CHAPTER 2

Configuring Route Maps and Policy-Based Routing

Perhaps one of the most colorful descriptions for route maps is that route maps are like duct tape for the network—not necessarily because they can be used to fix or mend something broken, but because they can be applied to numerous situations to address many issues. At times, they may not be the most "pretty solutions," but they will be very effective. After you learn to configure and use route maps, you will soon see why some engineers refer to them as *route tape*. In policy-based routing (PBR), for instance, you may use a route map when traffic has to follow a particular path through the internetwork. This path may differ from the path the routing protocol wants to forward traffic on. PBR, along with route maps, enables the network engineer to essentially override the route table and influence which way traffic flows.

You also can apply route maps in a number of ways. The following list contains some of the more common and powerful applications of route maps:

- Route filtering during redistribution between routing protocols
- Route control and attribute modification on BGP neighbors
- Route metric modification or *tagging* during redistribution between routing protocols
- Policy-based routing (PBR)

After you have route maps in your engineering tool kit, you will have one of the most powerful and versatile configuration options available on Cisco routers. This chapter discusses how to configure and use route maps and how to configure PBR.

Route Map Overview

Route maps are much like the "If . . . Then . . ." statements of many programming languages. *If* a certain condition is true, *then* do something. Route maps enable you to define routing policy that will be considered before the router examines its forwarding table; therefore, you can define routing policy that takes precedence over the different route processes. This is why route maps are some of the most powerful commands you can use on a router. Example 2-1 highlights route map logic.

Example 2-1 *Route Map Logic*

```
route-map route_map_name permit 10
 match criteria_1
 set perform_action_1
route-map route-map_name permit 20
 match criteria_2
 set perform_action_2
 set perform_action_3
route-map route-map_name permit 30
 match criteria_3 criteria_4 criteria_5
 set perform_action_2
 set perform_action_4
 set perform_action_5
route-map route-map_name deny 65536      ←implicit deny at the end
 match everything
```

In a nutshell, route maps work in the following manner:

1. Essentially, a process—whether it is a redistribution process, policy routing, or some other process such as Network Address Translation (NAT)—calls a route map by a text-based name.
2. The route map, in turn, has conditions or **match** statements, which are usually, but not always, an access list or extended access list. Border Gateway Protocol (BGP), for instance, can match on an autonomous system number (ASN) or different attributes. The **match** statement(s) can be followed by **set** statements.

   If the **match** statement returns a true result, the set statement(s) are executed.

Example 2-2 shows how a route map functions during redistribution.

Example 2-2 *Route Map Function During Redistribution*

```
router ospf 2001
 redistribute eigrp 65001 subnets route-map route_map_name   ←Call the route-map
                                                ←and send EIGRP routes for comparison
!
route-map route_map_name permit 10     ←Route-map with the lowest sequence number
                                        gets executed first
 match ip address access_list          ←Call access-list, the IF of the route-map
 set condition                         ←If access-list is true, THEN do something
!
route-map route_map_name permit 20     ←Next highest sequence number
                                       gets executed
 match ip address access_list          ←Call access-list, the IF of the route-map
 set condition                         ←If access-list is true, THEN do something
!
route-map route_map_name deny 65536    ←Implicit deny at the end all route-maps
 match ip address all_routes           This will not show up in the config
```

The next example is the syntax of an actual route map. Example 2-3 demonstrates how a route map can be applied during redistribution.

