport**{**connection-mode** {**active** | **passive**} | **path-mtu-discovery** | **multi-session** | **single-session**} **Example:** `Router(config-router-scope)# neighbor 172.16.1.2 transport multi-session`	Enables a TCP transport session option for a BGP session. • Use the **connection-mode** keyword to specify the type of connection, either active or passive. • Use the **path-mtu-discovery** keyword to enable TCP transport path maximum transmission unit (MTU) discovery. • Use the **multi-session** keyword to specify a separate TCP transport session for each address family. • Use the **single-session** keyword to specify that all address families use a single TCP transport session.
Step 7	**address-family ipv4** [**mdt** | **multicast** | **unicast**] **Example:** `Router(config-router-scope)# address-family ipv4`	Specifies the IPv4 address family and enters router scope address family configuration mode. • Use the **mdt** keyword to specify IPv4 MDT address prefixes. • Use the **multicast** keyword to specify IPv4 multicast address prefixes. • Use the **unicast** keyword to specify the IPv4 unicast address family. By default, the router is placed in address family configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • Non-topology-specific configuration parameters are configured in this configuration mode.
Step 8	**topology** {**base**| *topology-name*} **Example:** `Router(config-router-scope-af)# topology VIDEO`	Configures the topology instance in which BGP will route class-specific or base topology traffic, and enters router scope address family topology configuration mode.
Step 9	**bgp tid** *number* **Example:** `Router(config-router-scope-af-topo)# bgp tid 100`	Associates a BGP routing process with the specified topology ID. • Each topology must be configured with a unique topology ID.

Step 10	**neighbor** *ip-address* **activate**
	
	Example:
	
	`Router(config-router-scope-af-topo)#`
	
	`neighbor 172.16.1.2 activate`
		Enables the BGP neighbor to exchange prefixes for the NSAP address family with the local router.
		Note If you have configured a peer group as a BGP neighbor, you do not use this command because peer groups are automatically activated when any peer group parameter is configured.

	Command or Action	Purpose
Step 11	**neighbor** {*ip-address*| *peer-group-name*} **translate-topology** *number* **Example:** `Router(config-router-scope-af-topo)# neighbor 172.16.1.2 translate-topology 200`	(Optional) Configures BGP to install routes from a topology on another router to a topology on the local router. • The topology ID is entered for the *number* argument to identify the topology on the router.
Step 12	**end** **Example:** `Router(config-router-scope-af-topo)# end`	(Optional) Exits router scope address family topology configuration mode and returns to privileged EXEC mode.
Step 13	**clear ip bgp topology** {***** | *topology-name*} {*as-number* | **dampening** [*network-address* [*network-mask*]] | **flap-statistics** [*network-address* [*network-mask*]] | **peer-group** *peer-group-name* | **table-map** | **update-group** [*number* | *ip-address*]} [**in** [**prefix-filter**] | **out**| **soft** [**in** [**prefix-filter**] | **out**]] **Example:** `Router# clear ip bgp topology VIDEO 45000`	Resets BGP neighbor sessions under a specified topology or all topologies.
Step 14	**show ip bgp topology** {***** | *topology*} **summary** **Example:** `Router# show ip bgp topology VIDEO summary`	(Optional) Displays BGP information about a topology. • Most standard BGP keywords and arguments can be entered following the topology keyword. **Note** Only the syntax required for this task is shown. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following example shows summary output for the **show ip bgp topology** command and the VIDEO topology:

```
Router# show ip bgp topology VIDEO summary
BGP router identifier 192.168.3.1, local AS number 45000
BGP table version is 1, main routing table version 1
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.16.1.2      4 45000     289     289        1    0    0 04:48:44        0
192.168.3.2     4 50000       3       3        1    0    0 00:00:27        0
```

• What to Do Next, page 339

What to Do Next

Repeat this task for every topology that you want to enable, and repeat this configuration on all neighbor routers that are to use the topologies. If you want to import routes from one MTR topology to another on the same router, proceed to the next task.

Importing Routes from an MTR Topology Using BGP

Perform this task to import routes from one MTR topology to another on the same router, when multiple topologies are configured on the same router. In this task, a prefix list is defined to permit prefixes from the 10.2.2.0 network, and this prefix list is used with a route map to filter routes moved from the imported topology. A global scope is configured, address family IPv4 is entered, the VIDEO topology is specified, the VOICE topology is imported, and the routes are filtered using the route map named 10NET.

- You must be running a Cisco IOS Release 12.2(33)SRB, or later release, on any routers configured for MTR.
- A global topology configuration has been configured and activated.
- IP routing and CEF are enabled.

Note

- Redistribution within a topology is permitted. Redistribution from one topology to another is not permitted. This restriction is designed to prevent routing loops from occurring. You can use topology translation or topology import functionality to move routes from one topology to another.
- Only the IPv4 address family (multicast and unicast) is supported.
- Only a single multicast topology can be configured, and only the base topology can be specified if a multicast topology is created.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length*| **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*]
4. **route-map** *map-name* [**permit** | **deny**] [*sequence-number*]
5. **match ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...*| *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]}
6. **exit**
7. **router bgp** *autonomous-system-number*
8. **scope** {**global** | **vrf** *vrf-name*}
9. **address-family ipv4** [**mdt** | **multicast** | **unicast**]
10. **topology** {**base**| *topology-name*}
11. **import topology** {**base**| *topology-name*}[**route-map** *map-name*]
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network* / *length*| **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list TEN permit 10.2.2.0/24`	Configures an IP prefix list. • In this example, prefix list TEN permits advertising of the 10.2.2.0/24 prefix depending on a match set by the **match ip address** command.
Step 4	**route-map** *map-name* [**permit** | **deny**] [*sequence-number*] **Example:** `Router(config)# route-map 10NET`	Creates a route map and enters route map configuration mode. • In this example, the route map named 10NET is created.
Step 5	**match ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...*| *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]} **Example:** `Router(config-route-map)# match ip address prefix-list TEN`	Configures the route map to match a prefix that is permitted by a standard access list, an extended access list, or a prefix list. • In this example, the route map is configured to match prefixes permitted by prefix list TEN.
Step 6	**exit** **Example:** `Router(config-route-map)# exit`	Exits route map configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 7	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 50000`	Enters router configuration mode to create or configure a BGP routing process.
Step 8	**scope** {**global** | **vrf** *vrf-name*} **Example:** `Router(config-router)# scope global`	Defines the scope to the BGP routing process and enters router scope configuration mode. • BGP general session commands that apply to a single network, or a specified VRF, are entered in this configuration mode. • Use the **global** keyword to specify that BGP uses the global routing table. • Use the **vrf** keyword and *vrf-name* argument to specify that BGP uses a specific VRF routing table. The VRF must already exist.
Step 9	**address-family ipv4** [**mdt** | **multicast** | **unicast**] **Example:** `Router(config-router-scope)# address-family ipv4`	Enters router scope address family configuration mode to configure an address family session under BGP. • Non-topology-specific configuration parameters are configured in this configuration mode.
Step 10	**topology** {**base**| *topology-name*} **Example:** `Router(config-router-scope-af)# topology VIDEO`	Configures the topology instance in which BGP will route class-specific or base topology traffic, and enters router scope address family topology configuration mode.
Step 11	**import topology** {**base**| *topology-name*}[**route-map** *map-name*] **Example:** `Router(config-router-scope-af-topo)# import topology VOICE route-map 10NET`	(Optional) Configures BGP to move routes from one topology to another on the same router. • The **route-map** keyword can be used to filter routes that moved between topologies.
Step 12	**end** **Example:** `Router(config-router-scope-af-topo)# end`	(Optional) Exits router scope address family topology configuration mode, and returns to privileged EXEC mode.

Where to Go Next

- If you want to connect to an external service provider and use other external BGP features, see the "Connecting to a Service Provider Using External BGP" module.
- If you want to configure some internal BGP features, see the "Configuring Internal BGP Features" chapter of the BGP section of the Cisco IOS IP Routing Protocols Configuration Guide.
- If you want to configure BGP neighbor session options, see the "Configuring BGP Neighbor Session Options" module.

Additional References

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
Overview of Cisco BGP conceptual information with links to all the individual BGP modules	"Cisco BGP Overview" module of the *Cisco IOS IP Routing Protocols Configuration Guide*.
Conceptual and configuration details for basic BGP tasks.	"Configuring a Basic BGP Network" module of the *Cisco IOS IP Routing Protocols Configuration Guide*.
Information about SNMP and SNMP operations.	"Configuring SNMP Support" section of the *Cisco IOS Network Management Configuration Guide*.

Standards

Standard	Title
MDT SAFI	MDT SAFI

MIBs

MIB	MIBs Link
CISCO-BGP4-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1657	*Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2*
RFC 1771	*A Border Gateway Protocol 4 (BGP-4)*
RFC 1772	*Application of the Border Gateway Protocol in the Internet*
RFC 1773	*Experience with the BGP Protocol*
RFC 1774	*BGP-4 Protocol Analysis*
RFC 1930	*Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)*
RFC 2519	*A Framework for Inter-Domain Route Aggregation*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 2918	*Route Refresh Capability for BGP-4*
RFC 3392	*Capabilities Advertisement with BGP-4*
RFC 4724	*Graceful Restart Mechanism for BGP*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Configuring Advanced BGP Features

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 19 Feature Information for Configuring Advanced BGP Features

Feature Name	Releases	Feature Configuration Information
BGP Graceful Restart per Neighbor	12.2(33)SRC 12.2(33)SB 15.0(1)M 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Graceful Restart per Neighbor feature enables or disables the BGP graceful restart capability for an individual BGP neighbor, including using peer session templates and BGP peer groups. In Cisco IOS Release 12.2(33)SB, platform support includes the Cisco 10000 series routers. The following commands were introduced or modified by this feature: **ha-mode graceful-restart**, **neighbor ha-mode graceful-restart**, **show ip bgp neighbors**.
BGP MIB Support Enhancements	12.0(26)S 12.2(25)S 12.3(7)T 12.2(33)SRA 12.2(33)SXH	The BGP MIB Support Enhancements feature introduced support in the CISCO-BGP4-MIB for new SNMP notifications. The following command was introduced in this feature: **snmp-server enable traps bgp**.

Feature Name	Releases	Feature Configuration Information
BGP Nonstop Forwarding (NSF) Awareness	12.2(15)T 15.0(1)S	Nonstop Forwarding (NSF) awareness allows a router to assist NSF-capable neighbors to continue forwarding packets during a Stateful Switchover (SSO) operation. The BGP Nonstop Forwarding Awareness feature allows an NSF-aware router that is running BGP to forward packets along routes that are already known for a router that is performing an SSO operation. This capability allows the BGP peers of the failing router to retain the routing information that is advertised by the failing router and continue to use this information until the failed router has returned to normal operating behavior and is able to exchange routing information. The peering session is maintained throughout the entire NSF operation. The following commands were introduced or modified by this feature: **bgp graceful-restart**, **show ip bgp**, **show ip bgp neighbors**.
BGP Selective Address Tracking	12.4(4)T 12.2(33)SRB	The BGP Selective Address Tracking feature introduces the use of a route map for next-hop route filtering and fast session deactivation. Selective next-hop filtering uses a route map to selectively define routes to help resolve the BGP next hop, or a route map can be used to determine if a peering session with a BGP neighbor should be reset when a route to the BGP peer changes. The following commands were modified by this feature: **bgp nexthop**, **neighbor fall-over**.

Feature Name	Releases	Feature Configuration Information
BGP Support for BFD	12.0(31)S 12.4(4)T 12.2(33)SRA 12.2(33)SXH 12.2(33)SB 15.0(1)S	Bidirectional Forwarding Detection (BFD) is a detection protocol designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols. In addition to fast forwarding path failure detection, BFD provides a consistent failure detection method for network administrators. Because the network administrator can use BFD to detect forwarding path failures at a uniform rate, rather than the variable rates for different routing protocol hello mechanisms, network profiling and planning will be easier, and reconvergence time will be consistent and predictable. The main benefit of implementing BFD for BGP is a significantly faster reconvergence time. The following commands were introduced or modified by this feature: **bfd**, **neighbor fall-over**, **show bfd neighbors**, **show ip bgp neighbors**.

Feature Name	Releases	Feature Configuration Information
BGP Support for MTR	12.2(33)SRB	BGP support for MTR introduces a new configuration hierarchy and command-line interface (CLI) commands to support multi-topology routing (MTR) topologies. The new configuration hierarchy, or scope, can be implemented by BGP independently of MTR. MTR allows the configuration of service differentiation through class-based forwarding. MTR supports multiple unicast topologies and a separate multicast topology. A topology is a subset of the underlying network (or base topology) characterized by an independent set of Network Layer Reachability Information (NLRI). In 12.2(33)SRB, this feature was introduced on the Cisco 7600. The following commands were introduced or modified by this feature: **address-family ipv4 (BGP)**, **bgp tid**, **clear ip bgp topology**, **import topology**, **neighbor translate-topology**, **neighbor transport**, **scope**, **show ip bgp topology**, **topology (BGP).**

Feature Name	Releases	Feature Configuration Information
BGP Support for Next-Hop Address Tracking	12.0(29)S 12.3(14)T 12.2(33)SXH 15.0(1)S	The BGP Support for Next-Hop Address Tracking feature is enabled by default when a supporting Cisco IOS software image is installed. BGP next-hop address tracking is event driven. BGP prefixes are automatically tracked as peering sessions are established. Next-hop changes are rapidly reported to the BGP routing process as they are updated in the RIB. This optimization improves overall BGP convergence by reducing the response time to next-hop changes for routes installed in the RIB. When a bestpath calculation is run in between BGP scanner cycles, only next-hop changes are tracked and processed. The following command was introduced in this feature: **bgp nexthop**.

BGP Link Bandwidth

The BGP (Border Gateway Protocol) Link Bandwidth feature is used to advertise the bandwidth of an autonomous system exit link as an extended community. This feature is configured for links between directly connected external BGP (eBGP) neighbors. The link bandwidth extended community attribute is propagated to iBGP peers when extended community exchange is enabled. This feature is used with BGP multipath features to configure load balancing over links with unequal bandwidth.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for BGP Link Bandwidth

- BGP load balancing or multipath load balancing must be configured before BGP Link Bandwidth feature is enabled.
- BGP extended community exchange must be enabled between iBGP neighbors to which the link bandwidth attribute is to be advertised.
- Cisco Express Forwarding or distributed Cisco Express Forwarding must be enabled on all participating routers.

Restrictions for BGP Link Bandwidth

- The BGP Link Bandwidth feature can be configured only under IPv4 and VPNv4 address family sessions.
- BGP can originate the link bandwidth community only for directly connected links to eBGP neighbors.
- Both iBGP and eBGP load balancing are supported in IPv4 and VPNv4 address families. However, eiBGP load balancing is supported only in VPNv4 address families.

Information About BGP Link Bandwidth

BGP Link Bandwidth Overview

The BGP Link Bandwidth feature is used to enable multipath load balancing for external links with unequal bandwidth capacity. This feature is enabled under an IPv4 or VPNv4 address family session by entering the **bgp dmzlink-bw** command. This feature supports iBGP, eBGP multipath load balancing, and eiBGP multipath load balancing in Multiprotocol Label Switching (MPLS) VPNs. When this feature is enabled, routes learned from directly connected external neighbor are propagated through the internal BGP (iBGP) network with the bandwidth of the source external link.

The link bandwidth extended community indicates the preference of an autonomous system exit link in terms of bandwidth. This extended community is applied to external links between directly connected eBGP peers by entering the **neighbor dmzlink-bw** command. The link bandwidth extended community attribute is propagated to iBGP peers when extended community exchange is enabled with the **neighbor send-community** command.

Link Bandwidth Extended Community Attribute

The link bandwidth extended community attribute is a 4-byte value that is configured for a link on the demilitarized zone (DMZ) interface that connects two single hop eBGP peers. The link bandwidth extended community attribute is used as a traffic sharing value relative to other paths while traffic is being forwarded. Two paths are designated as equal for load balancing if the weight, local-pref, as-path length, Multi Exit Discriminator (MED), and Interior Gateway Protocol (IGP) costs are the same.

Benefits of the BGP Link Bandwidth Feature

The BGP Link Bandwidth feature allows BGP to be configured to send traffic over multiple iBGP or eBGP learned paths where the traffic that is sent is proportional to the bandwidth of the links that are used to exit the autonomous system. The configuration of this feature can be used with eBGP and iBGP multipath features to enable unequal cost load balancing over multiple links. Unequal cost load balancing over links with unequal bandwidth was not possible in BGP before the BGP Link Bandwidth feature was introduced.

How to Configure BGP Link Bandwidth

• Configuring and Verifying BGP Link Bandwidth, page 353

Configuring and Verifying BGP Link Bandwidth

To configure the BGP Link Bandwidth feature, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4**
5. **address-family ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*]
6. **bgp dmzlink-bw**
7. **neighbor** *ip-address* **dmzlink-bw**
8. **neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**]
9. **end**
10. **show ip bgp** *ip-address* [**longer-prefixes** [**injected**] | **shorter-prefixes** [*mask-length*]]
11. **show ip route** *ip-address* [*mask*] [**longer-prefixes**]| *protocol* [*process-id*] | [**list** *access-list-number* | *access-list-name*] | **static download**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 50000`	Enters router configuration mode to create or configure a BGP routing process.

	Command or Action	Purpose
Step 4	**address-family ipv4** **Example:** `Router(config-router-af)# address-family ipv4`	Enters address family configuration mode.
Step 5	**address-family ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] **Example:** `Router(config-router)# address-family ipv4`	The BGP Link Bandwidth feature is supported only under the IPv4 and VPNv4 address families.
Step 6	**bgp dmzlink-bw** **Example:** `Router(config-router-af)# bgp dmzlink-bw`	Configures BGP to distribute traffic proportionally to the bandwidth of the link. • This command must be entered on each router that contains an external interface that is to be used for multipath load balancing.
Step 7	**neighbor** *ip-address* **dmzlink-bw** **Example:** `Router(config-router-af)# neighbor 172.16.1.1 dmzlink-bw`	Configures BGP to include the link bandwidth attribute for routes learned from the external interface specified IP address. • This command must be configured for each eBGP link that is to be configured as a multipath. Enabling this command allows the bandwidth of the external link to be propagated through the link bandwidth extended community.
Step 8	**neighbor** *ip-address* **send-community** [**both** | **extended** | **standard**] **Example:** `Router(config-router-af)# neighbor 10.10.10.1 send-community extended`	(Optional) Enables community or extended community exchange with the specified neighbor. • This command must be configured for iBGP peers to which the link bandwidth extended community attribute is to be propagated.
Step 9	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode, and enters privileged EXEC mode.

	Command or Action	Purpose
Step 10	**show ip bgp** *ip-address* [**longer-prefixes** [**injected**] | **shorter-prefixes** [*mask-length*]] **Example:** `Router# show ip bgp 10.0.0.0`	(Optional) Displays information about the TCP and BGP connections to neighbors. • The output displays the status of the link bandwidth configuration. The bandwidth of the link is shown in kilobytes.
Step 11	**show ip route** *ip-address* [*mask*] [**longer-prefixes**]| *protocol* [*process-id*] | [**list** *access-list-number* | *access-list-name*] | **static download**] **Example:** `Router# show ip route 10.0.0.0`	(Optional) Displays the current state of the routing table. • The output displays traffic share values, including the weights of the links that are used to direct traffic proportionally to the bandwidth of each link.

Configuration Examples for BGP Link Bandwidth

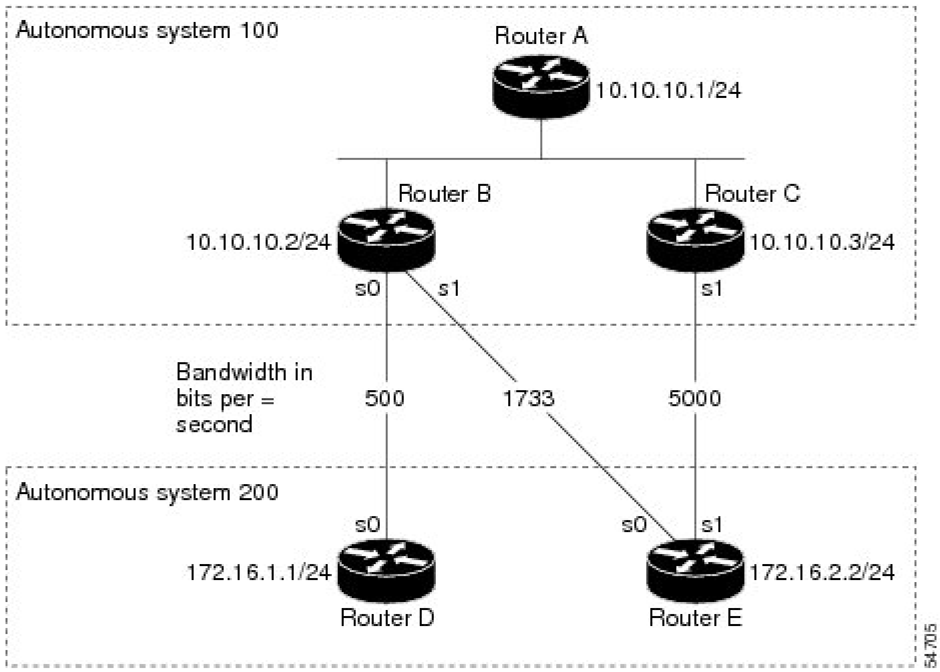
Example BGP Link Bandwidth Configuration

In the following examples, the BGP Link Bandwidth feature is configured so BGP will distribute traffic proportionally to the bandwidth of each external link. The figure below shows two external autonomous systems connected by three links that each carry a different amount of bandwidth (unequal cost links). Multipath load balancing is enabled and traffic is balanced proportionally.