Example 2-3 *Route Map Application During Redistribution*

```
router ospf 65
 log-adjacency-changes
 log-adjacency-changes
 redistribute eigrp 65001 subnets route-map set_tag ←Call the route-map "set_tag"
 network 10.10.3.0 0.0.0.255 area 0
 default-metric 10
!
access-list 10 permit 172.16.32.0 0.0.0.255   ←Match the 172.16.32.0/24 subnet
access-list 11 permit 172.16.1.0 0.0.0.255    ← Match the 172.16.1.0/24 subnet
!
route-map set_tag permit 100      ←Route-map "set_tag"
 match ip address 10              ←Call access-list 10, if this is true then…
 set tag 10                       ←If access-list is true set the tag of 10
!
route-map set_tag permit 200      ←If no match above, try and match the following:
 match ip address 11                 ←access list 11
 set metric-type type-1           ←If the ACL is true, set the OSPF metric type to 1
 set tag 11                       ←and set a tag of 11
!
route-map set_tag permit 300
 set tag 300                      ←All other routes get a tag of 300
!
```

In the preceding example, a route map is used to control and tag the routes from Enhanced Interior Gateway Routing Protocol (EIGRP) when they are redistributed into Open Shortest Path First (OSPF). During the OSPF redistribution process, a route map titled set_tag is called. The route map consists of three parts. The first part calls access control list (ACL) 10, which will permit the network 172.16.32.x and set a tag of 10. The second part calls ACL 11, which in turn matches IP address 172.16.1.x. If a match occurs, the metric will be set such that when the route is redistributed, it becomes an OSPF type 1 route; finally, the tag will be set to 11. The last part of the route map doesn't call an ACL, so all routes are matched, and the set condition is applied. In this example, the router is setting the tag to 300. You can set tags in this manner to help document the network, or you can use the tags to identify routes that you may want to filter or perform some other action on.

Route maps have the following common characteristics:

- Route maps are executed in the order of the lowest sequence number to the highest. You can edit or modify maps by using the sequence number.
- If a match is found within a route map instance, execution of further route map instances stops.

- You can use route maps to permit or deny the information found true by the **match** statements.
- If multiple **match** statements are called within a single route map instance, all **match** statements must match for the route map instance to yield a true result.
- If route maps are applied in a policy-routing environment, packets that do not meet the match criteria are then forwarded according to the route table.
- If there is no **match** statement in the route map instance, all routes and packets are matched. The **set** statement will apply to all routes or packets.
- If there is not a corresponding ACL to the **match** statement in the route map instance, all routes are matched. The **set** statement, in turn, applies to all routes.
- As with ACLs, an implicit **deny** is included at the end of the route map policy.
- You can use route maps to create policies based on the following:
  - IP address
  - End-system ID
  - Application
  - Protocol
  - Packet size

Configuring Route Maps

The route map syntax is composed of roughly three separate Cisco commands, depending on what the route map is accomplishing and what type of process is calling it. This discussion covers the following commands in detail as route maps are configured throughout this chapter:

- **route-map** commands
- **match** commands
- **set** commands

When configuring route maps, you can follow a basic five-step configuration process. Depending on the route map application, additional configuration may be needed, such as with BGP communities or PBR.

Step 1 (Optional) Configure any ACLs, AS_PATH list, or any other match criteria that the route map may be using on the **match** commands. This should be done first, so you do not call an empty ACL or AS-PATH list.

Step 2 Configure the route map instance. This is accomplished with the **route-map** *name* **permit | deny** *sequence_number* command. Be sure to leave room in between the sequence numbers for future updates or modifications. The route map instance with the lowest sequence number is executed first.

Step 3 Define the match criteria and configure the **match** statements that will be used in this single route map instance. You do this with the route map configuration **match** command. In the absence of any **match** commands, all packets or routes are matched.

Step 4 (Optional) Define the set criteria and configure the **set** statements that will be used in this single route map instance. You can do so with the route map configuration **set** command.

Step 5 (Optional) Configure any ACLs, AS_PATH list, or any other match criteria that the route map may be using on the **match** commands.

Step 6 Apply the route map. Once again, depending on the route map application, it can be applied in many ways. Some of the more common applications include route redistribution, PBR, and BGP.

With this configuration process in mind, we will discuss in more detail the three primary commands used to configure route maps.

route-map Commands

The complete syntax for the **route-map** command is as follows:

```
route-map route_map_name [permit|deny][sequence_number_1-65535]
```

The *route_map_name*, also called the *map tag*, is the text-based name of the route map. The name is unique and logically groups and defines the entire route map policy. This is the name that you use to call the route map during redistribution and other processes.

The **permit** and **deny** keywords are optional; the default keyword is **permit**. If the route map is called from a redistribution process, the keyword is set to **permit**, and the match criteria are met for the route map, the route(s) are redistributed. If the keyword were set to **deny**, in the same scenario the route(s) would be denied.

If the route map is called from a policy-routing statement, the match criteria are met for the route map, and the keyword is set to **permit**, the packet would be policy routed. Once again, **permit** is the default keyword. If the **deny** keyword is used, the packet is forwarded according to the normal route process.

The *sequence-number* indicates in what order the route map statements will be executed. When a route map is called, the route map with the lowest sequence number is executed first. If a match is not found in the route map with the lowest sequence number, the route map with the next highest sequence number is executed. This process repeats itself until a match is found or no more route map statements exist. If a match is found, execution for that individual packet or route stops, and the next packet or route begins the process again starting with the **route-map** statement with the lowest sequence number. The default sequence number is 10.

NOTE When creating route maps, leave room in between sequence numbers for future editing. Begin your first route map with a sequence of 10 or 100, depending on how big you expect the route map to be. By using increments of 10 or 100, you leave room for 65 to 650 route map instances. Starting at a higher sequence number and leaving space in between your sequence numbers will make editing your route maps easier. The maximum route map instance is 65,535.

match Commands

The **match** commands enable you to define the criteria of the route map. For instance, you can use the **match** command to call an ACL to compare routes against. The **match** statement could also match a route tag, a route type, or the length of a packet. BGP offers many exclusive **match** statements that are discussed in Chapters 4 and 5. Table 2-1 lists the **match** parameters available in Cisco IOS Software Release 12.2.

Table 2-1 **match** *Commands in Cisco IOS Software 12.2*

Command	What It Matches
as-path	BGP AS_PATH list
clns	CLNS* information
community	BGP community list
extcommunity	BGP/VPN** extended community list
interface	First-hop interface of a route
ip	IP-specific information
length	Packet length
metric	Route metric
route-type	Route type
tag	Route tag

* CLNS = Connectionless Network Service

** VPN = virtual private network

The **match ip address** command is by far the most commonly used of the **match** commands. The **match ip address** command enables you to call a standard, extended, or expanded-range ACL. You can use it during redistribution, with BGP, NAT, and during policy routing, as well as for other functions. The syntax for this **match** command is as follows:

```
match ip {address [access_list | prefix-list] | next-hop [access_list] | route-source
[access_list | prefix-list]}
```

In IP networks, this command enables you to match routes that have a network address matching one or more in the specified ACL or prefix list. You can use a standard, extended, or expanded-range ACL.

The **next-hop** keyword enables you to match routes that have a next-hop address matching one or more in the specified ACL. This is primarily used in BGP.

The **route-source** keyword enables you to match the advertising router's IP address of the route/network. You can use a standard, extended, or expanded-range ACL. For BGP, you may also use a prefix list.

NOTE When using the **match ip address** command in BGP, you can use route maps only to filter outbound updates. The use of a **match ip address** route map is not supported on inbound BGP updates.

The **next-hop** keyword is used primarily in BGP, but it can also be used when redistributing routes based on the next-hop IP address that correlates to the route. In this case, the router will examine the NEXT_HOP attribute for this comparison.