Note The BGP Link Bandwidth feature functions for simple topologies that have a single path toward the exit points.

Caution The BGP Link Bandwidth feature might not function properly if load balancing is required toward the exit points.

Figure 31 *BGP Link Bandwidth Configuration*



Router A Configuration

In the following example, Router A is configured to support iBGP multipath load balancing and to exchange the BGP extended community attribute with iBGP neighbors:

```
RouterA(config)# router bgp 100

RouterA(config-router)# neighbor 10.10.10.2 remote-as 100

RouterA(config-router)# neighbor 10.10.10.2 update-source Loopback 0

RouterA(config-router)# neighbor 10.10.10.3 remote-as 100

RouterA(config-router)# neighbor 10.10.10.3 update-source Loopback 0

RouterA(config-router)# address-family ipv4

RouterA(config-router-af)# bgp dmzlink-bw
RouterA(config-router-af)# neighbor 10.10.10.2 activate

RouterA(config-router-af)# neighbor 10.10.10.2 send-community both

RouterA(config-router-af)# neighbor 10.10.10.3 activate

RouterA(config-router-af)# neighbor 10.10.10.3 send-community both

RouterA(config-router-af)# maximum-paths ibgp 6
```

Router B Configuration

In the following example Router B is configured to support multipath load balancing, to distribute Router D and Router E link traffic proportionally to the bandwidth of each link, and to advertise the bandwidth of these links to iBGP neighbors as an extended community:

```
RouterB(config)# router bgp 100

RouterB(config-router)# neighbor 10.10.10.1 remote-as 100

RouterB(config-router)# neighbor 10.10.10.1 update-source Loopback 0

RouterB(config-router)# neighbor 10.10.10.3 remote-as 100

RouterB(config-router)# neighbor 10.10.10.3 update-source Loopback 0
RouterB(config-router)# neighbor 172.16.1.1 remote-as 200

RouterB(config-router)# neighbor 172.16.1.1 ebgp-multihop 1
RouterB(config-router)# neighbor 172.16.2.2 remote-as 200

RouterB(config-router)# neighbor 172.16.2.2 ebgp-multihop 1
RouterB(config-router)# address-family ipv4

RouterB(config-router-af)# bgp dmzlink-bw

RouterB(config-router-af)# neighbor 10.10.10.1 activate

RouterB(config-router-af)# neighbor 10.10.10.1 next-hop-self

RouterB(config-router-af)# neighbor 10.10.10.1 send-community both

RouterB(config-router-af)# neighbor 10.10.10.3 activate

RouterB(config-router-af)# neighbor 10.10.10.3 next-hop-self

RouterB(config-router-af)# neighbor 10.10.10.3 send-community both

RouterB(config-router-af)# neighbor 172.16.1.1
 activate
RouterB(config-router-af)# neighbor 172.16.1.1 dmzlink-bw

RouterB(config-router-af)# neighbor 172.16.2.2 activate
RouterB(config-router-af)# neighbor 172.16.2.2 dmzlink-bw
RouterB(config-router-af)# maximum-paths ibgp 6
RouterB(config-router-af)# maximum-paths 6
```

Router C Configuration

In the following example Router C is configured to support multipath load balancing and to advertise the bandwidth of the link with Router E to iBGP neighbors as an extended community:

```
RouterC(config)# router bgp 100
RouterC(config-router)# neighbor 10.10.10.1 remote-as 100
RouterC(config-router)# neighbor 10.10.10.1 update-source Loopback 0
RouterC(config-router)# neighbor 10.10.10.2 remote-as 100
RouterC(config-router)# neighbor 10.10.10.2 update-source Loopback 0
RouterC(config-router)# neighbor 172.16.3.30 remote-as 200
RouterC(config-router)# neighbor 172.16.3.30 ebgp-multihop 1
RouterC(config-router)# address-family ipv4
RouterC(config-router-af)# bgp dmzlink-bw

RouterC(config-router-af)# neighbor 10.10.10.1 activate
RouterC(config-router-af)# neighbor 10.10.10.1 send-community both
RouterC(config-router-af)# neighbor 10.10.10.1 next-hop-self
RouterC(config-router-af)# neighbor 10.10.10.2 activate
RouterC(config-router-af)# neighbor 10.10.10.2 send-community both
RouterC(config-router-af)# neighbor 10.10.10.2 next-hop-self
RouterC(config-router-af)# neighbor 172.16.3.3 activate
RouterC(config-router-af)# neighbor 172.16.3.3 dmzlink-bw
```

```
RouterC(config-router-af)# maximum-paths ibgp 6
RouterC(config-router-af)# maximum-paths 6
```

Verifying BGP Link Bandwidth

The examples in this section show the verification of this feature on Router A, Router B, and Router C.

Router B

In the following example, the **show ip bgp** command is entered on Router B to verify that two unequal cost best paths have been installed into the BGP routing table. The bandwidth for each link is displayed with each route.

```
RouterB# show ip bgp 192.168.1.0
BGP routing table entry for 192.168.1.0/24, version 48
Paths: (2 available, best #2)
Multipath: eBGP
  Advertised to update-groups:
     1          2
  200
    172.16.1.1 from 172.16.1.2 (192.168.1.1)
      Origin incomplete, metric 0, localpref 100, valid, external, multipath, best
      Extended Community: 0x0:0:0
      DMZ-Link Bw 278 kbytes
  200
    172.16.2.2 from 172.16.2.2 (192.168.1.1)
      Origin incomplete, metric 0, localpref 100, valid, external, multipath, best
      Extended Community: 0x0:0:0
      DMZ-Link Bw 625 kbytes
```

Router A

In the following example, the **show ip bgp** command is entered on Router A to verify that the link bandwidth extended community has been propagated through the iBGP network to Router A. Exit links are located on Router B and Router C. The output shows that a route for each exit link to autonomous system 200 has been installed as a best path in the BGP routing table.

```
RouterA# show ip bgp 192.168.1.0

BGP routing table entry for 192.168.1.0/24, version 48
Paths: (3 available, best #3)
Multipath: eBGP
  Advertised to update-groups:
     1          2
  200
    172.16.1.1 from 172.16.1.2 (192.168.1.1)
      Origin incomplete, metric 0, localpref 100, valid, external, multipath
      Extended Community: 0x0:0:0
      DMZ-Link Bw 278 kbytes
  200
    172.16.2.2 from 172.16.2.2 (192.168.1.1)
      Origin incomplete, metric 0, localpref 100, valid, external, multipath, best
      Extended Community: 0x0:0:0
      DMZ-Link Bw 625 kbytes
  200
    172.16.3.3 from 172.16.3.3 (192.168.1.1)
      Origin incomplete, metric 0, localpref 100, valid, external, multipath, best
      Extended Community: 0x0:0:0
      DMZ-Link Bw 2500 kbytes
```

Router A

In the following example, the **show ip route** command is entered on Router A to verify the multipath routes that are advertised and the associated traffic share values:

```
RouterA# show ip route 192.168.1.0
Routing entry for 192.168.1.0/24
  Known via "bgp 100", distance 200, metric 0
  Tag 200, type internal
  Last update from 172.168.1.1 00:01:43 ago
  Routing Descriptor Blocks:
  * 172.168.1.1, from 172.168.1.1, 00:01:43 ago
      Route metric is 0, traffic share count is 13
      AS Hops 1, BGP network version 0
      Route tag 200
    172.168.2.2, from 172.168.2.2, 00:01:43 ago
      Route metric is 0, traffic share count is 30
      AS Hops 1, BGP network version 0
      Route tag 200
    172.168.3.3, from 172.168.3.3, 00:01:43 ago
      Route metric is 0, traffic share count is 120
      AS Hops 1, BGP network version 0
      Route tag 200
```

Where to Go Next

For information about the BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN feature, refer to the following document: "BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN".

For more information about the iBGP Multipath Load Sharing feature, refer to the following document: "iBGP Multipath Load Sharing".

Additional References

The following sections provide references related to the BGP Link Bandwidth feature.

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
CEF configuration tasks	"Cisco Express Forwarding Overview" module

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL: http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFC	Title
draft-ramachandra-bgp-ext-communities-09.txt	*BGP Extended Communities Attribute*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for BGP Link Bandwidth

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 20 Feature Information for BGP Link Bandwidth

Feature Name	Releases	Feature Information
BGP Link Bandwidth	12.2(2)T 12.2(14)S	This feature advertises the bandwidth of an autonomous system exit link as an extended community. The link bandwidth extended community attribute is propagated to iBGP peers when extended community exchange is enabled. The following commands were introduced or modified: **router bgp, address-family ipv4, address-family ipv4**, **bgp dmzlink-bw, neighbor**, **show ip bgp**, **show ip route**.

iBGP Multipath Load Sharing

This feature module describes the iBGP Multipath Load Sharing feature.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for iBGP Multipath Load Sharing

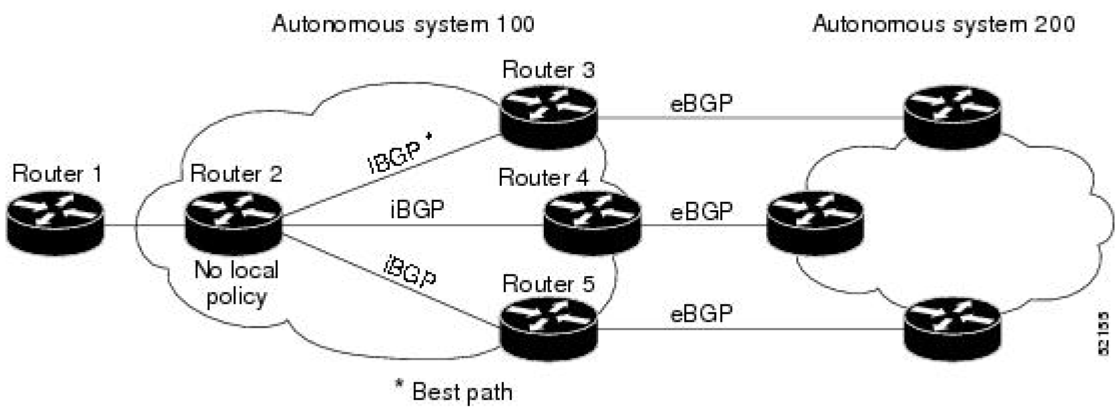
- Route Reflector Limitation--With multiple iBGP paths installed in a routing table, a route reflector will advertise only one of the paths (one next hop).
- Memory Consumption Restriction--Each IP routing table entry for a BGP prefix that has multiple iBGP paths uses approximately 350 bytes of additional memory. We recommend not using this feature on a router with a low amount of available memory and especially when the router is carrying a full Internet routing table.
- The iBGP Multipath Load Sharing feature is supported for the following platforms in Cisco IOS Release 12.2(14)S:
    - Cisco 7200 series
    - Cisco 7400 series
    - Cisco 7500 series

Information about iBGP Multipath Load Sharing

iBGP Multipath Load Sharing Overview

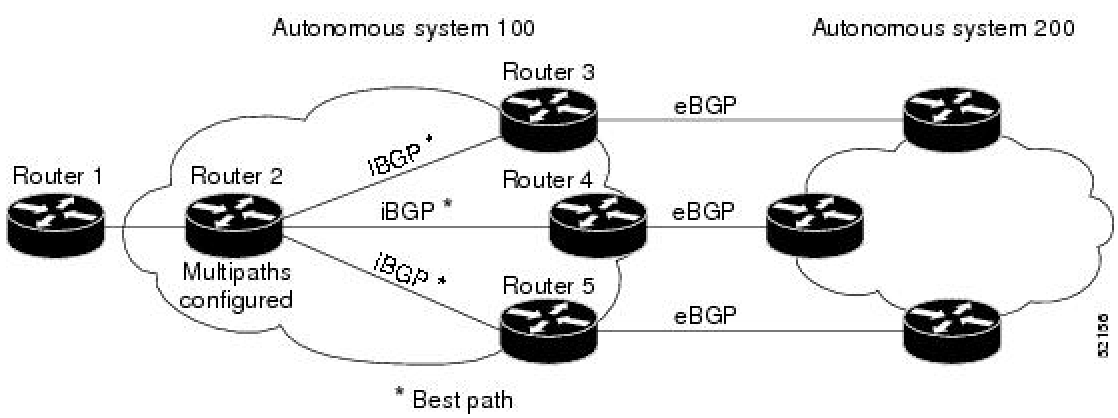
When a Border Gateway Protocol (BGP) speaking router with no local policy configured receives multiple network layer reachability information (NLRI) from the internal BGP (iBGP) for the same destination, the router will choose one iBGP path as the best path. The best path is then installed in the IP routing table of the router. For example, in the figure below, although there are three paths to autonomous system 200, Router 2 determines that one of the paths to autonomous system 200 is the best path and uses this path only to reach autonomous system 200.

Figure 32 Non-MPLS Topology with One Best Path



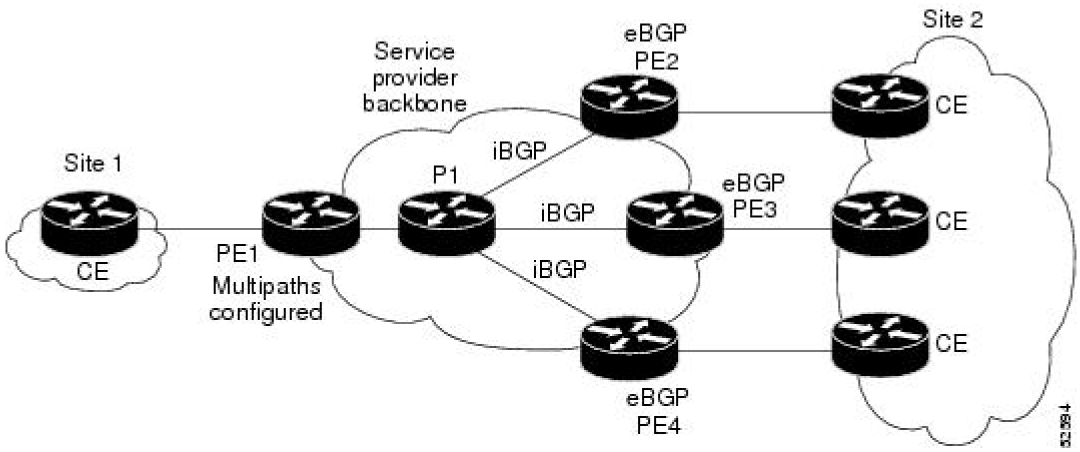
The iBGP Multipath Load Sharing feature enables the BGP speaking router to select multiple iBGP paths as the best paths to a destination. The best paths or multipaths are then installed in the IP routing table of the router. For example, on router 2 in the figure below, the paths to routers 3, 4, and 5 are configured as multipaths and can be used to reach autonomous system 200, thereby equally sharing the load to autonomous system 200.

Figure 33 Non-MPLS Topology with Three Multipaths



The iBGP Multipath Load Sharing feature functions similarly in a Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) with a service provider backbone. For example, on router PE1 in the figure below, the paths to routers PE2, PE3, and PE4 can be selected as multipaths and can be used to equally share the load to site 2.

Figure 34 MPLS VPN with Three Multipaths



For multiple paths to the same destination to be considered as multipaths, the following criteria must be met:

- All attributes must be the same. The attributes include weight, local preference, autonomous system path (entire attribute and not just length), origin code, Multi Exit Discriminator (MED), and Interior Gateway Protocol (IGP) distance.
- The next hop router for each multipath must be different.

Even if the criteria are met and multiple paths are considered multipaths, the BGP speaking router will still designate one of the multipaths as the best path and advertise this best path to its neighbors.

The iBGP Multipath Load Sharing feature is similar to BGP multipath support for external BGP (eBGP) paths; however, the iBGP Multipath Load Sharing feature is applied to internal rather than eBGP paths.

Benefits of iBGP Multipath Load Sharing

Configuring multiple iBGP best paths enables a router to evenly share the traffic destined for a particular site.

How To Configure iBGP Multipath Load Sharing

Configuring iBGP Multipath Load Sharing

To configure the iBGP Multipath Load Sharing feature, use the following command in router configuration mode:

Command	Purpose
`Router(config-router)#` **maximum-paths ibgp** *maximum-number*	Controls the maximum number of parallel iBGP routes that can be installed in a routing table.

Verifying iBGP Multipath Load Sharing

To verify that the iBGP Multipath Load Sharing feature is configured correctly, perform the following steps:

SUMMARY STEPS

1. Enter the **show ip bgp** *network-number* EXEC command to display attributes for a network in a non-MPLS topology, or the **show ip bgp vpnv4 all** *ip-prefix*EXEC commandtodisplay attributes for a network in an MPLS VPN:
2. In the display resulting from the **show ip bgp** *network-number* EXEC command or the **show ip bgp vpnv4 all** *ip-prefix*EXEC command, verify that the intended multipaths are marked as "multipaths." Notice that one of the multipaths is marked as "best."
3. Enter the **show ip route** *ip-address* EXEC command to display routing information for a network in a non-MPLS topology or the **show ip route vrf** *vrf-name ip-prefix* EXEC command to display routing information for a network in an MPLS VPN:
4. Verify that the paths marked as "multipath" in the display resulting from the **show ip bgp** *ip-prefix*EXEC command or the **show ip bgp vpnv4 all** *ip-prefix* EXEC command are included in the routing information. (The routing information is displayed after performing Step 3.)

DETAILED STEPS

Step 1 Enter the **show ip bgp** *network-number* EXEC command to display attributes for a network in a non-MPLS topology, or the **show ip bgp vpnv4 all** *ip-prefix*EXEC commandtodisplay attributes for a network in an MPLS VPN:

Example:

```
Router# show ip bgp 10.22.22.0
BGP routing table entry for 10.22.22.0/24, version 119
Paths:(6 available, best #1)
Multipath:iBGP
Flag:0x820
  Advertised to non peer-group peers:
  10.1.12.12
  22
    10.2.3.8 (metric 11) from 10.1.3.4 (100.0.0.5)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath, best
      Originator:100.0.0.5, Cluster list:100.0.0.4
  22
    10.2.1.9 (metric 11) from 10.1.1.2 (100.0.0.9)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Originator:100.0.0.9, Cluster list:100.0.0.2
  22
    10.2.5.10 (metric 11) from 10.1.5.6 (100.0.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Originator:100.0.0.10, Cluster list:100.0.0.6
  22
    10.2.4.10 (metric 11) from 10.1.4.5 (100.0.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Originator:100.0.0.10, Cluster list:100.0.0.5
```

```
  22
    10.2.6.10 (metric 11) from 10.1.6.7 (100.0.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Originator:100.0.0.10, Cluster list:100.0.0.7
Router# show ip bgp vpnv4 all 10.22.22.0
BGP routing table entry for 100:1:10.22.22.0/24, version 50
Paths:(6 available, best #1)
Multipath:iBGP
  Advertised to non peer-group peers:
  200.1.12.12
  22
    10.22.7.8 (metric 11) from 10.11.3.4 (100.0.0.8)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath, best
      Extended Community:RT:100:1
      Originator:100.0.0.8, Cluster list:100.1.1.44
  22
    10.22.1.9 (metric 11) from 10.11.1.2 (100.0.0.9)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:RT:100:1
      Originator:100.0.0.9, Cluster list:100.1.1.22
  22
    10.22.6.10 (metric 11) from 10.11.6.7 (100.0.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:RT:100:1
      Originator:100.0.0.10, Cluster list:100.0.0.7
  22
    10.22.4.10 (metric 11) from 10.11.4.5 (100.0.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:RT:100:1
      Originator:100.0.0.10, Cluster list:100.0.0.5
  22
    10.22.5.10 (metric 11) from 10.11.5.6 (100.0.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:RT:100:1
      Originator:100.0.0.10, Cluster list:100.0.0.6
```

Step 2 In the display resulting from the **show ip bgp** *network-number* EXEC command or the **show ip bgp vpnv4 all** *ip-prefix*EXEC command, verify that the intended multipaths are marked as "multipaths." Notice that one of the multipaths is marked as "best."