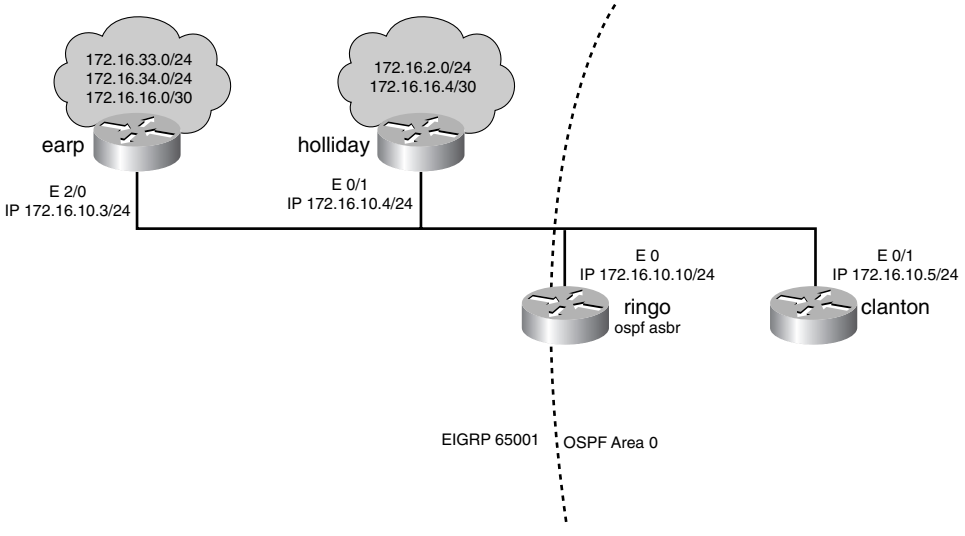
The **route-source** keyword enables you to match a router's advertising IP address. If you view the IP route table, and route 172.16.3.0/24 is advertised from the IP address of 172.16.2.1, for instance, the **route-source** keyword is used to match the advertising router's IP address of 172.16.2.1. In the next sections, these commands are applied to practical examples to show you how they function.

Practical Example: Matching the Route Source and IP Address

In this model, four routers on a common LAN segment are running two routing protocols. The routers earp and holliday are running EIGRP as the routing protocol, and the routers ringo and clanton are running OSPF. The router ringo is functioning as an OSPF autonomous system boundary router (ASBR) by redistributing between EIGRP and OSPF. The ringo router is receiving several routes from the earp and holliday routers, as depicted in Figure 2-1.

In this practical example, a route map is applied during the redistribution of EIGRP into OSPF on the ringo router. The route map named set_tag3 is called on the redistribution process for OSPF on the ringo router. The first route map instance, **route-map set_tag3 permit 100**, will perform a match on **IP route-source**. This statement will match only routes where the advertising IP address is found in ACL 5—in this case, the address 172.16.10.3. Not only will these routes be allowed for redistribution, but the tag of 3 will also be set.

Figure 2-1 *Route Map Practical Example: Matching the Route Source and IP Address*



NOTE When using a route map with OSPF, the advertising OSPF router ID becomes the route source. Use the OSPF router ID for the IP address of the route source when using the **route-source** keyword with OSPF networks.

Example 2-4 lists the forwarding/route table of the ringo router. Notice that routes 172.16.16.0/30, 172.16.33.0/24, and 172.16.34.0/24 are from the earp router, 172.16.10.3. The 172.16.2.0/24 and 172.16.16.4/30 routes are from the holliday router, 172.16.10.4.

Example 2-4 *Forwarding/Route Table of the ringo Router*

```
ringo# show ip route
<<<text omitted>>>>
C    192.168.10.0/24 is directly connected, Loopback20
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.33.0/24 [90/1812992] via 172.16.10.3, 00:07:13, Ethernet0
D       172.16.34.0/24 [90/1812992] via 172.16.10.3, 00:07:13, Ethernet0
D       172.16.16.4/30 [90/2195456] via 172.16.10.4, 00:07:13, Ethernet0
D       172.16.16.0/30 [90/1787392] via 172.16.10.3, 00:07:13, Ethernet0
C       172.16.10.0/24 is directly connected, Ethernet0
D       172.16.2.0/24 [90/307200] via 172.16.10.4, 00:07:14, Ethernet0
ringo#
```

Example 2-5 lists the configuration of the route map on the ringo router.