Step 3 Enter the **show ip route** *ip-address* EXEC command to display routing information for a network in a non-MPLS topology or the **show ip route vrf** *vrf-name ip-prefix* EXEC command to display routing information for a network in an MPLS VPN:

Example:

```
Router# show ip route 10.22.22.0
Routing entry for 10.22.22.0/24
  Known via "bgp 1", distance 200, metric 0
  Tag 22, type internal
  Last update from 10.2.6.10 00:00:03 ago
  Routing Descriptor Blocks:
  * 10.2.3.8, from 10.1.3.4, 00:00:03 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.2.1.9, from 10.1.1.2, 00:00:03 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.2.5.10, from 10.1.5.6, 00:00:03 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.2.4.10, from 10.1.4.5, 00:00:03 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.2.6.10, from 10.1.6.7, 00:00:03 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
Router# show ip route vrf PATH 10.22.22.0
Routing entry for 10.22.22.0/24
```

```
Known via "bgp 1", distance 200, metric 0
Tag 22, type internal
Last update from 10.22.5.10 00:01:07 ago
Routing Descriptor Blocks:
* 10.22.7.8 (Default-IP-Routing-Table), from 10.11.3.4, 00:01:07 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
  10.22.1.9 (Default-IP-Routing-Table), from 10.11.1.2, 00:01:07 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
  10.22.6.10 (Default-IP-Routing-Table), from 10.11.6.7, 00:01:07 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
  10.22.4.10 (Default-IP-Routing-Table), from 10.11.4.5, 00:01:07 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
  10.22.5.10 (Default-IP-Routing-Table), from 10.11.5.6, 00:01:07 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
```

Step 4 Verify that the paths marked as "multipath" in the display resulting from the **show ip bgp** *ip-prefix*EXEC command or the **show ip bgp vpnv4 all** *ip-prefix* EXEC command are included in the routing information. (The routing information is displayed after performing Step 3.)

Monitoring and Maintaining iBGP Multipath Load Sharing

To display iBGP Multipath Load Sharing information, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router#` **show ip bgp** *ip-prefix*	Displays attributes and multipaths for a network in a non-MPLS topology.
`Router#` **show ip bgp vpnv4 all** *ip-prefix*	Displays attributes and multipaths for a network in an MPLS VPN.
`Router#` **show ip route** *ip-prefix*	Displays routing information for a network in a non-MPLS topology.
`Router#` **show ip route vrf** *vrf-name ip-prefix*	Displays routing information for a network in an MPLS VPN.

Configuration Examples for iBGP Multipath Load Sharing

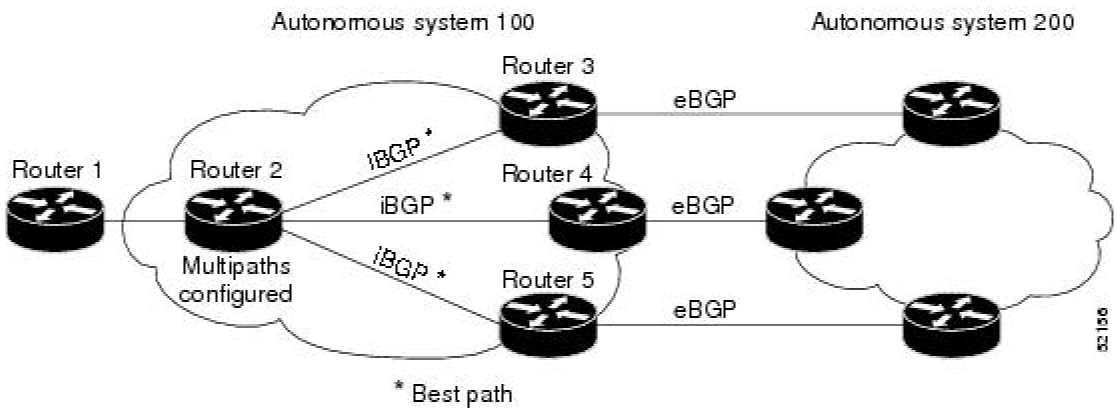
The examples assume that the appropriate attributes for each path are equal and that the next hop router for each multipath is different.

- Example iBGP Multipath Load Sharing in a Non-MPLS Topology, page 369
- Example iBGP Multipath Load Sharing in an MPLS VPN Topology, page 369

Example iBGP Multipath Load Sharing in a Non-MPLS Topology

The following example shows how to set up the iBGP Multipath Load Sharing feature in a non-MPLS topology (see the figure below).

Figure 35 Non-MPLS Topology Example



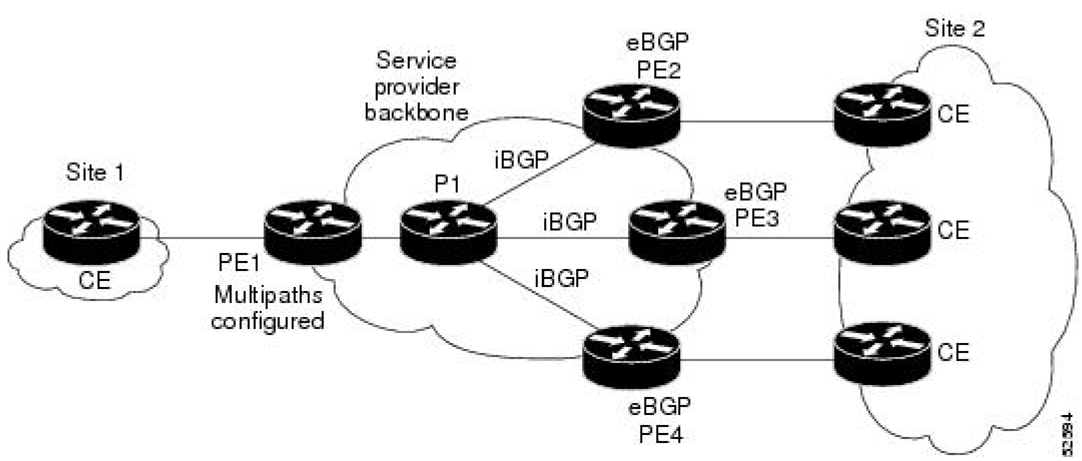
Router 2 Configuration

```
router bgp 100
maximum-paths ibgp 3
```

Example iBGP Multipath Load Sharing in an MPLS VPN Topology

The following example shows how to set up the iBGP Multipath Load Sharing feature in an MPLS VPN topology (see the figure below).

Figure 36 MPLS VPN Topology Example



Router PE1 Configuration

```
router bgp 100
```

```
address-family ipv4 unicast vrf site2
 maximum-paths ibgp 3
```

Additional References

The following sections provide references related to the iBGP Multipath Load Sharing feature.

Related Documents

Related Topic	Document Title
BGP multipath load sharing for both eBGP and iBGP in an MPLS-VPN	"BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN"
Advertising the bandwidth of an autonomous system exit link as an extended community	" BGP Link Bandwidth"
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*
Cisco IOS master command list, all releases	Cisco IOS Master Command List, All Releases

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS IP Routing: BGP Command Reference*. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

New Commands

- **maximum-paths ibgp**

Modified Commands

- **show ip bgp**
- **show ip bgp vpnv4**
- **show ip route**
- **show ip route vrf**

Feature Information for iBGP Multipath Load Sharing

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 21 Feature Information for iBGP Multipath Load Sharing

Feature Name	Releases	Feature Information
iBGP Multipath Load Sharing	12.2(14)S 12.2(2)T	The iBGP Multipath Load Sharing feature enables the BGP speaking router to select multiple iBGP paths as the best paths to a destination. The following commands were introduced or modified: **maximum-paths ibgp**, **show ip bgp**, **show ip bgp vpnv4**, **show ip route**, **show ip route vrf**.

BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

The BGP Multipath Load Sharing for eBGP and iBGP feature allows you to configure multipath load balancing with both external BGP (eBGP) and internal BGP (iBGP) paths in Border Gateway Protocol (BGP) networks that are configured to use Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs). This feature provides improved load balancing deployment and service offering capabilities and is useful for multi-homed autonomous systems and Provider Edge (PE) routers that import both eBGP and iBGP paths from multihomed and stub networks.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Load Balancing is Configured Under CEF

Cisco Express Forwarding (CEF) or distributed CEF (dCEF) must be enabled on all participating routers.

Restrictions for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Address Family Support

This feature is configured on a per VPN routing and forwarding instance (VRF) basis. This feature can be configured under only the IPv4 VRF address family.

Memory Consumption Restriction

Each BGP multipath routing table entry will use additional memory. We recommend that you do not use this feature on a router with a low amount of available memory and especially if router is carries full Internet routing tables.

Route Reflector Limitation

When multiple iBGP paths installed in a routing table, a route reflector will advertise only one paths (next hop). If a router is behind a route reflector, all routers that are connected to multihomed sites will not be advertised unless a different route distinguisher is configured for each VRF.

Information About BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Multipath Load Sharing Between eBGP and iBGP

A BGP routing process will install a single path as the best path in the routing information base (RIB) by default. The **maximum-paths** command allows you to configure BGP to install multiple paths in the RIB for multipath load sharing. BGP uses the best path algorithm to still select a single multipath as the best path and advertise the best path to BGP peers.

Note The number of paths of multipaths that can be configured is documented on the **maximum-paths** command reference page.

Load balancing over the multipaths is performed by CEF. CEF load balancing is configured on a per-packet round robin or on a per session (source and destination pair) basis. For information about CEF, refer to the "Cisco Express Forwarding Overview" documentation:

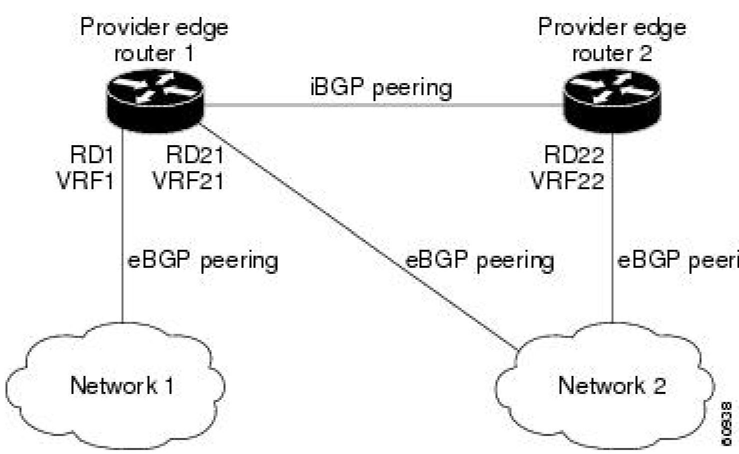
The BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature is enabled only under the IPv4 VRF address family configuration mode. When enabled, this feature can perform load balancing on eBGP and/or iBGP paths that are imported into the VRF. The number of multipaths is configured on a per VRF basis. Separate VRF multipath configurations are isolated by unique route distinguisher.

Note The BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature operates within the parameters of configured outbound routing policy.

eBGP and iBGP Multipath Load Sharing in a BGP MPLS Network

The figure below shows a service provider BGP MPLS network that connects two remote networks to PE router 1 and PE router 2. PE router 1 and PE router 2 are both configured for VPNv4 unicast iBGP peering. Network 2 is a multihomed network that is connected to PE router 1 and PE router 2. Network 2 also has extranet VPN services configured with Network 1. Both Network 1 and Network 2 are configured for eBGP peering with the PE routers.

Figure 37 *A Service Provider BGP MPLS Network*

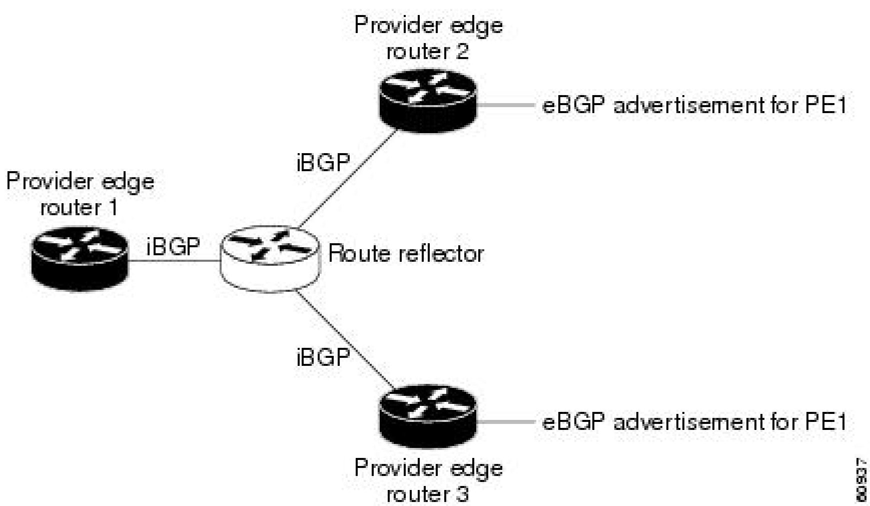


PE router 1 can be configured with the BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature so that both iBGP and eBGP paths can be selected as multipaths and imported into the VRF of Network 1. The multipaths will be used by CEF to perform load balancing. IP traffic that is sent from Network 2 to PE router 1 and PE router 2 will be sent across the eBGP paths as IP traffic. IP traffic that is sent across the iBGP path will be sent as MPLS traffic, and MPLS traffic that is sent across an eBGP path will be sent as IP traffic. Any prefix that is advertised from Network 2 will be received by PE router 1 through route distinguisher (RD) 21 and RD 22.The advertisement through RD 21 will be carried in IP packets, and the advertisement through RD 22 will be carried in MPLS packets. Both paths can be selected as multipaths for VRF1 and installed into the VRF1 RIB.

eBGP and iBGP Multipath Load Sharing With Route Reflectors

The figure below shows a topology that contains three PE routers and a route reflector, all configured for iBGP peering. PE router 2 and PE router 3 each advertise an equal preference eBGP path to PE router 1. By default, the route reflector will choose only one path and advertise PE router 1.

Figure 38 A Topology with a Route Reflector



For all equal preference paths to PE router 1 to be advertised through the route reflector, you must configure each VRF with a different RD. The prefixes received by the route reflector will be recognized differently and advertised to PE router 1.

Benefits of Multipath Load Sharing for Both eBGP and iBGP

The BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature allows multihomed autonomous systems and PE routers to be configured to distribute traffic across both eBGP and iBGP paths.

How to Configure BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Configuring Multipath Load Sharing for Both eBGP an iBGP

To configure this feature, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **address-family ipv4 vrf** *vrf-name*
5. **maximum-paths eibgp** *number* [**import** *number*]
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 40000`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**address-family ipv4 vrf** *vrf-name* **Example:** `Router(config-router)# address-family ipv4 vrf RED`	Places the router in address family configuration mode. • Separate VRF multipath configurations are isolated by unique route distinguisher.
Step 5	**maximum-paths eibgp** *number* [**import** *number*] **Example:** `Router(config-router-af)# maximum-paths eibgp 6`	Configures the number of parallel iBGP and eBGP routes that can be installed into a routing table. **Note** The maximum-paths eibgp command can be configured only under the IPv4 VRF address family configuration mode and cannot be configured in any other address family configuration mode.

	Command or Action	Purpose
Step 6	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode, and enters Privileged EXEC mode.

Verifying Multipath Load Sharing for Both eBGP an iBGP

To verify this feature, perform the steps in this section

SUMMARY STEPS

1. **enable**
2. **show ip bgp neighbors** [*neighbor-address* [**advertised-routes** | **dampened-routes** | **flap-statistics**| **paths**[*regexp*] | **received** *prefix-filter* | **received-routes** | **routes**]]
3. **show ip bgp vpnv4** {**all** | **rd** *route-distinguisher*| **vrf** *vrf-name*}
4. **show ip route vrf** *vrf-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip bgp neighbors** [*neighbor-address* [**advertised-routes** | **dampened-routes** | **flap-statistics**| **paths**[*regexp*] | **received** *prefix-filter* | **received-routes** | **routes**]] **Example:** `Router# show ip bgp neighbors`	Displays information about the TCP and BGP connections to neighbors.
Step 3	**show ip bgp vpnv4** {**all** | **rd** *route-distinguisher*| **vrf** *vrf-name*} **Example:** `Router# show ip bgp vpnv4 vrf RED`	Displays VPN address information from the BGP table. This command is used to verify that the VRF has been received by BGP.

	Command or Action	Purpose
Step 4	**show ip route vrf** *vrf-name* **Example:** `Router# show ip route vrf RED`	Displays the IP routing table associated with a VRF instance. The show ip route vrf command is used to verify that the VRF is in the routing table.

Configuration Examples for the BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN Feature

eBGP and iBGP Multipath Load Sharing Configuration Example

This following configuration example configures a router in address-family mode to select six BGP routes (eBGP or iBGP) as multipaths:

```
Router(config)# router bgp 40000
 Router(config-router)# address-family ipv4 vrf RED
 Router(config-router-af)# maximum-paths eibgp 6
 Router(config-router-af)# end
```

eBGP and iBGP Multipath Load Sharing Verification Examples

To verify that iBGP and eBGP routes have been configured for load sharing, use the **show ip bgp vpnv4**EXEC command or the **show ip route vrf** EXEC command.

In the following example, the **show ip bgp vpnv4** command is entered to display multipaths installed in the VPNv4 RIB:

```
Router# show ip bgp vpnv4 all 10.22.22.0
BGP routing table entry for 10:1:22.22.22.0/24, version 19
Paths:(5 available, best #5)
Multipath:eiBGP
  Advertised to non peer-group peers:
  10.0.0.2 10.0.0.3 10.0.0.4 10.0.0.5
  22
    10.0.0.2 (metric 20) from 10.0.0.4 (10.0.0.4)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:0x0:0:0 RT:100:1 0x0:0:0
      Originator:10.0.0.2, Cluster list:10.0.0.4
  22
    10.0.0.2 (metric 20) from 10.0.0.5 (10.0.0.5)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:0x0:0:0 RT:100:1 0x0:0:0
      Originator:10.0.0.2, Cluster list:10.0.0.5
  22
    10.0.0.2 (metric 20) from 10.0.0.2 (10.0.0.2)
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:RT:100:1 0x0:0:0
  22
    10.0.0.2 (metric 20) from 10.0.0.3 (10.0.0.3)
```

```
      Origin IGP, metric 0, localpref 100, valid, internal, multipath
      Extended Community:0x0:0:0 RT:100:1 0x0:0:0
      Originator:10.0.0.2, Cluster list:10.0.0.3
  22
    10.1.1.12 from 10.1.1.12 (10.22.22.12)
      Origin IGP, metric 0, localpref 100, valid, external, multipath, best
      Extended Community:RT:100:1
```

In the following example, the **show ip route vrf** command is entered to display multipath routes in the VRF table:

```
Router# show ip route vrf PATH 10.22.22.0
Routing entry for 10.22.22.0/24
  Known via "bgp 1", distance 20, metric 0
  Tag 22, type external
  Last update from 10.1.1.12 01:59:31 ago
  Routing Descriptor Blocks:
  * 10.0.0.2 (Default-IP-Routing-Table), from 10.0.0.4, 01:59:31 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.0.0.2 (Default-IP-Routing-Table), from 10.0.0.5, 01:59:31 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.0.0.2 (Default-IP-Routing-Table), from 10.0.0.2, 01:59:31 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.0.0.2 (Default-IP-Routing-Table), from 10.0.0.3, 01:59:31 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
    10.1.1.12, from 10.1.1.12, 01:59:31 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
```

Where to Go Next

For information about advertising the bandwidth of an autonomous system exit link as an extended community, refer to the "BGP Link Bandwidth" document.

Additional References

For additional information related to BGP Multipath Load sharing for Both eBGP and iBGP in an MPLS VPN, refer to the following references:

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
Comprehensive BGP link bandwidth configuration examples and tasks	"BGP Link Bandwidth" module
CEF configuration tasks	"Cisco Express Forwarding Overview" module

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL: http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1771	A Border Gateway Protocol 4 (BGP4)
RFC 2547	BGP/MPLS VPNs
RFC 2858	Multiprotocol Extensions for BGP-4

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 22 Feature Information for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Feature Name	Releases	Feature Configuration Information
BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN	12.0(24)S 12.2(14)S 12.2(18)SXE 12.2(4)T 15.0(1)S Cisco IOS XE 3.1.0SG	The BGP Multipath Load Sharing for eBGP and iBGP feature allows you to configure multipath load balancing with both eBGP and iBGP paths in BGP networks that are configured to use MPLS VPNs. This feature provides improved load balancing deployment and service offering capabilities and is useful for multi-homed autonomous systems and PE routers that import both eBGP and iBGP paths from multihomed and stub networks. The following command was introduced or modified by this feature: **maximum-paths eibgp.**

Loadsharing IP Packets Over More Than Six Parallel Paths

The Loadsharing IP Packets Over More Than Six Parallel Paths feature increases the maximum number of parallel routes that can be installed to the routing table for multipath loadsharing.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Overview of Loadsharing IP Packets over More Than Six Parallel Paths

The Loadsharing IP Packets over More Than Six Parallel Paths feature increases the maximum number of parallel routes that can be installed to the routing table. The maximum number has been increased from six to sixteen for the following commands:

- **maximum-paths**
- **maximum-paths eibgp**
- **maximum-paths ibgp**

The output of the **show ip route summary** command has been updated to display the number of parallel routes supported by the routing table.