Example 2-5 *Configuration of the ringo Router*

```
!
interface Loopback20
 ip address 192.168.10.10 255.255.255.0
!
interface Ethernet0
 ip address 172.16.10.10 255.255.255.0
!
<<<text omitted>>>
!
router eigrp 65001
 network 172.16.0.0
 network 192.168.10.0
 no auto-summary
 no eigrp log-neighbor-changes
!
router ospf 7
 log-adjacency-changes
 redistribute eigrp 65001 subnets route-map set_tag3 ←Route-map called
 network 172.16.10.10 0.0.0.0 area 0
 default-metric 10
!
access-list 5 permit 172.16.10.3     ←Match route 172.16.10.3 only
access-list 50 permit any            ←Match all remaining routes
!
route-map set_tag3 permit 100
 match ip route-source 5             ←Match routes from 172.16.10.3 / ACL 5
 set tag 3                           ←set the tag to three
!
route-map set_tag3 permit 200        ←Second Route-map instance
 match ip address 50                 ←Call access-list 50 to match all routes
 set metric-type type-1              ←Set OSPF route type to External Type-1
 set tag 500                         ←Set the tag to 500 for these routes
```

In the preceding example, the second instance of the route map calls ACL 50. Access list 50 will allow the remaining routes to be redistributed and will set a tag of 500 and the metric-type to an OSPF type-1 external.

By viewing the OSPF database, you can clearly see the tags and how redistribution is working. Example 2-6 demonstrates the **show ip ospf database** command on the ringo router.

Example 2-6 **show ip ospf database** *Command*

```
ringo# show ip ospf database
            OSPF Router with ID (192.168.10.10) (Process ID 7)
                Router Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum Link count
172.16.10.5     172.16.10.5     1005        0x8000000B 0x18D8   1
```

Example 2-6 **show ip ospf database** *Command (Continued)*

```
192.168.10.10   192.168.10.10   1027        0x8000000A 0x7017   1
                Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
172.16.10.5     172.16.10.5     1005        0x8000000A 0x75DA
                Type-5 AS External Link States
Link ID         ADV Router      Age         Seq#       Checksum Tag
172.16.2.0      192.168.10.10   1027        0x80000009 0x10E0   500
172.16.16.0     192.168.10.10   1027        0x80000009 0xD285   3
172.16.16.4     192.168.10.10   1027        0x80000009 0x3BA6   500
172.16.33.0     192.168.10.10   1027        0x80000009 0x291B   3
172.16.34.0     192.168.10.10   1027        0x80000009 0x1E25   3
192.168.10.0    192.168.10.10   1027        0x80000009 0x8BB0   500
ringo#
```

Examining the route table of a downstream OSPF router, such as clanton, you can see the effects of the **set metric-type type-1** command. Notice in Example 2-6 that the 172.16.2.0/24, 192.168.10.0/24, and 172.16.16.4/30 routes are OSPF external type 1 routes. Normally, or by default, the routes would be OSPF external type 2 routes. For more information on the different link-state advertisement (LSA) types and their use, refer to *CCIE Practical Studies, Volume I*. You will learn more about the various **set** commands in the upcoming section. Example 2-7 lists the forwarding table of the clanton router.

Example 2-7 *Route Table of the clanton Router*

```
clanton# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
O E1 192.168.10.0/24 [110/20] via 172.16.10.10, 04:47:26, Ethernet0/0
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O E2    172.16.33.0/24 [110/10] via 172.16.10.10, 04:47:27, Ethernet0/0
O E2    172.16.34.0/24 [110/10] via 172.16.10.10, 04:47:27, Ethernet0/0
O E1    172.16.16.4/30 [110/20] via 172.16.10.10, 04:47:27, Ethernet0/0
O E2    172.16.16.0/30 [110/10] via 172.16.10.10, 04:47:27, Ethernet0/0
C       172.16.10.0/24 is directly connected, Ethernet0/0
O E1    172.16.2.0/24 [110/20] via 172.16.10.10, 04:47:27, Ethernet0/0
clanton#
```

BGP uses many specific **match** commands, as the next couple of examples show. BGP can use route maps to call an AS-Path rather than an ACL to control routing information. Table 2-2 lists the syntax for the **match as-path** command.