The benefits of this feature include the following:

- More flexible configuration of parallel routes in the routing table.
- Ability to configure multipath loadsharing over more links to allow for the configuration of higher-bandwidth aggregation using lower-speed links.

Additional References

For additional information related to multipath loadsharing and the configuration of parallel routes, see the following references:

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
eiBGP Multipath Load Sharing	"BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN" module
iBGP Multipath Load Sharing	"iBGP Multipath Load Sharing" module
Cisco IOS master command list, all releases	Cisco IOS Master Command List, All Releases

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Loadsharing IP Packets Over More Than Six Parallel Paths

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 23 Feature Information for Loadsharing IP Packets Over More Than Six Parallel Paths

Feature Name	Releases	Feature Information
Loadsharing IP Packets Over More Than Six Parallel Paths	12.3(2)T, 12.2(25)S, Cisco IOS XE 3.1.0SG	The Loadsharing IP Packets Over More Than Six Parallel Paths feature increases the maximum number of parallel routes that can be installed to the routing table for multipath loadsharing. The following commands were modified: • **maximum-paths** • **maximum-paths eibgp** • **maximum-paths ibgp** • **show ip route summary**

BGP Policy Accounting

Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is sent to, or received from, different peers. Policy accounting is enabled on an input interface, and counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites

Before using the BGP Policy Accounting feature, you must enable BGP and CEF or dCEF on the router.

Information About BGP Policy Accounting

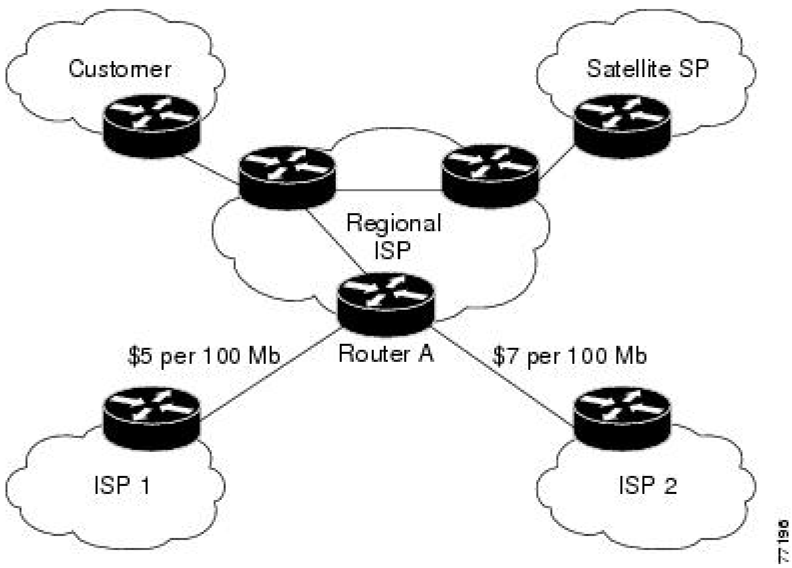
BGP Policy Accounting Overview

Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is sent to, or received from, different peers. Policy accounting is enabled on an input interface, and counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Using the BGP **table-map** command, prefixes added to the routing table are classified by BGP attribute, autonomous system number, or autonomous system path. Packet and byte counters are incremented per input interface. A Cisco IOS policy-based classifier maps the traffic into one of eight possible buckets, representing different traffic classes.

Using BGP policy accounting, you can account for traffic according to the route it traverses. Service providers (SPs) can identify and account for all traffic by customer and bill accordingly. In the figure below, BGP policy accounting can be implemented in Router A to measure packet and byte volumes in autonomous system buckets. Customers are billed appropriately for traffic that is routed from a domestic, international, or satellite source.

Figure 39 Sample Topology for BGP Policy Accounting



BGP policy accounting using autonomous system numbers can be used to improve the design of network circuit peering and transit agreements between Internet service providers (ISPs).

Benefits of BGP Policy Accounting

Account for IP Traffic Differentially

BGP policy accounting classifies IP traffic by autonomous system number, autonomous system path, or community list string, and increments packet and byte counters. Service providers can account for traffic and apply billing, according to the route specific traffic traverses.

Efficient Network Circuit Peering and Transit Agreement Design

Implementing BGP policy accounting on an edge router can highlight potential design improvements for peering and transit agreements.

How to Configure BGP Policy Accounting

Specifying the Match Criteria for BGP Policy Accounting

The first task in configuring BGP policy accounting is to specify the criteria that must be matched. Community lists, autonomous system paths, or autonomous system numbers are examples of BGP attributes that can be specified and subsequently matched using a route map.

To specify the BGP attribute to use for BGP policy accounting and create the match criteria in a route map, use the following commands in global configuration mode:

SUMMARY STEPS

1. Router(config)# **ip community-list** *community-list-number*{**permit**| **deny**} *community-number*
2. Router(config)# **route-map** *map-name*[**permit**| **deny**] [*sequence-number*]
3. Router(config-route-map)# **match community-list** *community-list-number*[**exact**]
4. Router(config-route-map)# **set traffic-index** *bucket-number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **ip community-list** *community-list-number*{**permit**| **deny**} *community-number*	Creates a community list for BGP and controls access to it. This step must be repeated for each community to be specified.
Step 2	Router(config)# **route-map** *map-name*[**permit**| **deny**] [*sequence-number*]	Enters route-map configuration mode and defines the conditions for policy routing. The *map-name* argument identifies a route map. The optional **permit** and **deny** keywords work with the match and set criteria to control how the packets are accounted for. The optional *sequence-number* argument indicates the position a new route map is to have in the list of route maps already configured with the same name.
Step 3	Router(config-route-map)# **match community-list** *community-list-number*[**exact**]	Matches a BGP community.
Step 4	Router(config-route-map)# **set traffic-index** *bucket-number*	Indicates where to output packets that pass a match clause of a route map for BGP policy accounting.

Classifying the IP Traffic and Enabling BGP Policy Accounting

After a route map has been defined to specify match criteria, you must configure a way to classify the IP traffic before enabling BGP policy accounting.

Using the **table-map** command, BGP classifies each prefix it adds to the routing table based on the match criteria. When the **bgp-policy accounting** command is configured on an interface, BGP policy accounting is enabled.

To classify the IP traffic and enable BGP policy accounting, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. Router(config)# **router bgp** *as-number*
2. Router(config-router)# **table-map** *route-map-name*
3. Router(config-router)# **network** *network-number*[**mask** *network-mask*]
4. Router(config-router)# **neighbor** *ip-address* **remote-as** *as-number*
5. Router(config-router)# **exit**
6. Router(config)# **interface** *interface-type interface-number*
7. Router(config-if)# **no ip directed-broadcast**
8. Router(config-if)# **ip address** *ip-address mask*
9. Router(config-if)# **bgp-policy accounting**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Router(config)# **router bgp** *as-number*	Configures a BGP routing process and enters router configuration mode for the specified routing process.
Step 2	Router(config-router)# **table-map** *route-map-name*	Classifies BGP prefixes entered in the routing table.
Step 3	Router(config-router)# **network** *network-number*[**mask** *network-mask*]	Specifies a network to be advertised by the BGP routing process.
Step 4	Router(config-router)# **neighbor** *ip-address* **remote-as** *as-number*	Specifies a BGP peer by adding an entry to the BGP routing table.
Step 5	Router(config-router)# **exit**	Exits to global configuration mode.
Step 6	Router(config)# **interface** *interface-type interface-number*	Specifies the interface type and number and enters interface configuration mode.
Step 7	Router(config-if)# **no ip directed-broadcast**	Configures the interface to drop directed broadcasts destined for the subnet to which that interface is attached, rather than being broadcast. This is a security issue.
Step 8	Router(config-if)# **ip address** *ip-address mask*	Configures the interface with an IP address.
Step 9	Router(config-if)# **bgp-policy accounting**	Enables BGP policy accounting for the interface.

Verifying BGP Policy Accounting

To verify that BGP policy accounting is operating, perform the following steps:

SUMMARY STEPS

1. Enter the **show ip cef** EXEC command with the **detail** keyword to learn which accounting bucket is assigned to a specified prefix.
2. Enter the **show ip bgp** EXEC command for the same prefix used in Step 1--192.168.5.0-- to learn which community is assigned to this prefix.
3. Enter the **show cef interface policy-statistics** EXEC command to display the per-interface traffic statistics.

DETAILED STEPS

Step 1 Enter the **show ip cef** EXEC command with the **detail** keyword to learn which accounting bucket is assigned to a specified prefix.

In this example, the output is displayed for the prefix 192.168.5.0. It shows that the accounting bucket number 4 (traffic_index 4) is assigned to this prefix.

Example:

```
Router# show ip cef 192.168.5.0 detail
192.168.5.0/24, version 21, cached adjacency to POS7/2
0 packets, 0 bytes, traffic_index 4
  via 10.14.1.1, 0 dependencies, recursive
   next hop 10.14.1.1, POS7/2 via 10.14.1.0/30
   valid cached adjacency
```

Step 2 Enter the **show ip bgp** EXEC command for the same prefix used in Step 1--192.168.5.0-- to learn which community is assigned to this prefix.

In this example, the output is displayed for the prefix 192.168.5.0. It shows that the community of 100:197 is assigned to this prefix.

Example:

```
Router# show ip bgp 192.168.5.0
BGP routing table entry for 192.168.5.0/24, version 2
Paths: (1 available, best #1)
  Not advertised to any peer
  100
    10.14.1.1 from 10.14.1.1 (32.32.32.32)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 100:197
```

Step 3 Enter the **show cef interface policy-statistics** EXEC command to display the per-interface traffic statistics.

In this example, the output shows the number of packets and bytes that have been assigned to each accounting bucket:

Example:

```
LC-Slot7# show cef interface policy-statistics
POS7/0 is up (if_number 8)
Bucket    Packets               Bytes
1          0                     0
```

```
2          0                      0
3          50                     5000
4          100                    10000
5          100                    10000
6          10                     1000
7          0                      0
8          0                      0
```

Monitoring and Maintaining BGP Policy Accounting

To monitor and maintain the BGP Policy Accounting feature, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router#` **show cef interface** [*type number*] **policy-statistics**	Displays detailed CEF policy statistical information for all interfaces.
`Router#` **show ip bgp** [*network*] [*network mask*] [**longer-prefixes**]	Displays entries in the BGP routing table.
`Router#` **show ip cef** [*network* [*mask*]] [**detail**]	Displays entries in the Forwarding Information Base (FIB) or FIB summary information.

Configuration Examples

- Specifying the Match Criteria for BGP Policy Accounting Example, page 392
- Classifying the IP Traffic and Enabling BGP Policy Accounting Example, page 393

Specifying the Match Criteria for BGP Policy Accounting Example

In the following example, BGP communities are specified in community lists, and a route map named set_bucket is configured to match each of the community lists to a specific accounting bucket using the **set traffic-index** command:

```
ip community-list 30 permit 100:190
ip community-list 40 permit 100:198
ip community-list 50 permit 100:197
ip community-list 60 permit 100:296
!
route-map set_bucket permit 10
match community 30
set traffic-index 2
!
route-map set_bucket permit 20
match community 40
set traffic-index 3
!
route-map set_bucket permit 30
match community 50
set traffic-index 4
```

```
!
route-map set_bucket permit 40
match community 60
set traffic-index 5
```

Classifying the IP Traffic and Enabling BGP Policy Accounting Example

In the following example, BGP policy accounting is enabled on POS interface 7/0 and the **table-map** command is used to modify the bucket number when the IP routing table is updated with routes learned from BGP:

```
router bgp 65000
 table-map set_bucket
 network 10.15.1.0 mask 255.255.255.0
 neighbor 10.14.1.1 remote-as 65100
!
ip classless
ip bgp-community new-format
!
interface POS7/0
 ip address 10.15.1.2 255.255.255.0
 no ip directed-broadcast
 bgp-policy accounting
 no keepalive
 crc 32
 clock source internal
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*
Cisco Express Forwarding (CEF) and distributed CEF (dCEF) commands	*Cisco IOS IP Switching Command Reference*
Cisco Express Forwarding (CEF) and distributed CEF (dCEF) configuration information	"Cisco Express Forwarding Overview" module of the Cisco IOS Switching Services Configuration Guide

MIBs

MIB	MIBs Link
• CISCO-BGP-POLICY-ACCOUNTING-MIB **Note** CISCO-BGP-POLICY-ACCOUNTING-MIB is only available in the Cisco IOS Release 12.0(9)S, 12.0(17)ST, and later releases. This MIB is not available on any mainline and T-train release.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for BGP Policy Accounting

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 24 Feature Information for BGP Policy Accounting

Feature Name	Releases	Feature Information
BGP Policy Accounting	12.0(9)S 12.0(17)ST 12.2(13)T 15.0(1)S 12.2(50)SY	Border Gateway Protocol (BGP) policy accounting measures and classifies IP traffic that is sent to, or received from, different peers. Policy accounting is enabled on an input interface, and counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic. The following commands were introduced or modified: • **bgp-policy** • **set traffic-index** • **show cef interface policy-statistics** • **show ip bgp** • **show ip cef**

BGP Policy Accounting Output Interface Accounting

Border Gateway Protocol (BGP) policy accounting (PA) measures and classifies IP traffic that is sent to, or received from, different peers. Policy accounting was previously available on an input interface only. The BGP Policy Accounting Output Interface Accounting feature introduces several extensions to enable BGP PA on an output interface and to include accounting based on a source address for both input and output traffic on an interface. Counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Feature History for BGP PA Output Interface Accounting

Release	Modification
12.0(9)S	This feature was introduced.
12.0(17)ST	This feature was integrated into Cisco IOS Release 12.0(17)ST.
12.0(22)S	Output interface accounting was added, and the bucket size was increased.
12.3(4)T	This feature was integrated into Cisco IOS Release 12.3(4)T.
12.2(22)S	This feature was integrated into Cisco IOS Release 12.2(22)S.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for BGP PA Output Interface Accounting

Before using the BGP Policy Accounting Output Interface Accounting feature, you must enable BGP and Cisco Express Forwarding or distributed CEF on the router.

Restrictions for BGP PA Output Interface Accounting

The CISCO-BGP-POLICY-ACCOUNTING-MIB is only available in the Cisco IOS Release 12.0(9)S, 12.0(17)ST, 12.2(22)S, and later releases. This MIB is not available on any mainline and T-train release.

Information About BGP PA Output Interface Accounting

- BGP PA Output Interface Accounting, page 398
- Benefits of BGP PA Output Interface Accounting, page 399

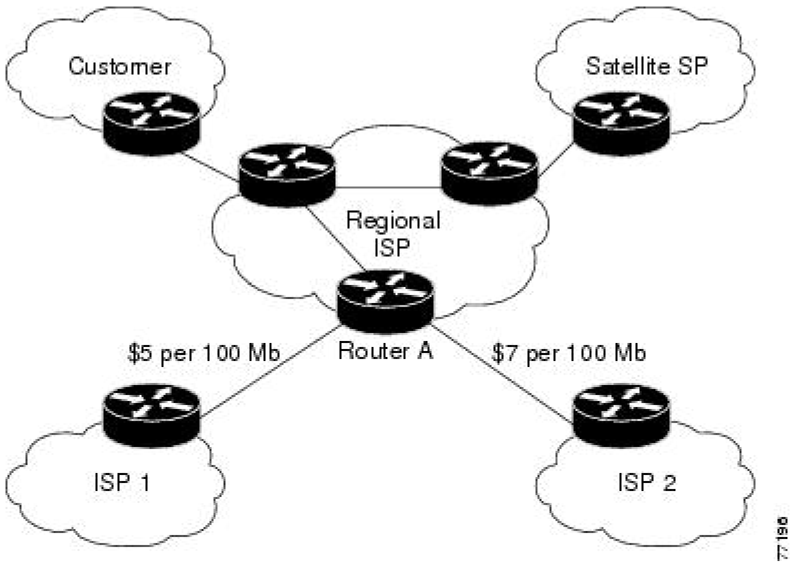
BGP PA Output Interface Accounting

Policy accounting using BGP measures and classifies IP traffic that is sent to, or received from, different peers. Originally, BGP PA was available on an input interface only. BGP PA output interface accounting introduces several extensions to enable BGP PA on an output interface and to include accounting based on a source address for both input and output traffic on an interface. Counters based on parameters such as community list, autonomous system number, or autonomous system path are assigned to identify the IP traffic.

Using the BGP **table-map** command, prefixes added to the routing table are classified by BGP attribute, autonomous system number, or autonomous system path. Packet and byte counters are incremented per input or output interface. A Cisco policy-based classifier maps the traffic into one of eight possible buckets that represent different traffic classes.

Using BGP PA, you can account for traffic according to its origin or the route it traverses. Service providers (SPs) can identify and account for all traffic by customer and can bill accordingly. In the figure below, BGP PA can be implemented in Router A to measure packet and byte volumes in autonomous system buckets. Customers are billed appropriately for traffic that is routed from a domestic, international, or satellite source.

Figure 40 *Sample Topology for BGP Policy Accounting*



BGP policy accounting using autonomous system numbers can be used to improve the design of network circuit peering and transit agreements between Internet service providers (ISPs).

Benefits of BGP PA Output Interface Accounting

Accounting for IP Traffic Differentially

BGP policy accounting classifies IP traffic by autonomous system number, autonomous system path, or community list string, and increments packet and byte counters. Policy accounting can also be based on the source address. Service providers can account for traffic and apply billing according to the origin of the traffic or the route that specific traffic traverses.

Efficient Network Circuit Peering and Transit Agreement Design

Implementing BGP policy accounting on an edge router can highlight potential design improvements for peering and transit agreements.

How to Configure BGP PA Output Interface Accounting

Specifying the Match Criteria for BGP PA

The first task in configuring BGP PA is to specify the criteria that must be matched. Community lists, autonomous system paths, or autonomous system numbers are examples of BGP attributes that can be specified and subsequently matched using a route map. Perform this task to specify the BGP attribute to use for BGP PA and to create the match criteria in a route map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip community-list** {*standard-list-number*| *expanded-list-number*[*regular-expression*] | {**standard**| **expanded**} *community-list-name*} {**permit**| **deny**} {*community-number*| *regular-expression*}
4. **route-map** *map-name* [**permit**| **deny**] [*sequence-number*]
5. **match community-list** *community-list-number* [**exact**]
6. **set traffic-index** *bucket-number*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip community-list** {*standard-list-number*| *expanded-list-number*[*regular-expression*] | {**standard**| **expanded**} *community-list-name*} {**permit**| **deny**} {*community-number*| *regular-expression*} **Example:** `Router(config)# ip community-list 30 permit 100:190`	Creates a community list for BGP and controls access to it. • Repeat this step for each community to be specified.

	Command or Action	Purpose
Step 4	**route-map** *map-name* [**permit**| **deny**] [*sequence-number*] **Example:** `Router(config)# route-map set_bucket permit 10`	Enters route-map configuration mode and defines the conditions for policy routing. • The *map-name* argument identifies a route map. • The optional **permit** and **deny** keywords work with the match and set criteria to control how the packets are accounted for. • The optional *sequence-number* argument indicates the position that a new route map is to have in the list of route maps already configured with the same name.
Step 5	**match community-list** *community-list-number* [**exact**] **Example:** `Router(config-route-map)# match community-list 30`	Matches a BGP community.
Step 6	**set traffic-index** *bucket-number* **Example:** `Router(config-route-map)# set traffic-index 2`	Indicates where to output packets that pass a match clause of a route map for BGP policy accounting.
Step 7	**exit** **Example:** `Router(config-route-map)# exit`	Exits route-map configuration mode and returns to global configuration mode.

Classifying the IP Traffic and Enabling BGP PA

After a route map has been defined to specify match criteria, you must configure a way to classify the IP traffic before enabling BGP policy accounting.

Using the **table-map** command, BGP classifies each prefix that it adds to the routing table according to the match criteria. When the **bgp-policy accounting** command is configured on an interface, BGP policy accounting is enabled.

Perform this task to classify the IP traffic and enable BGP policy accounting.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **table-map** *route-map-name*
5. **network** *network-number* [**mask** *network-mask*]
6. **neighbor** *ip-address* **remote-as** *as-number*
7. **exit**
8. **interface** *type number*
9. **ip address** *ip-address mask*
10. **bgp-policy accounting** [**input**| **output**] [**source**]
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Router(config)# router bgp 65000`	Configures a BGP routing process and enters router configuration mode for the specified routing process. • The *as-number* argument identifies a BGP autonomous system number.
Step 4	**table-map** *route-map-name* **Example:** `Router(config-router)# table-map set_bucket`	Classifies BGP prefixes entered in the routing table.

	Command or Action	Purpose
Step 5	**network** *network-number* [**mask** *network-mask*] **Example:** `Router(config-router)# network 10.15.1.0 mask 255.255.255.0`	Specifies a network to be advertised by the BGP routing process.
Step 6	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Router(config-router)# neighbor 10.14.1.1 remote-as 65100`	Specifies a BGP peer by adding an entry to the BGP routing table.
Step 7	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and returns to global configuration mode.
Step 8	**interface** *type number* **Example:** `Router(config)# interface POS 7/0`	Specifies the interface type and number and enters interface configuration mode. • The *type* argument identifies the type of interface. • The *number* argument identifies the slot and port numbers of the interface. The space between the interface type and number is optional.
Step 9	**ip address** *ip-address mask* **Example:** `Router(config-if)# ip-address 10.15.1.2 255.255.255.0`	Configures the interface with an IP address.
Step 10	**bgp-policy accounting** [**input**| **output**] [**source**] **Example:** `Router(config-if)# bgp-policy accounting input source`	Enables BGP policy accounting for the interface. • Use the optional **input** or **output** keyword to account for traffic either entering or leaving the router. By default, BGP policy accounting is based on traffic entering the router. • Use the optional **source** keyword to account for traffic based on source address.
Step 11	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

Verifying BGP Policy Accounting

Perform this task to verify that BGP policy accounting is operating.

SUMMARY STEPS

1. **show ip cef** [*network*[*mask*]] [**detail**]
2. **show ip bgp** [*network*] [*network-mask*] [**longer-prefixes**]
3. **show cef interface** [*type number*] **policy-statistics**[**input**| **output**]
4. **show cef interface** [*type number*] [**statistics**] [**detail**]

DETAILED STEPS

Step 1 **show ip cef** [*network*[*mask*]] [**detail**]

Enter the **show ip cef** command with the **detail** keyword to learn which accounting bucket is assigned to a specified prefix.

In this example, the output is displayed for the prefix 192.168.5.0. It shows that accounting bucket number 4 (traffic_index 4) is assigned to this prefix.

Example:

```
Router# show ip cef 192.168.5.0 detail
192.168.5.0/24, version 21, cached adjacency to POS7/2
0 packets, 0 bytes, traffic_index 4
  via 10.14.1.1, 0 dependencies, recursive
   next hop 10.14.1.1, POS7/2 via 10.14.1.0/30
   valid cached adjacency
```

Step 2 **show ip bgp** [*network*] [*network-mask*] [**longer-prefixes**]

Enter the **show ip bgp** command for the same prefix used in Step 1--192.168.5.0--to learn which community is assigned to this prefix.

In this example, the output is displayed for the prefix 192.168.5.0. It shows that the community of 100:197 is assigned to this prefix.

Example:

```
Router# show ip bgp 192.168.5.0
BGP routing table entry for 192.168.5.0/24, version 2
Paths: (1 available, best #1)
  Not advertised to any peer
  100
    10.14.1.1 from 10.14.1.1 (32.32.32.32)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 100:197
```

Step 3 **show cef interface** [*type number*] **policy-statistics**[**input**| **output**]

Enter the **show cef interface policy-statistics** command to display the per-interface traffic statistics.

In this example, the output shows the number of packets and bytes that have been assigned to each accounting bucket:

Example:

```
Router# show cef interface policy-statistics input
```

```
FastEthernet1/0/0 is up (if_number 6)
  Corresponding hwidb fast_if_number 6
  Corresponding hwidb firstsw->if_number 6
  BGP based Policy accounting on input is enabled
 Index         Packets          Bytes
     1            9999         999900
     2               0              0
     3               0              0
     4               0              0
     5               0              0
     6               0              0
     7               0              0
     8               0              0
     9               0              0
    10               0              0
    11               0              0
    12               0              0
    13               0              0
    14               0              0
    15               0              0
    16               0              0
    17               0              0
    18               0              0
    19               0              0
    20               0              0
    21               0              0
    22               0              0
    23               0              0
    24               0              0
    25               0              0
    26               0              0
    27               0              0
    28               0              0
    29               0              0
    30               0              0
    31               0              0
    32               0              0
    33               0              0
    34            1234         123400
    35               0              0
    36               0              0
    37               0              0
    38               0              0
    39               0              0
    40               0              0
    41               0              0
    42               0              0
    43               0              0
    44               0              0
    45            1000         100000
    46               0              0
    47               0              0
    48               0              0
    49               0              0
    50               0              0
    51               0              0
    52               0              0
    53               0              0
    54            5123        1198782
    55               0              0
    56               0              0
    57               0              0
    58               0              0
    59               0              0
    60               0              0
    61               0              0
    62               0              0
    63               0              0
    64               0              0
```

Step 4 **show cef interface** [*type number*] [**statistics**] [**detail**]

Enter the **show cef interface**EXEC command to display the state of BGP policy accounting on a specified interface.

In this example, the output shows that BGP policy accounting has been configured to be based on input traffic at Fast Ethernet interface 1/0/0:

Example:

```
Router# show cef interface Fast Ethernet 1/0/0
FastEthernet1/0/0 is up (if_number 6)
  Corresponding hwidb fast_if_number 6
  Corresponding hwidb firstsw->if_number 6
  Internet address is 10.1.1.1/24
  ICMP redirects are always sent
  Per packet load-sharing is disabled
  IP unicast RPF check is disabled
  Inbound access list is not set
  Outbound access list is not set
  IP policy routing is disabled
  BGP based policy accounting on input is enabled
  BGP based policy accounting on output is disabled
  Hardware idb is FastEthernet1/0/0 (6)
  Software idb is FastEthernet1/0/0 (6)
  Fast switching type 1, interface type 18
  IP Distributed CEF switching enabled
  IP Feature Fast switching turbo vector
  IP Feature CEF switching turbo vector
  Input fast flags 0x100, Output fast flags 0x0, Flags 0x0
  ifindex 7(7)
  Slot 1 Slot unit 0 VC -1
  Transmit limit accumulator 0xE8001A82 (0xE8001A82)
  IP MTU 1500
```

Configuration Examples for BGP PA Output Interface Accounting

Specifying the Match Criteria for BGP Policy Accounting Example

In the following example, BGP communities are specified in community lists, and a route map named set_bucket is configured to match each of the community lists to a specific accounting bucket using the **set traffic-index** command:

```
ip community-list 30 permit 100:190
ip community-list 40 permit 100:198
ip community-list 50 permit 100:197
ip community-list 60 permit 100:296
!
route-map set_bucket permit 10
 match community-list 30
 set traffic-index 2
!
route-map set_bucket permit 20
 match community-list 40
 set traffic-index 3
!
```

```
route-map set_bucket permit 30
 match community-list 50
 set traffic-index 4
!
route-map set_bucket permit 40
 match community-list 60
 set traffic-index 5
```

Classifying the IP Traffic and Enabling BGP Policy Accounting Example

In the following example, BGP policy accounting is enabled on POS interface 2/0/0. The policy accounting criteria is based on the source address of the input traffic, and the **table-map** command is used to modify the bucket number when the IP routing table is updated with routes learned from BGP.

```
router bgp 65000
 table-map set_bucket
 network 10.15.1.0 mask 255.255.255.0
 neighbor 10.14.1.1 remote-as 65100
!
ip classless
ip bgp-community new-format
!
interface POS2/0/0
 ip address 10.15.1.2 255.255.255.0
 bgp-policy accounting input source
 no keepalive
 crc 32
 clock source internal
```

Where to Go Next

Additional BGP, CEF, and dCEF command and configuration information is available from the appropriate Cisco IOS command reference or configuration guide documents. For more details, see the "Additional References" section.

Additional References

The following sections provide references related to BGP policy accounting.

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
Switching commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS IP Switching Command Reference*
CEF and dCEF configuration information	"Cisco Express Forwarding Overview" module

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
CISCO-BGP-POLICY-ACCOUNTING-MIB **Note** This MIB is available only in Cisco IOS Release 12.0(9)S, 12.0(17)ST, 12.2(22)S, and later releases. This MIB is not available on any mainline and T-train release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS IP Routing: BGP Command Reference*. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **bgp-policy**
- **set traffic-index**
- **show cef interface**
- **show cef interface policy-statistics**

Glossary

AS --autonomous system. An IP term to describe a routing domain that has its own independent routing policy and is administered by a single authority.

BGP --Border Gateway Protocol. Interdomain routing protocol that exchanges reachability information with other BGP systems.

CEF --Cisco Express Forwarding.

dCEF --distributed Cisco Express Forwarding.

BGP Cost Community

The BGP Cost Community feature introduces the cost extended community attribute. The cost community is a non-transitive extended community attribute that is passed to internal BGP (iBGP) and confederation peers but not to external BGP (eBGP) peers. The cost community feature allows you to customize the local route preference and influence the best path selection process by assigning cost values to specific routes.

In Cisco IOS Release 12.0(27)S, 12.3(8)T, 12.2(25)S, and later releases, support was introduced for mixed EIGRP MPLS VPN network topologies that contain VPN and backdoor links.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for the BGP Cost Community Feature

This document assumes that BGP is configured in your network and that peering has been established.

Restrictions for the BGP Cost Community Feature

- The BGP Cost Community feature can be configured only within an autonomous system or confederation. The cost community is a non-transitive extended community that is passed to iBGP and confederation peers only and is not passed to eBGP peers.
- The BGP Cost Community feature must be supported on all routers in the autonomous system or confederation before cost community filtering is configured. The cost community should be applied consistently throughout the local autonomous system or confederation to avoid potential routing loops.
- Multiple cost community set clauses may be configured with the **set extcommunity cost** command in a single route map block or sequence. However, each set clause must be configured with a different ID value (0-255) for each point of insertion (POI). The ID value determines preference when all other attributes are equal. The lowest ID value is preferred.

Information About the BGP Cost Community Feature

BGP Cost Community Overview

The cost community is a non-transitive extended community attribute that is passed to iBGP and confederation peers but not to eBGP peers. The configuration of the BGP Cost Community feature allows you to customize the BGP best path selection process for a local autonomous system or confederation.

The cost community attribute is applied to internal routes by configuring the **set extcommunity cost** command in a route map. The cost community set clause is configured with a cost community ID number (0-255) and cost number (0-4294967295). The cost number value determines the preference for the path. The path with the lowest cost community number is preferred. Paths that are not specifically configured with the cost community attribute are assigned a default cost number value of 2147483647 (The midpoint between 0 and 4294967295) and evaluated by the best path selection process accordingly. In the case where two paths have been configured with the same cost number value, the path selection process will then prefer the path with the lowest cost community ID. The cost extended community attribute is propagated to iBGP peers when extended community exchange is enabled with the **neighbor send-community** command.

The following commands can be used to apply the route map that is configured with the cost community set clause:

- aggregate-address
- neighbor default-originate route-map {in | out}
- neighbor route-map
- network route-map
- redistribute route-map

How the BGP Cost Community Influences the Best Path Selection Process

The cost community attribute influences the BGP best path selection process at the point of insertion (POI). By default, the POI follows the IGP metric comparison. When BGP receives multiple paths to the same destination, it uses the best path selection process to determine which path is the best path. BGP automatically makes the decision and installs the best path into the routing table. The POI allows you to assign a preference to o a specific path when multiple equal cost paths are available. If the POI is not valid for local best path selection, the cost community attribute is silently ignored.

Multiple paths can be configured with the cost community attribute for the same POI. The path with the lowest cost community ID is considered first. In other words, all of the cost community paths for a specific POI are considered, starting with the one with the lowest cost community. Paths that do not contain the cost community (for the POI and community ID being evaluated) are assigned the default community cost value (2147483647). If the cost community values are equal, then cost community comparison proceeds to the next lowest community ID for this POI.

Note Paths that are not configured with the cost community attribute are considered by the best path selection process to have the default cost-value (half of the maximum value [4294967295] or 2147483647).

Applying the cost community attribute at the POI allows you to assign a value to a path originated or learned by a peer in any part of the local autonomous system or confederation. The cost community can be used as a "tie breaker" during the best path selection process. Multiple instances of the cost community can be configured for separate equal cost paths within the same autonomous system or confederation. For example, a lower cost community value can be applied to a specific exit path in a network with multiple equal cost exits points, and the specific exit path will be preferred by the BGP best path selection process. See the scenario described in "Influencing Route Preference in a Multi-Exit IGP Network".

Cost Community Support for Aggregate Routes and Multipaths

Aggregate routes and multipaths are supported by the BGP Cost Community feature. The cost community attribute can be applied to either type of route. The cost community attribute is passed to the aggregate or multipath route from component routes that carry the cost community attribute. Only unique IDs are passed, and only the highest cost of any individual component route will be applied to the aggregate on a per-ID basis. If multiple component routes contain the same ID, the highest configured cost is applied to the route. For example, the following two component routes are configured with the cost community attribute via an inbound route map:

- 10.0.0.1 (POI=IGP, ID=1, Cost=100)
- 192.168.0.1 (POI=IGP, ID=1, Cost=200)

If these component routes are aggregated or configured as a multipath, the cost value 200 (POI=IGP, ID=1, Cost=200) will be advertised because it is the highest cost.

If one or more component routes does not carry the cost community attribute or if the component routes are configured with different IDs, then the default value (2147483647) will be advertised for the aggregate or multipath route. For example, the following three component routes are configured with the cost community attribute via an inbound route map. However, the component routes are configured with two different IDs.

- 10.0.0.1 (POI=IGP, ID=1, Cost=100)
- 172.16.0.1 (POI=IGP, ID=2, Cost=100)
- 192.168.0.1 (POI=IGP, ID=1, Cost=200)

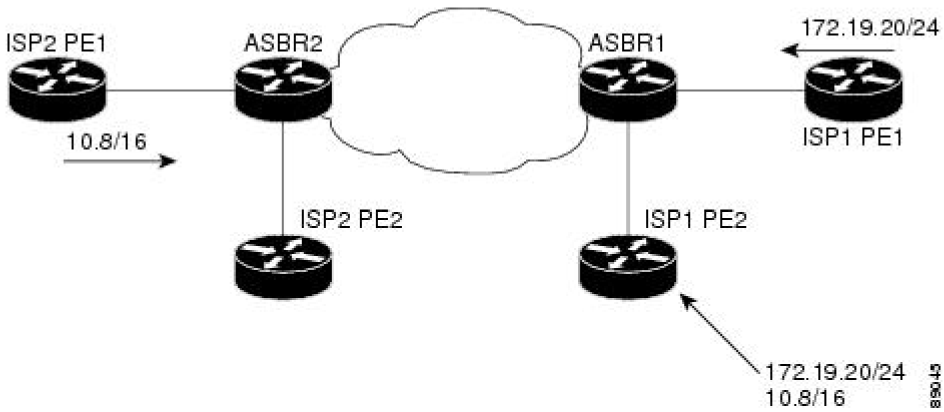
The single advertised path will include the aggregated cost communities as follows:

- {POI=IGP, ID=1, Cost=2147483647} {POI=IGP, ID=2, Cost=2147483647}

Influencing Route Preference in a Multi-Exit IGP Network

The figure below shows an Interior Gateway Protocol (IGP) network with two autonomous system boundary routers (ASBRs) on the edge. Each ASBR has an equal cost path to network 10.8/16.

Figure 41 Multi-Exit Point IGP Network



Both paths are considered to be equal by BGP. If multipath loadsharing is configured, both paths will be installed to the routing table and will be used to load balance traffic. If multipath load balancing is not configured, then BGP will select the path that was learned first as the best path and install this path to the routing table. This behavior may not be desirable under some conditions. For example, the path is learned from ISP1 PE2 first, but the link between ISP1 PE2 and ASBR1 is a low-speed link.

The configuration of the cost community attribute can be used to influence the BGP best path selection process by applying a lower cost community value to the path learned by ASBR2. For example, the following configuration is applied to ASBR2.

```
route-map ISP2_PE1 permit 10
 set extcommunity cost 1 1
 match ip address 13
!
ip access-list 13 permit 10.8.0.0 0.0.255.255
```

The above route map applies a cost community number value of 1 to the 10.8.0.0 route. By default, the path learned from ASBR1 will be assigned a cost community value of 2147483647. Because the path learned from ASBR2 has lower cost community value, this path will be preferred.

BGP Cost Community Support for EIGRP MPLS VPN PE-CE with Backdoor Links

Before EIGRP Site of Origin (SoO) BGP Cost Community support was introduced, BGP preferred locally sourced routes over routes learned from BGP peers. Back door links in an EIGRP MPLS VPN topology will be preferred by BGP if the back door link is learned first. (A back door link, or a route, is a connection that is configured outside of the VPN between a remote and main site. For example, a WAN leased line that connects a remote site to the corporate network).

The "pre-bestpath" point of insertion (POI) was introduced in the BGP Cost Community feature to support mixed EIGRP VPN network topologies that contain VPN and backdoor links. This POI is applied automatically to EIGRP routes that are redistributed into BGP. The "pre-best path" POI carries the EIGRP route type and metric. This POI influences the best path calculation process by influencing BGP to consider this POI before any other comparison step. No configuration is required. This feature is enabled automatically for EIGRP VPN sites when Cisco IOS Release 12.0(27)S is installed to a PE, CE, or back door router.

For information about configuring EIGRP MPLS VPNs, refer to the MPLS VPN Support for EIGRP Between Provider Edge and Customer Edge document in Cisco IOS Release 12.0(27)S.

For more information about the EIGRP MPLS VPN PE-CE Site of Origin (SoO) feature, refer to the EIGRP MPLS VPN PE-CE Site of Origin (SoO) feature documentation in Cisco IOS Release 12.0(27)S.

How to Configure the BGP Cost Community Feature

Configuring the BGP Cost Community

To configure the cost community, perform the task in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **ipv6** [**multicast** | **unicast**] | **vpnv4** [**unicast**]
6. **neighbor** *ip-address* **route-map** *map-name* {**in** | **out**}
7. **exit**
8. **route-map** *map-name* {**permit** | **deny**} [*sequence-number*]
9. **set extcommunity cost** [**igp**] *community-id cost-value*
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 50000`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 10.0.0.1 remote-as 101`	Establishes peering with the specified neighbor or peer-group.
Step 5	**address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **ipv6** [**multicast** | **unicast**] | **vpnv4** [**unicast**] **Example:** `Router(config-router)# address-family ipv4`	Places the router in address family configuration mode.
Step 6	**neighbor** *ip-address* **route-map** *map-name* {**in** | **out**} **Example:** `Router(config-router)# neighbor 10.0.0.1 route-map MAP-NAME in`	Applies an incoming or outgoing route map for the specified neighbor or peer-group.
Step 7	**exit** **Example:** `Router(config-router)# exit`	Exits router configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 8	**route-map** *map-name* {**permit** | **deny**} [*sequence-number*] **Example:** `Router(config)# route-map MAP-NAME permit 10`	Enters route map configuration mode to create or configure a route map.
Step 9	**set extcommunity cost** [**igp**] *community-id cost-value* **Example:** `Router(config-route-map)# set extcommunity cost 1 100`	Creates a set clause to apply the cost community attribute. • Multiple cost community set clauses can be configured in each route map block or sequence. Each cost community set clause must have a different ID (0-255). The cost community set clause with the lowest *cost-value* is preferred by the best path selection process when all other attributes are equal. • Paths that are not configured with the cost community attribute will be assigned the default *cost-value*, which is half of the maximum value (4294967295) or 2147483647.
Step 10	**end** **Example:** `Router(config-route-map)# end`	Exits route map configuration mode and enters privileged EXEC mode.

Verifying the Configuration of the BGP Cost Community

BGP cost community configuration can be verified locally or for a specific neighbor. To verify the local configuration cost community, use the **show route-map** or **show running-config**command. To verify that a specific neighbor carries the cost community, use the **show ip bgp** *ip-address* command. The output from these commands displays the POI (IGP is the default POI), the configured ID, and configured cost. For large cost community values, the output from these commands will also show, with + and - values, the difference between the configured cost and the default cost. See "Verifying the Configuration of the BGP Cost Community" for specific example output.

• Troubleshooting Tips, page 417

Troubleshooting Tips

The **bgp bestpath cost-community ignore** command can be used to disable the evaluation of the cost community attribute to help isolate problems and troubleshoot issues that relate to BGP best path selection.

The **debug ip bgp updates** command can be used to print BGP update messages. The cost community extended community attribute will be displayed in the output of this command when received from a neighbor. A message will also be displayed if a non-transitive extended community if received from an external peer.

Configuration Examples for the BGP Cost Community Feature

BGP Cost Community Configuration Example

The following example configuration shows the configuration of the **set extcommunity cost**command. The following example applies the cost community ID of 1 and cost community value of 100 to routes that are permitted by the route map. This configuration will cause the best path selection process to prefer this route over other equal cost paths that were not permitted by this route map sequence.