Table 2-2 **match as-path** *Command*

Command	Description
match as-path *[1-199]*	Used in BGP to match an autonomous system list. The valid path list is 1–199.

You can use this command in BGP to match the autonomous system path (AS_PATH) attribute.

Another BGP-specific **match** command is **match community**. You can use route maps to match and set the COMMUNITY attribute(s) in BGP.

The syntax for the **match community** command is as follows:

match [**community**|**extcommunity**|**exactmatch**]

The **community** keyword is used in BGP to call an IP community list. The valid range is 1 through 99 for a standard community list, and 100 through 199 for an expanded community list; alternatively, you can use **exact-match** to perform precise matching of communities.

You can use route maps to base the selection of the global address pool on the output interface as well as an ACL match for NAT. The **match interface** command is used in NAT applications. You can also use it to match routes whose next-hop address is an interface, such as a static route pointing at an interface. Table 2-3 shows the syntax for the **match interface** command.

Table 2-3 **match interface** *Command*

Command	Description
match interface *interface_name*	Used in route maps for NAT to match the output interface, or routes that have an interface as the next-hop address rather than an IP address.

Tags very effectively enable you to control and track routes during redistribution. Cisco routers enable the network engineer to mark certain routes with a numeric value. A tag value is an extra value that is transported along by the routing protocol. The tag value does not influence router forwarding decisions and has no intrinsic value to the routing protocol. The tag is used primarily during redistribution to *tag* or *flag* routes. After a route has been *tagged*, the tag value can be acted on during the redistribution process to control route redistribution. Tags are supported in RIPv2, OSPF, Integrated IS-IS, EIGRP, BGP, and CLNS. IGRP and RIPv1 do not support tags. To view tags, use the **show eigrp topology** *ip_address subnet_mask* and the **show ip ospf database** commands for EIGRP and OSPF, respectively. You can also view the tag value by using the extended **show ip route** command **show ip route** *ip_address*.Table 2-4 shows the syntax for the **match tag** command.

Table 2-4 **match tag** *Command*

Command	Description
match tag [*0-4294967295*]	Use the **match tag** command to match tag values in routing protocols such as RIPv2, IS-IS, OSPF, EIGRP, BGP, and CLNS.

You can also use route maps to match specific route types in Cisco IOS Software 12.0. For instance, you can match EIGRP external routes or OSPF external type 1 or type 2 routes. The **match route-type** keyword enables you to match the following route types:

- OSPF external type 1 (O E1) and type 2 routes (O E2), NSSA external type 1 (O N1) type 2 (O N2), intra-area routes (O), and interarea routes (O IA)
- EIGRP external routes (D EX)
- IS-IS level 1 routes (L1) and level 2 routes (L2)
- BGP external routes

The syntax for the **match route-type** command is as follows:

```
Match route-type {local|internal|external[type-1|type-2]|level-1|level2|nssa-external}
```

You can use the following keywords with the **match route-type** command:

External—External route (BGP, EIGRP, and OSPF type 1/2)

Internal—Internal route (including OSPF intra/interarea and EIGRP routes)

level-1—IS-IS level 1 route

level-2—IS-IS level 2 route

local—BGP locally generated route

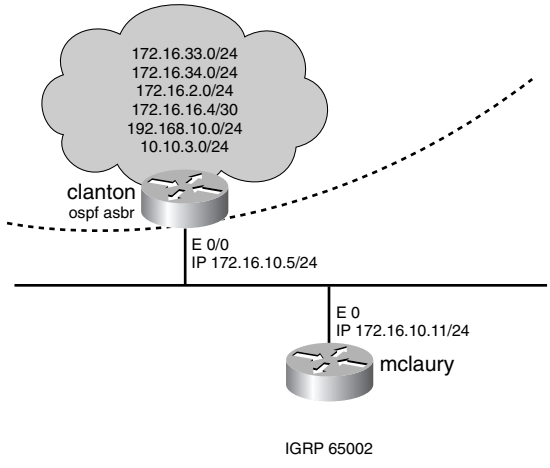
nssa-external—NSSA external route

Although you can use multiple **match** statements in a single line, you should use only one **match** criterion per line. This will make troubleshooting and modifying the route map easier.