```
Router(config)# router bgp 50000
Router(config-router)# neighbor 10.0.0.1 remote-as 50000
Router(config-router)# neighbor 10.0.0.1 update-source Loopback 0
Router(config-router)# address-family ipv4
Router(config-router-af)# neighbor 10.0.0.1 activate
Router(config-router-af)# neighbor 10.0.0.1 route-map COST1 in
Router(config-router-af)# neighbor 10.0.0.1 send-community both
Router(config-router-af)# exit
Router(config)# route-map COST1 permit 10
Router(config-route-map)# match ip-address 1
Router(config-route-map)# set extcommunity cost 1 100
```

BGP Cost Community Verification Examples

BGP cost community configuration can be verified locally or for a specific neighbor. To verify the local configuration cost community, use the **show route-map** or **show running-config**command. To verify that a specific neighbor carries the cost community, use the **show ip bgp** *ip-address* command.

The output of the **show route-map**command will display locally configured route-maps, match, set, continue clauses, and the status and configuration of the cost community attribute. The following sample output is similar to the output that will be displayed:

```
Router# show route-map
route-map COST1, permit, sequence 10
  Match clauses:
    as-path (as-path filter): 1
  Set clauses:
    extended community Cost:igp:1:100
  Policy routing matches: 0 packets, 0 bytes
route-map COST1, permit, sequence 20
  Match clauses:
    ip next-hop (access-lists): 2
  Set clauses:
    extended community Cost:igp:2:200
  Policy routing matches: 0 packets, 0 bytes
route-map COST1, permit, sequence 30
  Match clauses:
    interface FastEthernet0/0
    extcommunity (extcommunity-list filter):300
  Set clauses:
    extended community Cost:igp:3:300
  Policy routing matches: 0 packets, 0 bytes
```

The following sample output shows locally configured routes with large cost community values:

```
Router# show route-map
route-map set-cost, permit, sequence 10
  Match clauses:
```

```
  Set clauses:
    extended community RT:1:1 RT:2:2 RT:3:3 RT:4:4 RT:5:5 RT:6:6 RT:7:7
      RT:100:100 RT:200:200 RT:300:300 RT:400:400 RT:500:500 RT:600:600
      RT:700:700 additive
    extended community Cost:igp:1:4294967295 (default+2147483648)
      Cost:igp:2:200 Cost:igp:3:300 Cost:igp:4:400
      Cost:igp:5:2147483648 (default+1) Cost:igp:6:2147484648 (default+1001)
      Cost:igp:7:2147284648 (default-198999)
  Policy routing matches: 0 packets, 0 bytes
```

The output of the **show running config**command will display match, set, and continue clauses that are configured within a route-map. The following sample output is filtered to show only the relevant part of the running configuration:

```
Router# show running-config | begin route-map
route-map COST1 permit 20
 match ip next-hop 2
 set extcommunity cost igp 2 200
!
route-map COST1 permit 30
 match interface FastEthernet0/0
 match extcommunity 300
 set extcommunity cost igp 3 300
.
.
.
```

The output of the **show ip bgp** *ip-address* command can be used to verify if a specific neighbor carries a path that is configured with the cost community attribute. The cost community attribute information is displayed in the "Extended Community" field. The POI, the cost community ID, and the cost community number value are displayed. The following sample output shows that neighbor 172.16.1.2 carries a cost community with an ID of 1 and a cost of 100:

```
Router# show ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 2
Paths: (1 available, best #1)
  Not advertised to any peer
  2 2 2
    172.16.1.2 from 172.16.1.2 (172.16.1.2)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Extended Community: Cost:igp:1:100
```

If the specified neighbor is configured with the default cost community number value or if the default value is assigned automatically for cost community evaluation, "default" with + and - values will be displayed after the cost community number value in the output.

Where to Go Next

For more information about the EIGRP MPLS VPN PE-CE Site of Origin (SoO) feature, refer to the "EIGRP MPLS VPN PE-CE Site of Origin (SoO)" module .

Additional References

For additional information related to the BGP Cost Community feature, refer to the following references:

Related Documents

Related Topic	Document Title
BGP Best Path Selection	"BGP Best Path Selection Algorithm"
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL: http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
draft-retana-bgp-custom-decision-00.txt	BGP Custom Decision Process

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS IP Routing: BGP Command Reference*. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **bgp bestpath cost-community ignore**
- **debug ip bgp updates**
- **set extcommunity cost**

Feature Information for BGP Cost Community

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 25 Feature Information for BGP Cost Community

Feature Name	Releases	Feature Information
BGP Cost Community	12.0(24)S 12.3(2)T 12.2(18)S 12.2(27)SBC 15.0(1)S	The BGP Cost Community feature introduces the cost extended community attribute. The cost community is a non-transitive extended community attribute that is passed to internal BGP (iBGP) and confederation peers but not to external BGP (eBGP) peers. The cost community feature allows you to customize the local route preference and influence the best path selection process by assigning cost values to specific routes. The following commands were introduced or modified: **bgp bestpath cost-community ignore**, **debug ip bgp updates**, and **set extcommunity cost**.

Feature Name	Releases	Feature Information
BGP Cost Community Support for EIGRP MPLS VPN PE-CE with Backdoor Links	12.0(27)S 12.3(8)T 12.2(25)S	Back door links in an EIGRP MPLS VPN topology will be preferred by BGP if the back door link is learned first. The "pre-bestpath" point of insertion (POI) was introduced in the BGP Cost Community feature to support mixed EIGRP VPN network topologies that contain VPN and backdoor links. This POI is applied automatically to EIGRP routes that are redistributed into BGP and the POI influences the best path calculation process by influencing BGP to consider this POI before any other comparison step. No configuration is required. This feature is enabled automatically for EIGRP VPN sites when Cisco IOS Release 12.0(27)S, 12.3(8)T, 12,2(25)S or later releases, is installed to a PE, CE, or back door router. No commands were introduced or modified.

BGP Support for IP Prefix Import from Global Table into a VRF Table

The BGP Support for IP Prefix Import from Global Table into a VRF Table feature introduces the capability to import IPv4 unicast prefixes from the global routing table into a Virtual Private Network (VPN) routing/forwarding (VRF) instance table using an import route map.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for BGP Support for IP Prefix Import from Global Table into a VRF Table

- Border Gateway Protocol (BGP) peering sessions are established.
- CEF or dCEF (for distributed platforms) is enabled on all participating routers.

Restrictions for BGP Support for IP Prefix Import from Global Table into a VRF Table

- Only IPv4 unicast and multicast prefixes can be imported into a VRF with this feature.
- A maximum of five VRF instances per router can be created to import IPv4 prefixes from the global routing table.
- IPv4 prefixes imported into a VRF using this feature cannot be imported into a VPNv4 VRF.

Information About BGP Support for IP Prefix Import from Global Table into a VRF Table

Importing IPv4 Prefixes into a VRF

The BGP Support for IP Prefix Import from Global Table into a VRF Table feature introduces the capability to import IPv4 unicast prefixes from the global routing table into a Virtual Private Network (VPN) routing/forwarding instance (VRF) table using an import route map. This feature extends the functionality of VRF import-map configuration to allow IPv4 prefixes to be imported into a VRF based on a standard community. Both IPv4 unicast and multicast prefixes are supported. No Multiprotocol Label Switching (MPLS) or route target (import/export) configuration is required.

IP prefixes are defined as match criteria for the import map through standard Cisco filtering mechanisms. For example, an IP access-list, an IP prefix-list, or an IP as-path filter is created to define an IP prefix or IP prefix range, and then the prefix or prefixes are processed through a match clause in a route map. Prefixes that pass through the route map are imported into the specified VRF per the import map configuration.

Black Hole Routing

The BGP Support for IP Prefix Import from Global Table into a VRF Table feature can be configured to support Black Hole Routing (BHR). BHR is a method that allows the administrator to block undesirable traffic, such as traffic from illegal sources or traffic generated by a Denial of Service (DoS) attack, by dynamically routing the traffic to a dead interface or to a host designed to collect information for investigation, mitigating the impact of the attack on the network. Prefixes are looked up, and packets that come from unauthorized sources are blackholed by the ASIC at line rate.

Classifying Global Traffic

The BGP Support for IP Prefix Import from Global Table into a VRF Table feature can be used to classify global IP traffic based on physical location or class of service. Traffic is classified based on administration policy and then imported into different VRFs. On a college campus, for example, network traffic could be divided into an academic network and residence network traffic, a student network and faculty network, or a dedicated network for multicast traffic. After the traffic is divided along administration policy, routing decisions can be configured with the MPLS VPN--VRF Selection Using Policy Based Routing feature or the MPLS VPN--VRF Selection Based on Source IP Address feature.

Unicast Reverse Path Forwarding

Unicast Reverse Path Forwarding (Unicast RPF) can be optionally configured with the BGP Support for IP Prefix Import from Global Table into a VRF Table feature. Unicast RPF is used to verify that the source address is in the Forwarding Information Base (FIB). The **ip verify unicast vrf** command is configured in interface configuration mode and is enabled for each VRF. This command has **permit** and **deny**keywords that are used to determine if the traffic is forwarded or dropped after Unicast RPF verification.

How to Import IP Prefixes from Global Table into a VRF Table

Defining IPv4 IP Prefixes to Import

IPv4 unicast or multicast prefixes are defined as match criteria for the import route map using standard Cisco filtering mechanisms. This task uses an IP access-list and an IP prefix-list.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**]
4. **ip prefix-list** *prefix-list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] [**log**] **Example:** `Router(config)# access-list 50 permit 10.1.1.0 0.0.0.255`	Creates an access list and defines a range of IP prefixes to import into the VRF table. • The example creates a standard access list numbered 50. This filter will permit traffic from any host with an IP address in the 10.1.1.0/24 subnet.
Step 4	**ip prefix-list** *prefix-list-name* [**seq** *seq-value*] {**deny** *network* / *length* | **permit** *network* / *length*} [**ge** *ge-value*] [**le** *le-value*] **Example:** `Router(config)# ip prefix-list COLORADO permit 10.24.240.0/22`	Creates a prefix list and defines a range of IP prefixes to import into the VRF table. • The example creates an IP prefix list named COLORADO. This filter will permit traffic from any host with an IP address in the 10.24.240.0/22 subnet.

Creating the VRF and the Import Route Map

The IP prefixes that are defined for import are then processed through a match clause in a route map. IP prefixes that pass through the route map are imported into the VRF. A maximum of 5 VRFs per router can be configured to import IPv4 prefixes from the global routing table. 1000 prefixes per VRF are imported by default. You can manually configure from 1 to 2,147,483,647 prefixes for each VRF. We recommend that you use caution if you manually configure the prefix import limit. Configuring the router to import too many prefixes can interrupt normal router operation.

No MPLS or route target (import/export) configuration is required.

Import actions are triggered when a new routing update is received or when routes are withdrawn. During the initial BGP update period, the import action is postponed to allow BGP to convergence more quickly. Once BGP converges, incremental BGP updates are evaluated immediately and qualified prefixes are imported as they are received.

The following syslog message is introduced by the BGP Support for IP Prefix Import from Global Table into a VRF Table feature. It will be displayed when more prefixes are available for import than the user-defined limit:

```
00:00:33: %BGP-3-AFIMPORT_EXCEED: IPv4 Multicast prefixes imported to multicast vrf
exceed the limit 2
```

You can either increase the prefix limit or fine-tune the import route map filter to reduce the number of candidate routes.

Note

- Only IPv4 unicast and multicast prefixes can be imported into a VRF with this feature.
- A maximum of five VRF instances per router can be created to import IPv4 prefixes from the global routing table.
- IPv4 prefixes imported into a VRF using this feature cannot be imported into a VPNv4 VRF.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **rd** *route-distinguisher*
5. **import ipv4** {**unicast** | **multicast**} *[prefix-limit]* **map** *route-map*
6. **exit**
7. **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]
8. **match ip address** {*acl-number* [*acl-number* | *acl-name*] | *acl-name* [*acl-name* | *acl-number*] | **prefix-list** *prefix-list-name* [*prefix-list-name*]}
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf GREEN`	Creates a VRF routing table and specifies the VRF name (or tag). • The **ip vrf** *vrf-name* command creates a VRF routing table and a CEF table, and both are named using the *vrf-name* argument. Associated with these tables is the default route distinguisher value.

	Command or Action	Purpose
Step 4	**rd** *route-distinguisher* **Example:** `Router(config-vrf)# rd 100:10`	Creates routing and forwarding tables for the VRF instance. • There are two formats for configuring the route distinguisher argument. It can be configured in the as-number:network number (ASN:nn) format, as shown in the example, or it can be configured in the IP address:network number format (IP-address:nn).
Step 5	**import ipv4** {**unicast** | **multicast**} *[prefix-limit]* **map** *route-map* **Example:** `Router(config-vrf)# import ipv4 unicast 1000 map UNICAST`	Creates an import map to import IPv4 prefixes from the global routing table to a VRF table. • Unicast or multicast prefixes are specified. • Up to a 1000 prefixes will be imported by default. The *prefix-limit* argument is used to specify a limit from 1 to 2,147,483,647 prefixes. • The route-map that defines the prefixes to import is specified after the **map** keyword is entered. • The example creates an import map that will import up to 1000 unicast prefixes that pass through the route map named UNICAST.
Step 6	**exit** **Example:** `Router(config-vrf)# exit`	Exits VRF configuration mode and enters global configuration mode.
Step 7	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Router(config)# route-map UNICAST permit 10`	Defines the conditions for redistributing routes from one routing protocol into another, or enables policy routing. • The route map name must match the route map specified in Step 5. • The example creates a route map named UNICAST.
Step 8	**match ip address** {*acl-number* [*acl-number* | *acl-name*] | *acl-name* [*acl-name* | *acl-number*] | **prefix-list** *prefix-list-name* [*prefix-list-name*]} **Example:** `Router(config-route-map)# match ip address 50`	Distributes any routes that have a destination network number address that is permitted by a standard or extended access list, and performs policy routing on matched packets. • Both IP access lists and IP prefix lists are supported. • The example configures the route map to use standard access list 50 to define match criteria.
Step 9	**end** **Example:** `Router(config-route-map)# end`	Exits route-map configuration mode and returns to privileged EXEC mode.

Filtering on the Ingress Interface

The BGP Support for IP Prefix Import from Global Table into a VRF Table feature can be configured globally or on a per-interface basis. We recommend that you apply it to ingress interfaces to maximize performance.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number* [*name-tag*]
4. **ip policy route-map** *map-tag*
5. **ip verify unicast vrf** *vrf-name* {**deny** | **permit**}
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* [*name-tag*] **Example:** `Router(config)# interface Ethernet0/0`	Configures an interface and enters interface configuration mode.
Step 4	**ip policy route-map** *map-tag* **Example:** `Router(config-if)# ip policy route-map UNICAST`	Identifies a route map to use for policy routing on an interface. • The example attaches the route map named UNICAST to the interface.

	Command or Action	Purpose
Step 5	**ip verify unicast vrf** *vrf-name* {**deny** | **permit**} **Example:** `Router(config-if)# ip verify unicast vrf GREEN permit`	(Optional) Enables Unicast Reverse Path Forwarding verification for the specified VRF. • The example enables verification for the VRF named GREEN. Traffic that passes verification will be forwarded.
Step 6	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Verifying Global IP Prefix Import

Perform the steps in this task to display information about the VRFs that are configured with the BGP Support for IP Prefix Import from Global Table into a VRF Table feature and to verify that global IP prefixes are imported into the specified VRF table.

SUMMARY STEPS

1. **enable**
2. **show ip bgp vpnv4** {**all** | **rd** *route-distinguisher* | **vrf** *vrf-name*}
3. **show ip vrf** [**brief** | **detail** | **interfaces** | **id**] [*vrf-name*]

DETAILED STEPS

Step 1 **enable**
Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Router# enable
```

Step 2 **show ip bgp vpnv4** {**all** | **rd** *route-distinguisher* | **vrf** *vrf-name*}
Displays VPN address information from the BGP table. The output displays the import route map, the traffic type (unicast or multicast), the default or user-defined prefix import limit, the actual number of prefixes that are imported, and individual import prefix entries.

Example:

```
Router# show ip bgp vpnv4 all
BGP table version is 15, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
```

```
Route Distinguisher: 100:1 (default for vrf academic)
Import Map: ACADEMIC, Address-Family: IPv4 Unicast, Pfx Count/Limit: 6/1000
*> 10.50.1.0/24     172.17.2.2                             0 2 3 ?
*> 10.50.2.0/24     172.17.2.2                             0 2 3 ?
*> 10.50.3.0/24     172.17.2.2                             0 2 3 ?
*> 10.60.1.0/24     172.17.2.2                             0 2 3 ?
*> 10.60.2.0/24     172.17.2.2                             0 2 3 ?
*> 10.60.3.0/24     172.17.2.2                             0 2 3 ?
Route Distinguisher: 200:1 (default for vrf residence)
Import Map: RESIDENCE, Address-Family: IPv4 Unicast, Pfx Count/Limit: 3/1000
*> 10.30.1.0/24     172.17.2.2                  0          0 2 i
*> 10.30.2.0/24     172.17.2.2                  0          0 2 i
*> 10.30.3.0/24     172.17.2.2                  0          0 2 i
Route Distinguisher: 300:1 (default for vrf BLACKHOLE)
Import Map: BLACKHOLE, Address-Family: IPv4 Unicast, Pfx Count/Limit: 3/1000
*> 10.40.1.0/24     172.17.2.2                  0          0 2 i
*> 10.40.2.0/24     172.17.2.2                  0          0 2 i
*> 10.40.3.0/24     172.17.2.2                  0          0 2 i
Route Distinguisher: 400:1 (default for vrf multicast)
Import Map: MCAST, Address-Family: IPv4 Multicast, Pfx Count/Limit: 2/2
*> 10.70.1.0/24     172.17.2.2                  0          0 2 i
*> 10.70.2.0/24     172.17.2.2                  0          0 2 i
```

Step 3 **show ip vrf** [**brief** | **detail** | **interfaces** | **id**] [*vrf-name*]

Displays defined VRFs and their associated interfaces. The output displays the import route map, the traffic type (unicast or multicast), and the default or user-defined prefix import limit. The following example output shows that the import route map named UNICAST is importing IPv4 unicast prefixes and that the prefix import limit is 1000.

Example:

```
Router# show ip vrf detail
VRF academic; default RD 100:10; default VPNID <not set>
VRF Table ID = 1
  No interfaces
  Connected addresses are not in global routing table
  Export VPN route-target communities
    RT:100:10
  Import VPN route-target communities
    RT:100:10
  Import route-map for ipv4 unicast: UNICAST (prefix limit: 1000)
  No export route-map
```

Configuration Examples for BGP Support for IP Prefix Import from Global Table into a VRF Table

Configuring Global IP Prefix Import Example

The following example imports unicast prefixes into the VRF named *green* using an IP prefix list and a route map:

This example starts in global configuration mode:

```
!
```

```
ip prefix-list COLORADO seq 5 permit 10.131.64.0/19
ip prefix-list COLORADO seq 10 permit 172.31.2.0/30
ip prefix-list COLORADO seq 15 permit 172.31.1.1/32
!
ip vrf green
 rd 200:1
 import ipv4 unicast map UNICAST
 route-target export 200:10
 route-target import 200:10
!
 exit
!
route-map UNICAST permit 10
 match ip address prefix-list COLORADO
!
 exit
```

Verifying Global IP Prefix Import Example

The **show ip vrf**command or the **show ip bgp vpnv4** command can be used to verify that prefixes are imported from the global routing table to the VRF table.

The following example from the **show ip vrf** command shows the import route map named UNICAST is importing IPv4 unicast prefixes and the prefix import limit is 1000:

```
Router# show ip vrf detail
VRF green; default RD 200:1; default VPNID <not set>
  Interfaces:
    Se2/0
VRF Table ID = 1
  Export VPN route-target communities
    RT:200:10
  Import VPN route-target communities
    RT:200:10
  Import route-map for ipv4 unicast: UNICAST (prefix limit: 1000)
  No export route-map
  VRF label distribution protocol: not configured
  VRF label allocation mode: per-prefix
VRF red; default RD 200:2; default VPNID <not set>
  Interfaces:
    Se3/0
VRF Table ID = 2
  Export VPN route-target communities
    RT:200:20
  Import VPN route-target communities
    RT:200:20
  No import route-map
  No export route-map
  VRF label distribution protocol: not configured
  VRF label allocation mode: per-prefix
```

The following example from the **show ip bgp vpnv4**command shows the import route map names, the prefix import limit and the actual number of imported prefixes, and the individual import entries:

```
Router# show ip bgp vpnv4 all

BGP table version is 18, local router ID is 10.131.127.252
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200:1 (default for vrf green)
Import Map: UNICAST, Address-Family: IPv4 Unicast, Pfx Count/Limit: 1/1000
*>i10.131.64.0/19   10.131.95.252            0    100      0 i
*> 172.16.1.1/32    172.16.2.1               0         32768 i
*> 172.16.2.0/30    0.0.0.0                  0         32768 i
*>i172.31.1.1/32    10.131.95.252            0    100      0 i
*>i172.31.2.0/30    10.131.95.252            0    100      0 i
Route Distinguisher: 200:2 (default for vrf red)
```

```
*> 172.16.1.1/32    172.16.2.1               0         32768 i
*> 172.16.2.0/30    0.0.0.0                  0         32768 i
*>i172.31.1.1/32    10.131.95.252            0    100      0 i
*>i172.31.2.0/30    10.131.95.252            0    100      0 i
```

Additional References

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
MPLS Layer 3 VPN configuration tasks	"Configuring MPLS Layer 3 VPNs"
VRF selection using policy based routing	"Directing MPLS VPN Traffic Using Policy-Based Routing"
VRF selection based on source IP address	"MPLS VPN-- VRF Selection Based on Source IP Address"

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for BGP Support for IP Prefix Import from Global Table into a VRF Table

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 26 Feature Information for BGP Support for IP Prefix Import from Global Table into a VRF Table

Feature Name	Releases	Feature Information
BGP Support for IP Prefix Import from Global Table into a VRF Table	12.0(29)S 12.2(25)S 12.2(27)SBC 12.2(33)SRA 12.2(33)SXH 12.3(14)T 15.0(1)S	The BGP Support for IP Prefix Import from Global Table into a VRF Table feature introduces the capability to import IPv4 unicast prefixes from the global routing table into a Virtual Private Network (VPN) routing/forwarding (VRF) instance table using an import route map. The following commands were introduced or modified by this feature: **debug ip bgp import**, **import ipv4**, **ip verify unicast vrf**.

Per-VRF Assignment of BGP Router ID

The Per-VRF Assignment of BGP Router ID feature introduces the ability to have VRF-to-VRF peering in Border Gateway Protocol (BGP) on the same router. BGP is designed to refuse a session with itself because of the router ID check. The per-VRF assignment feature allows a separate router ID per VRF using a new keyword in the existing **bgp router-id** command. The router ID can be manually configured for each VRF or can be assigned automatically either globally under address family configuration mode or for each VRF.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Per-VRF Assignment of BGP Router ID

Before you configure this feature, Cisco Express Forwarding (CEF) or distributed CEF (dCEF) must be enabled in the network, and basic BGP peering is assumed to be running in the network.

Information About Per-VRF Assignment of BGP Router ID

BGP Router ID

The BGP router identifier (ID) is a 4-byte field that is set to the highest IP address on the router. Loopback interface addresses are considered before physical interface addresses because loopback interfaces are more stable than physical interfaces. The BGP router ID is used in the BGP algorithm for determining the best path to a destination where the preference is for the BGP router with the lowest router ID. It is possible to manually configure the BGP router ID using the **bgp router-id** command to influence the best path algorithm.

Per-VRF Router ID Assignment

In Cisco IOS Release 12.2(31)SB2, 12.2(33)SRA, 12.2(33)SXH, 12.4(20)T, and later releases, support for configuring separate router IDs for each Virtual Private Network (VPN) routing/forwarding (VRF) instance was introduced. The Per-VRF Assignment of BGP Router ID feature introduces the ability to have VRF-to-VRF peering in Border Gateway Protocol (BGP) on the same router. BGP is designed to refuse a session with itself because of the router ID check. The per-VRF assignment feature allows a separate router ID per VRF using a new keyword in the existing **bgp router-id** command. The router ID can be manually configured for each VRF or can be assigned automatically either globally under address family configuration mode or for each VRF.

Route Distinguisher

A route distinguisher (RD) creates routing and forwarding tables and specifies the default route distinguisher for a VPN. The RD is added to the beginning of an IPv4 prefix to change it into a globally unique VPN-IPv4 prefix. An RD can be composed in one of two ways: with an autonomous system number and an arbitrary number or with an IP address and an arbitrary number.

You can enter an RD in either of these formats:

- Enter a 16-bit autonomous system number, a colon, and a 32-bit number. For example:

45000:3

- Enter a 32-bit IP address, a colon, and a 16-bit number. For example:

192.168.10.15:1

How to Configure Per-VRF Assignment of BGP Router ID

There are two main ways to configure a BGP router ID for each separate VRF. To configure a per-VRF BGP router ID manually, you must perform the first three tasks listed below. To automatically assign a BGP router ID to each VRF, perform the first task and the fourth task.

- Configuring VRF Instances, page 438
- Associating VRF Instances with Interfaces, page 440
- Manually Configuring a BGP Router ID per VRF, page 443
- Automatically Assigning a BGP Router ID per VRF, page 448

Configuring VRF Instances

Perform this task to configure VRF instances to be used with the per-VRF assignment tasks. In this task, a VRF instance named vrf_trans is created. To make the VRF functional, a route distinguisher is created.

When the route distinguisher is created, the routing and forwarding tables are created for the VRF instance named vrf_trans.

This task assumes that you have CEF or dCEF enabled.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip vrf** *vrf-name*
4. **rd** *route-distinguisher*
5. **route-target** {**import** | **both**} *route-target-ext-community*
6. **route-target** {**export** | **both**} *route-target-ext-community*
7. **exit**
8. Repeat Step 3 through Step 7 for each VRF to be defined.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Router(config)# ip vrf vrf_trans`	Defines a VRF instance and enters VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Router(config-vrf)# rd 45000:2`	Creates routing and forwarding tables for a VRF and specifies the default RD for a VPN. • Use the *route-distinguisher* argument to specify the default RD for a VPN. There are two formats you can use to specify an RD. For more details, see the GUID-1750DF8F-F6DA-4C2D-9134-0F2982E23636. • In this example, the RD uses an autonomous system number with the number 2 after the colon.

	Command or Action	Purpose
Step 5	**route-target** {**import** | **both**} *route-target-ext-community* **Example:** `Router(config-vrf)# route-target import 55000:5`	Creates a route-target extended community for a VRF. • Use the **import** keyword to import routing information from the target VPN extended community. • Use the **both** keyword to both import routing information from and export routing information to the target VPN extended community. • Use the *route-target-ext-community* argument to specify the VPN extended community.
Step 6	**route-target** {**export** | **both**} *route-target-ext-community* **Example:** `Router(config-vrf)# route-target export 55000:1`	Creates a route-target extended community for a VRF. • Use the **export** keyword to export routing information to the target VPN extended community. • Use the **both** keyword to both import routing information from and export routing information to the target VPN extended community. • Use the *route-target-ext-community* argument to specify the VPN extended community.
Step 7	**exit** **Example:** `Router(config-vrf)# exit`	Exits VRF configuration mode and returns to global configuration mode.
Step 8	Repeat Step 3 through Step 7 for each VRF to be defined.	--

Associating VRF Instances with Interfaces

Perform this task to associate VRF instances with interfaces to be used with the per-VRF assignment tasks. In this task, a VRF instance named vrf_trans is associated with a serial interface.

Make a note of the IP addresses for any interface to which you want to associate a VRF instance because the **ip vrf forwarding** command removes the IP address. Step 8 allows you to reconfigure the IP address.

- This task assumes that you have CEF or dCEF enabled.
- This task assumes that VRF instances have been configured in the Configuring VRF Instances, page 438.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip address** *ip-address mask* [**secondary**]
5. **exit**
6. **interface** *type number*
7. **ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*]
8. **ip address** *ip-address mask* [**secondary**]
9. Repeat Step 5 through Step 8 for each VRF to be associated with an interface.
10. **end**
11. show ip vrf [**brief** | **detail** | **interfaces** | **id**] [*vrf-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface loopback0`	Configures an interface type and enters interface configuration mode. • In this example, loopback interface 0 is configured.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 172.16.1.1 255.255.255.255`	Configures an IP address. • In this example, the loopback interface is configured with an IP address of 172.16.1.1.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Router(config)# interface serial2/0`	Configures an interface type and enters interface configuration mode. • In this example, serial interface 2/0 is configured.
Step 7	**ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*] **Example:** `Router(config-if)# ip vrf forwarding vrf_trans`	Associates a VRF with an interface or subinterface. • In this example, the VRF named vrf_trans is associated with serial interface 2/0. **Note** Executing this command on an interface removes the IP address. The IP address should be reconfigured.
Step 8	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 192.168.4.1 255.255.255.0`	Configures an IP address. • In this example, serial interface 2/0 is configured with an IP address of 192.168.4.1.
Step 9	Repeat Step 5 through Step 8 for each VRF to be associated with an interface.	--
Step 10	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 11	show ip vrf [**brief** | **detail** | **interfaces** | **id**] [*vrf-name*] **Example:** `Router# show ip vrf interfaces`	(Optional) Displays the set of defined VRFs and associated interfaces. • In this example, the output from this command shows the VRFs that have been created and their associated interfaces.

Examples

The following output show s that two VRF instances named vrf_trans and vrf_users were configured on two serial interfaces.

```
Router# show ip vrf interfaces
```

```
Interface          IP-Address       VRF                        Protocol
Serial2            192.168.4.1      vrf_trans                  up
Serial3            192.168.5.1      vrf_user                   up
```

Manually Configuring a BGP Router ID per VRF

Perform this task to manually configure a BGP router ID for each VRF. In this task, several address family configurations are shown and the router ID is configured in the IPv4 address family mode for one VRF instance. Step 22 shows you how to repeat certain steps to permit the configuration of more than one VRF on the same router.

This task assumes that you have previously created the VRF instances and associated them with interfaces. For more details, see the Configuring VRF Instances, page 438 and the Associating VRF Instances with Interfaces, page 440.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **no bgp default ipv4-unicast**
5. **bgp log-neighbor-changes**
6. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
7. **neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number*
8. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
9. **neighbor** {*ip-address*| *peer-group-name*} **activate**
10. **neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**}
11. **exit-address-family**
12. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
13. **redistribute connected**
14. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
15. **neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]
16. **neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*]
17. **neighbor** {*ip-address*| *peer-group-name*} **activate**
18. **neighbor** *ip-address* **allowas-in** [*number*]
19. **no auto-summary**
20. **no synchronization**
21. **bgp router-id** {*ip-address*| **auto-assign**}
22. Repeat Step 11 to Step 21 to configure another VRF instance.
23. **end**
24. **show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:** `Router(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured with the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast**router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 6	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor is an internal neighbor.

	Command or Action	Purpose
Step 7	**neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 update-source loopback0`	Allows BGP sessions to use any operational interface for TCP connections. • In this example, BGP TCP connections for the specified neighbor are sourced with the IP address of the loopback interface rather than the best local address.
Step 8	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family vpnv4`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example creates a VPNv4 address family session.
Step 9	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 172.16.1.1 activate`	Activates the neighbor under the VPNv4 address family. • In this example, the neighbor 172.16.1.1 is activated.
Step 10	**neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**} **Example:** `Router(config-router-af)# neighbor 172.16.1.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • In this example, an extended communities attribute is sent to the neighbor at 172.16.1.1.
Step 11	**exit-address-family** **Example:** `Router(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.

	Command or Action	Purpose
Step 12	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 vrf vrf_trans`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example specifies that the VRF instance named vrf_trans is to be associated with subsequent IPv4 address family configuration commands.
Step 13	**redistribute connected** **Example:** `Router(config-router-af)# redistribute connected`	Redistributes from one routing domain into another routing domain. • In this example, the **connected** keyword is used to represent routes that are established automatically when IP is enabled on an interface. • Only the syntax applicable to this step is displayed. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 14	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.1.1 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor at 192.168.1.1 is an external neighbor.
Step 15	**neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 local-as 50000 no-prepend`	Customizes the AS_PATH attribute for routes received from an eBGP neighbor. • The autonomous system number from the local BGP routing process is prepended to all external routes by default. • Use the **no-prepend** keyword to not prepend the local autonomous system number to any routes received from the eBGP neighbor. • In this example, routes from the neighbor at 192.168.1.1 will not contain the local autonomous system number.
Step 16	**neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 ebgp-multihop 2`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected. • In this example, BGP is configured to allow connections to or from neighbor 192.168.1.1, which resides on a network that is not directly connected.

	Command or Action	Purpose
Step 17	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.1 activate`	Activates the neighbor under the IPV4 address family. • In this example, the neighbor 192.168.1.1 is activated.
Step 18	**neighbor** *ip-address* **allowas-in** [*number*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 allowas-in 1`	Configures provider edge (PE) routers to allow the readvertisement of all prefixes that contain duplicate autonomous system numbers. • In the example, the PE router with autonomous system number 45000 is configured to allow prefixes from the VRF vrf-trans. The neighboring PE router with the IP address 192.168.1.1 is set to be readvertised once to other PE routers with the same autonomous system number.
Step 19	**no auto-summary** **Example:** `Router(config-router-af)# no auto-summary`	Disables automatic summarization and sends subprefix routing information across classful network boundaries.
Step 20	**no synchronization** **Example:** `Router(config-router-af)# no synchronization`	Enables the Cisco IOS software to advertise a network route without waiting for synchronization with an Internal Gateway Protocol (IGP).
Step 21	**bgp router-id** {*ip-address*| **auto-assign**} **Example:** `Router(config-router-af)# bgp router-id 10.99.1.1`	Configures a fixed router ID for the local BGP routing process. • In this example, the specified BGP router ID is assigned for the VRF instance associated with this IPv4 address family configuration.
Step 22	Repeat Step 11 to Step 21 to configure another VRF instance.	--
Step 23	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 24	**show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*} **Example:** `Router# show ip bgp vpnv4 all`	(Optional) Displays VPN address information from the BGP table. • In this example, the complete VPNv4 database is displayed. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Multiprotocol Label Switching Command Reference*.

Examples

The following sample output assumes that two VRF instances named vrf_trans and vrf_user were configured each with a separate router ID. The router ID is shown next to the VRF name.

```
Router# show ip bgp vpnv4 all
BGP table version is 5, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1 (default for vrf vrf_trans) VRF Router ID 10.99.1.2
*> 192.168.4.0      0.0.0.0                  0         32768 ?
Route Distinguisher: 42:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
*> 192.168.5.0      0.0.0.0                  0         32768 ?
```

Automatically Assigning a BGP Router ID per VRF

Perform this task to automatically assign a BGP router ID for each VRF. In this task, a loopback interface is associated with a VRF and the **bgp router-id** command is configured at the router configuration level to automatically assign a BGP router ID to all VRF instances. Step 9 shows you how to repeat certain steps to configure each VRF that is to be associated with an interface. Step 30 shows you how to configure more than one VRF on the same router.

This task assumes that you have previously created the VRF instances. For more details, see the Configuring VRF Instances, page 438.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip address** *ip-address mask* [**secondary**]
5. **exit**
6. **interface** *type number*
7. **ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*]
8. **ip address** *ip-address mask* [**secondary**]
9. Repeat Step 5 through Step 8 for each VRF to be associated with an interface.
10. **exit**
11. **router bgp** *autonomous-system-number*
12. **bgp router-id** {*ip-address*| **vrf auto-assign**}
13. **no bgp default ipv4-unicast**
14. **bgp log-neighbor-changes**
15. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
16. **neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number*
17. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
18. **neighbor** {*ip-address*| *peer-group-name*} **activate**
19. **neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**}
20. **exit-address-family**
21. **address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]}
22. **redistribute connected**
23. **neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number*
24. **neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]]
25. **neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*]
26. **neighbor** {*ip-address*| *peer-group-name*} **activate**
27. **neighbor** *ip-address* **allowas-in** [*number*]
28. **no auto-summary**
29. **no synchronization**
30. Repeat Step 20 to Step 29 to configure another VRF instance.
31. **end**
32. **show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface loopback0`	Configures an interface type and enters interface configuration mode. • In this example, loopback interface 0 is configured.
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 172.16.1.1 255.255.255.255`	Configures an IP address. • In this example, the loopback interface is configured with an IP address of 172.16.1.1.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Router(config)# interface loopback1`	Configures an interface type and enters interface configuration mode. • In this example, loopback interface 1 is configured.
Step 7	**ip vrf forwarding** *vrf-name* [**downstream** *vrf-name2*] **Example:** `Router(config-if)# ip vrf forwarding vrf_trans`	Associates a VRF with an interface or subinterface. • In this example, the VRF named vrf_trans is associated with loopback interface 1. **Note** Executing this command on an interface removes the IP address. The IP address should be reconfigured.

	Command or Action	Purpose
Step 8	**ip address** *ip-address mask* [**secondary**] **Example:** `Router(config-if)# ip address 10.99.1.1 255.255.255.255`	Configures an IP address. • In this example, loopback interface 1 is configured with an IP address of 10.99.1.1.
Step 9	Repeat Step 5 through Step 8 for each VRF to be associated with an interface.	--
Step 10	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 11	**router bgp** *autonomous-system-number* **Example:** `Router(config)# router bgp 45000`	Enters router configuration mode for the specified routing process.
Step 12	**bgp router-id** {*ip-address*| **vrf auto-assign**} **Example:** `Router(config-router)# bgp router-id vrf auto-assign`	Configures a fixed router ID for the local BGP routing process. • In this example, a BGP router ID is automatically assigned for each VRF instance.
Step 13	**no bgp default ipv4-unicast** **Example:** `Router(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured with the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast**router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 14	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.

	Command or Action	Purpose
Step 15	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 remote-as 45000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor is an internal neighbor.
Step 16	**neighbor** {*ip-address*| *peer-group-name*} **update-source** *interface-type interface-number* **Example:** `Router(config-router)# neighbor 192.168.1.1 update-source loopback0`	Allows BGP sessions to use any operational interface for TCP connections. • In this example, BGP TCP connections for the specified neighbor are sourced with the IP address of the loopback interface rather than the best local address.
Step 17	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family vpnv4`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example creates a VPNv4 address family session.
Step 18	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 172.16.1.1 activate`	Activates the neighbor under the VPNv4 address family. • In this example, the neighbor 172.16.1.1 is activated.
Step 19	**neighbor** {*ip-address*| *peer-group-name*} **send-community**{**both**| **standard**| **extended**} **Example:** `Router(config-router-af)# neighbor 172.16.1.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • In this example, an extended communities attribute is sent to the neighbor at 172.16.1.1.

	Command or Action	Purpose
Step 20	**exit-address-family** **Example:** `Router(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 21	**address-family** {**ipv4** [**mdt** | **multicast** | **unicast** [**vrf** *vrf-name*] | **vrf** *vrf-name*] | **vpnv4** [**unicast**]} **Example:** `Router(config-router)# address-family ipv4 vrf vrf_trans`	Enters address family configuration mode to configure BGP peers to accept address-family-specific configurations. • The example specifies that the VRF instance named vrf_trans is to be associated with subsequent IPv4 address family configuration mode commands.
Step 22	**redistribute connected** **Example:** `Router(config-router-af)# redistribute connected`	Redistributes from one routing domain into another routing domain. • In this example, the **connected** keyword is used to represent routes that are established automatically when IP is enabled on an interface. • Only the syntax applicable to this step is displayed. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 23	**neighbor** {*ip-address*| *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Router(config-router-af)# neighbor 192.168.1.1 remote-as 40000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local router. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor at 192.168.1.1 is an external neighbor.
Step 24	**neighbor** *ip-address* **local-as** *autonomous-system-number* [**no-prepend** [**replace-as** [**dual-as**]]] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 local-as 50000 no-prepend`	Customizes the AS_PATH attribute for routes received from an eBGP neighbor. • The autonomous system number from the local BGP routing process is prepended to all external routes by default. • Use the **no-prepend** keyword to not prepend the local autonomous system number to any routes received from the eBGP neighbor. • In this example, routes from the neighbor at 192.168.1.1 will not contain the local autonomous system number.

	Command or Action	Purpose
Step 25	**neighbor** {*ip-address*| *peer-group-name*} **ebgp-multihop**[*ttl*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 ebgp-multihop 2`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected. • In this example, BGP is configured to allow connections to or from neighbor 192.168.1.1, which resides on a network that is not directly connected.
Step 26	**neighbor** {*ip-address*| *peer-group-name*} **activate** **Example:** `Router(config-router-af)# neighbor 192.168.1.1 activate`	Activates the neighbor under the IPV4 address family. • In this example, the neighbor 192.168.1.1 is activated.
Step 27	**neighbor** *ip-address* **allowas-in** [*number*] **Example:** `Router(config-router-af)# neighbor 192.168.1.1 allowas-in 1`	Configures provider edge (PE) routers to allow the readvertisement of all prefixes that contain duplicate autonomous system numbers. • In the example, the PE router with autonomous system number 45000 is configured to allow prefixes from the VRF vrf-trans. The neighboring PE router with the IP address 192.168.1.1 is set to be readvertised once to other PE routers with the same autonomous system number.
Step 28	**no auto-summary** **Example:** `Router(config-router-af)# no auto-summary`	Disables automatic summarization and sends subprefix routing information across classful network boundaries.
Step 29	**no synchronization** **Example:** `Router(config-router-af)# no synchronization`	Enables the Cisco IOS software to advertise a network route without waiting for synchronization with an Internal Gateway Protocol (IGP).
Step 30	Repeat Step 20 to Step 29 to configure another VRF instance.	--
Step 31	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 32	**show ip bgp vpnv4** {**all**| **rd** *route-distinguisher*| **vrf** *vrf-name*} **Example:** `Router# show ip bgp vpnv4 all`	(Optional) Displays VPN address information from the BGP table. • In this example, the complete VPNv4 database is displayed. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Multiprotocol Label Switching Command Reference*.

Examples

The following sample output assumes that two VRF instances named vrf_trans and vrf_user were configured, each with a separate router ID. The router ID is shown next to the VRF name.

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 172.16.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1 (default for vrf vrf_trans) VRF Router ID 10.99.1.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.1.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 10:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 42:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.1.2/32     172.22.1.1               0             0 2 1 ?
```

Configuration Examples for Per-VRF Assignment of BGP Router ID

Manually Configuring a BGP Router ID per VRF Examples

The following example shows how to configure two VRFs--vrf_trans and vrf_user--with sessions between each other on the same router. The BGP router ID for each VRF is configured manually under separate IPv4 address families. The **show ip bgp vpnv4** command can be used to verify that the router IDs have been configured for each VRF. The configuration starts in global configuration mode.

```
ip vrf vrf_trans
```

```
 rd 45000:1
 route-target export 50000:50
 route-target import 40000:1
!
ip vrf vrf_user
 rd 65500:1
 route-target export 65500:1
 route-target import 65500:1
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Ethernet0/0
 ip vrf forwarding vrf_trans
 ip address 172.22.1.1 255.255.0.0
!
interface Ethernet1/0
 ip vrf forwarding vrf_user
 ip address 172.23.1.1 255.255.0.0
!
router bgp 45000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.168.3.1 remote-as 45000
 neighbor 192.168.3.1 update-source Loopback0
 !
 address-family vpnv4
  neighbor 192.168.3.1 activate
  neighbor 192.168.3.1 send-community extended
  exit-address-family
 !
 address-family ipv4 vrf vrf_user
  redistribute connected
  neighbor 172.22.1.1 remote-as 40000
  neighbor 172.22.1.1 local-as 50000 no-prepend
  neighbor 172.22.1.1 ebgp-multihop 2
  neighbor 172.22.1.1 activate
  neighbor 172.22.1.1 allowas-in 1
  no auto-summary
  no synchronization
  bgp router-id 10.99.1.1
  exit-address-family
 !
 address-family ipv4 vrf vrf_trans
  redistribute connected
  neighbor 172.23.1.1 remote-as 50000
  neighbor 172.23.1.1 local-as 40000 no-prepend
  neighbor 172.23.1.1 ebgp-multihop 2
  neighbor 172.23.1.1 activate
  neighbor 172.23.1.1 allowas-in 1
  no auto-summary
  no synchronization
  bgp router-id 10.99.1.2
  exit-address-family
```

After the configuration, the output of the **show ip bgp vpnv4 all** command shows the router ID displayed next to the VRF name:

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 45000:1 (default for vrf vrf_trans) VRF Router ID 10.99.1.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.2.2/32     0.0.0.0                  0         32768 ?
```

```
Route Distinguisher: 50000:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.2.2/32     172.22.1.1               0             0 2 1 ?
```

The output of the **show ip bgp vpnv4 vrf** command for a specified VRF displays the router ID in the output header:

```
Router# show ip bgp vpnv4 vrf vrf_user
BGP table version is 43, local router ID is 10.99.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.2.2/32     172.22.1.1               0             0 2 1 ?
```

The output of the **show ip bgp vpnv4 vrf summary** command for a specified VRF displays the router ID in the first line of the output:

```
Router# show ip bgp vpnv4 vrf vrf_user summary
BGP router identifier 10.99.1.1, local AS number 45000
BGP table version is 43, main routing table version 43
8 network entries using 1128 bytes of memory
8 path entries using 544 bytes of memory
16/10 BGP path/bestpath attribute entries using 1856 bytes of memory
6 BGP AS-PATH entries using 144 bytes of memory
3 BGP extended community entries using 72 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 3744 total bytes of memory
BGP activity 17/0 prefixes, 17/0 paths, scan interval 15 secs
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
172.22.1.1      4     2      20      21       43    0    0 00:12:33        3
```

When the path is sourced in the VRF, the correct router ID is displayed in the output of the **show ip bgp vpnv4 vrf**command for a specified VRF and network address:

```
Router# show ip bgp vpnv4 vrf vrf_user 172.23.0.0
BGP routing table entry for 65500:1:172.23.0.0/8, version 22
Paths: (1 available, best #1, table vrf_user)
  Advertised to update-groups:
     2          3
  Local
    0.0.0.0 from 0.0.0.0 (10.99.1.1)
      Origin incomplete, metric 0, localpref 100, weight 32768, valid, sourced, best
      Extended Community: RT:65500:1
```

Automatically Assigning a BGP Router ID per VRF Examples

The following three configuration examples show different methods of configuring BGP to automatically assign a separate router ID to each VRF instance:

Globally Automatically Assigned Router ID Using Loopback Interface IP Addresses

The following example shows how to configure two VRFs--vrf_trans and vrf_user--with sessions between each other on the same router. Under router configuration mode, BGP is globally configured to automatically assign each VRF a BGP router ID. Loopback interfaces are associated with individual VRFs to source an IP address for the router ID. The **show ip bgp vpnv4** command can be used to verify that the router IDs have been configured for each VRF.

```
ip vrf vrf_trans
 rd 45000:1
 route-target export 50000:50
 route-target import 40000:1
!
ip vrf vrf_user
 rd 65500:1
 route-target export 65500:1
 route-target import 65500:1
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip vrf forwarding vrf_user
 ip address 10.99.1.1 255.255.255.255
!
interface Loopback2
 ip vrf forwarding vrf_trans
 ip address 10.99.2.2 255.255.255.255
!
interface Ethernet0/0
 ip vrf forwarding vrf_trans
 ip address 172.22.1.1 255.0.0.0
!
interface Ethernet1/0
 ip vrf forwarding vrf_user
 ip address 172.23.1.1 255.0.0.0
!
router bgp 45000
 bgp router-id vrf auto-assign
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.168.3.1 remote-as 45000
 neighbor 192.168.3.1 update-source Loopback0
 !
address-family vpnv4
  neighbor 192.168.3.1 activate
  neighbor 192.168.3.1 send-community extended
  exit-address-family
 !
 address-family ipv4 vrf vrf_user
  redistribute connected
  neighbor 172.22.1.1 remote-as 40000
  neighbor 172.22.1.1 local-as 50000 no-prepend
  neighbor 172.22.1.1 ebgp-multihop 2
  neighbor 172.22.1.1 activate
  neighbor 172.22.1.1 allowas-in 1
  no auto-summary
  no synchronization
  exit-address-family
 !
 address-family ipv4 vrf vrf_trans
  redistribute connected
  neighbor 172.23.1.1 remote-as 50000
  neighbor 172.23.1.1 local-as 2 no-prepend
  neighbor 172.23.1.1 ebgp-multihop 2
  neighbor 172.23.1.1 activate
  neighbor 172.23.1.1 allowas-in 1
  no auto-summary
  no synchronization
  exit-address-family
```

After the configuration, the output of the **show ip bgp vpnv4 all** command shows the router ID displayed next to the VRF name. Note that the router IDs used in this example are sourced from the IP addresses configured for loopback interface 1 and loopback interface 2. The router IDs are the same as in the Manually Configuring a BGP Router ID per VRF Examples, page 455.

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 45000:1 (default for vrf vrf_trans) VRF Router ID 10.99.2.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.1.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 50000:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.1.2/32     172.22.1.1               0             0 2 1 ?
```

Globally Automatically Assigned Router ID with No Default Router ID

The following example shows how to configure a router and associate a VRF that is automatically assigned a BGP router ID when no default router ID is allocated.

```
ip vrf vpn1
 rd 45000:1
 route-target export 45000:1
 route-target import 45000:1
!
interface Loopback0
 ip vrf forwarding vpn1
 ip address 10.1.1.1 255.255.255.255
!
interface Ethernet0/0
 ip vrf forwarding vpn1
 ip address 172.22.1.1 255.0.0.0
!
router bgp 45000
 bgp router-id vrf auto-assign
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 !
 address-family ipv4 vrf vpn1
  neighbor 172.22.1.2 remote-as 40000
  neighbor 172.22.1.2 activate
  no auto-summary
  no synchronization
  exit-address-family
```

Assuming that a second router is configured to establish a session between the two routers, the output of the **show ip interface brief** command shows only the VRF interfaces that are configured.

```
Router# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
Ethernet0/0            172.22.1.1      YES NVRAM  up                    up
```

```
Ethernet1/0           unassigned      YES NVRAM  administratively down down
Serial2/0             unassigned      YES NVRAM  administratively down down
Serial3/0             unassigned      YES NVRAM  administratively down down
Loopback0             10.1.1.1        YES NVRAM  up                    up
```

The **show ip vrf** command can be used to verify that a router ID is assigned for the VRF:

```
Router# show ip vrf
Name                             Default RD          Interfaces
  vpn1                             45000:1             Loopback0
                                                       Ethernet0/0
VRF session is established:
```

Per-VRF Automatically Assigned Router ID

The following example shows how to configure two VRFs--vrf_trans and vrf_user--with sessions between each other on the same router. Under the IPv4 address family associated with an individual VRF, BGP is configured to automatically assign a BGP router ID. Loopback interfaces are associated with individual VRFs to source an IP address for the router ID. The output of the **show ip bgp vpnv4** command can be used to verify that the router IDs have been configured for each VRF.

```
ip vrf vrf_trans
 rd 45000:1
 route-target export 50000:50
 route-target import 40000:1
!
ip vrf vrf_user
 rd 65500:1
 route-target export 65500:1
 route-target import 65500:1
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface Loopback1
 ip vrf forwarding vrf_user
 ip address 10.99.1.1 255.255.255.255
!
interface Loopback2
 ip vrf forwarding vrf_trans
 ip address 10.99.2.2 255.255.255.255
!
interface Ethernet0/0
 ip vrf forwarding vrf_trans
 ip address 172.22.1.1 255.0.0.0
!
interface Ethernet1/0
 ip vrf forwarding vrf_user
 ip address 172.23.1.1 255.0.0.0
!
router bgp 45000
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 192.168.3.1 remote-as 45000
 neighbor 192.168.3.1 update-source Loopback0
 !
address-family vpnv4
  neighbor 192.168.3.1 activate
  neighbor 192.168.3.1 send-community extended
  exit-address-family
!
 address-family ipv4 vrf vrf_user
  redistribute connected
  neighbor 172.22.1.1 remote-as 40000
  neighbor 172.22.1.1 local-as 50000 no-prepend
  neighbor 172.22.1.1 ebgp-multihop 2
  neighbor 172.22.1.1 activate
  neighbor 172.22.1.1 allowas-in 1
  no auto-summary
  no synchronization
```

```
  bgp router-id auto-assign
  exit-address-family
 !
 address-family ipv4 vrf vrf_trans
  redistribute connected
  neighbor 172.23.1.1 remote-as 50000
  neighbor 172.23.1.1 local-as 40000 no-prepend
  neighbor 172.23.1.1 ebgp-multihop 2
  neighbor 172.23.1.1 activate
  neighbor 172.23.1.1 allowas-in 1
  no auto-summary
  no synchronization
  bgp router-id auto-assign
  exit-address-family
```

After the configuration, the output of the **show ip bgp vpnv4 all** command shows the router ID displayed next to the VRF name. Note that the router IDs used in this example are sourced from the IP addresses configured for loopback interface 1 and loopback interface 2.

```
Router# show ip bgp vpnv4 all
BGP table version is 43, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 45000:1 (default for vrf vrf_trans) VRF Router ID 10.99.2.2
*> 172.22.0.0       0.0.0.0                  0         32768 ?
r> 172.23.0.0       172.23.1.1               0             0 3 1 ?
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
*> 10.52.1.0/24     172.23.1.1                             0 3 1 ?
*> 10.52.2.1/32     172.23.1.1                             0 3 1 3 i
*> 10.52.3.1/32     172.23.1.1                             0 3 1 3 i
*> 10.99.1.1/32     172.23.1.1               0             0 3 1 ?
*> 10.99.1.2/32     0.0.0.0                  0         32768 ?
Route Distinguisher: 50000:1
*>i10.21.1.1/32     192.168.3.1              0    100      0 2 i
Route Distinguisher: 65500:1 (default for vrf vrf_user) VRF Router ID 10.99.1.1
r> 172.22.0.0       172.22.1.1               0             0 2 1 ?
*> 172.23.0.0       0.0.0.0                  0         32768 ?
*> 10.21.1.1/32     172.22.1.1                             0 2 1 2 i
*>i10.52.1.0/24     192.168.3.1              0    100      0 ?
*>i10.52.2.1/32     192.168.3.1              0    100      0 3 i
*>i10.52.3.1/32     192.168.3.1              0    100      0 3 i
*> 10.99.1.1/32     0.0.0.0                  0         32768 ?
*> 10.99.1.2/32     172.22.1.1               0             0 2 1 ?
```

Additional References

The following sections provide references related to the Per-VRF Assignment of BGP Router ID feature.

Related Documents

Related Topic	Document Title
BGP commands: complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS IP Routing: BGP Command Reference*
MPLS commands: complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Multiprotocol Label Switching Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS IP Routing: BGP Command Reference*. For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases*, at http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html.

- **bgp router-id**
- **show ip bgp vpnv4**

Feature Information for Per-VRF Assignment of BGP Router ID

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 27 Feature Information for Per-VRF Assignment of BGP Router ID

Feature Name	Releases	Feature Information
Per-VRF Assignment of BGP Router ID	12.2(31)SB2 12.2(33)SRA 12.2(33)SXH 12.4(20)T 15.0(1)S	The Per-VRF Assignment of BGP Router ID feature introduces the ability to have VRF-to-VRF peering in Border Gateway Protocol (BGP) on the same router. BGP is designed to refuse a session with itself because of the router ID check. The per-VRF assignment feature allows a separate router ID per VRF using a new keyword in the existing **bgp router-id** command. The router ID can be manually configured for each VRF or can be assigned automatically either globally under address family configuration mode or for each VRF. The following commands were introduced or modified by this feature: **bgp router-id**, **show ip bgp vpnv4**.

BGP Next Hop Unchanged

In an external BGP (eBGP) session, by default, the router changes the next hop attribute of a BGP route (to its own address) when the router sends out a route. The BGP Next Hop Unchanged feature allows BGP to send an update to an eBGP multihop peer with the next hop attribute unchanged.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Information About BGP Next Hop Unchanged

BGP Next Hop Unchanged

In an external BGP (eBGP) session, by default, the router changes the next hop attribute of a BGP route (to its own address) when the router sends out a route. If the BGP Next Hop Unchanged feature is configured, BGP will send routes to an eBGP multihop peer without modifying the next hop attribute. The next hop attribute is unchanged.

Note There is an exception to the default behavior of the router changing the next hop attribute of a BGP route when the router sends out a route. When the next hop is in the same subnet as the peering address of the eBGP peer, the next hop is not modified. This is referred to as third party next-hop.

The BGP Next Hop Unchanged feature provides flexibility when designing and migrating networks. It can be used only between eBGP peers configured as multihop. It can be used in a variety of scenarios between two autonomous systems. One scenario is when multiple autonomous systems are connected that share the same IGP, or at least the routers have another way to reach each other's next hops (which is why the next hop can remain unchanged).

A common use of this feature is to configure Multiprotocol Label Switching (MPLS) inter-AS with multihop MP-eBGP for VPNv4 between RRs.

Another common use of this feature is a VPNv4 inter-AS Option C configuration, as defined in RFC4364, Section 10. In this configuration, VPNv4 routes are passed among autonomous systems between RR of different autonomous systems. The RRs are several hops apart, and have **neighbor next-hop unchanged** configured. PEs of different autonomous systems establish an LSP between them (via a common IGP or by advertising the next-hops--that lead to the PEs--via labeled routes among the ASBRs--routes from different autonomous systems separated by one hop). PEs are able to reach the next hops of the PEs in another AS via the LSPs, and can therefore install the VPNv4 routes in the VRF RIB.

Restriction

The BGP Next Hop Unchanged feature can be configured only between multihop eBGP peers. The following error message will be displayed if you try to configure this feature for a directly connected neighbor:

```
%BGP: Can propagate the nexthop only to multi-hop EBGP neighbor
```

How to Configure BGP Next Hop Unchanged

- Configuring the BGP Next Hop Unchanged for an eBGP Peer, page 467

Configuring the BGP Next Hop Unchanged for an eBGP Peer

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family** {**ipv4** [**unicast**]| **vpnv4** [**unicast**]}
5. **neighbor** *ip-address* **remote-as** *as-number*
6. **neighbor** *ip-address* **activate**
7. **neighbor** *ip-address* **ebgp-multihop** *ttl*
8. **neighbor** *ip-address* **next-hop-unchanged**
9. **end**
10. **show ip bgp**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Router(config)# router bgp 65535`	Enters router configuration mode, and creates a BGP routing process.
Step 4	**address-family** {**ipv4** [**unicast**]| **vpnv4** [**unicast**]} **Example:** `Router(config-router-af)# address-family vpnv4`	Enters address family configuration mode to configure BGP peers to accept address family specific configurations.

	Command or Action	Purpose
Step 5	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Router(config-router-af)# neighbor 10.0.0.100 remote-as 65600`	Adds an entry to the BGP neighbor table.
Step 6	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 10.0.0.100 activate`	Enables the exchange of information with the peer.
Step 7	**neighbor** *ip-address* **ebgp-multihop** *ttl* **Example:** `Router(config-router-af)# neighbor 10.0.0.100 ebgp-multihop 255`	Configures the local router to accept and initiate connections to external peers that reside on networks that are not directly connected.
Step 8	**neighbor** *ip-address* **next-hop-unchanged** **Example:** `Router(config-router-af)# neighbor 10.0.0.100 next-hop-unchanged`	Configures the router to send BGP updates to the specified eBGP peer without modifying the next hop attribute.
Step 9	**end** **Example:** `Router(config-router-af)# end`	Exits address family configuration mode, and enters privileged EXEC mode.
Step 10	**show ip bgp** **Example:** `Router# show ip bgp`	(Optional) Displays entries in the BGP routing table. • The output will indicate if the **neighbor next-hop-unchanged** command has been configured for the selected address.

Configuration Example for BGP Next Hop Unchanged

• Example BGP Next Hop Unchanged for an eBGP Peer, page 469

Example BGP Next Hop Unchanged for an eBGP Peer

The following example configures a multihop eBGP peer at 10.0.0.100 in a remote AS. When the local router sends updates to that peer, it will send them without modifying the next hop attribute.

```
router bgp 65535
 address-family ipv4
 neighbor 10.0.0.100 remote-as 65600
 neighbor 10.0.0.100 activate
 neighbor 10.0.0.100 ebgp-multihop 255
 neighbor 10.0.0.100 next-hop-unchanged
 end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*
BGP configuration tasks	*IP Routing: BGP Configuration Guide*
MPLS configuration tasks	*MPLS Configuration Guide*
BGP Outbound Route Map on Route Reflector to Set IP Next Hop for iBGP Peer	"Configuring Internal BGP Features"

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for BGP Next Hop Unchanged

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 28 Feature Information for BGP Next Hop Unchanged

Feature Name	Releases	Feature Configuration Information
BGP Next Hop Unchanged	12.0(22)S 12.0(16)ST 12.2 12.2(14)S 15.0(1)S	The BGP Next Hop Unchanged feature allows BGP to send an update to an eBGP multihop peer with the next hop attribute unchanged. The following command was added by this feature: **neighbor next-hop-unchanged**.