Practical Example: Matching Tags

Manipulating the model from the preceding practical example, the following example has the router clanton running OSPF and IGRP as the routing protocols. The clanton router will call a route map on redistribution. This route map will redistribute routes with a tag of 3 and OSPF external type 1 (O E1) routes into IGRP. Figure 2-2 shows the new network model.

Figure 2-2 *Route Map Practical Example: Matching Tags*



Example 2-8 lists the route table of the clanton router, with the OSPF external type-1 routes highlighted. Example 2-9 lists the OSPF database on the clanton router, highlighting the routes that have a tag of 3.

Example 2-8 *Route Table of the clanton Router*

```
clanton# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
O E1 192.168.10.0/24 [110/20] via 172.16.10.10, 01:59:17, Ethernet0/0
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O E2    172.16.33.0/24 [110/10] via 172.16.10.10, 01:49:44, Ethernet0/0
O E2    172.16.34.0/24 [110/10] via 172.16.10.10, 01:49:44, Ethernet0/0
O E2    172.16.16.0/30 [110/10] via 172.16.10.10, 01:49:44, Ethernet0/0
C       172.16.10.0/24 is directly connected, Ethernet0/0
O E2    172.16.2.0/24 [110/10] via 172.16.10.10, 01:49:44, Ethernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
O E1    10.10.3.0 [110/20] via 172.16.10.10, 01:59:18, Ethernet0/0
clanton#
```

Example 2-9 *OSPF Database of the clanton Router*

```
clanton# show ip ospf database
       OSPF Router with ID (172.16.10.5) (Process ID 7)

                Router Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum Link count
172.16.10.5     172.16.10.5     557         0x80000006 0x22D3   1
192.168.10.10   192.168.10.10   1642        0x80000005 0x7A12   1
                Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
172.16.10.5     172.16.10.5     557         0x80000005 0x7FD5
                Type-5 AS External Link States
Link ID         ADV Router      Age         Seq#       Checksum Tag
10.10.3.0       192.168.10.10   1642        0x80000004 0x9904   500
172.16.2.0      192.168.10.10   1133        0x80000005 0x87DF   3
172.16.16.4     192.168.10.10   1642        0x80000004 0x45A1   500
172.16.33.0     192.168.10.10   1133        0x80000005 0x3117   3
172.16.34.0     192.168.10.10   1133        0x80000005 0x2621   3
192.168.10.0    192.168.10.10   1643        0x80000004 0x95AB   500
clanton#
```

To control redistribution between OSPF and IGRP, use a route map on the redistribution process. The route map to accomplish must have two route map instances. The first route map instance will match all routes in OSPF that have a tag value of 3. The second route map instance will match OSPF external type 1 routes. Example 2-10 lists the significant portions of the configuration on the clanton router.

Example 2-10 *Route Map Configuration on the clanton Router*

```
hostname clanton
!
router ospf 7
 network 172.16.10.5 0.0.0.0 area 0
!
router igrp 65002
 redistribute ospf 7 route-map match_me    ←Redistribute OSPF and call the route-map
 network 172.16.0.0
 default-metric 10000 100 254 1 1500
!
route-map match_match_me permit 10
 match tag 3                               ←Match routes with a tag 3
!
route-map match_match_me permit 20
 match route-type external type-1          ←Match OSPF external type-1 routes
```

To verify redistribution and that the route maps worked properly, view the route table of the mclaury router. Example 2-11 lists the route table of the mclaury router. Notice that routes with

a tag value of 3 are present: 172.16.2.0/24, 172.16.33.0/24, and 172.16.34.0/24. Also, notice that the OSPF external type 1 routes are present: 192.168.10.0/24 and 10.0.0.0/8 as summarized subnets.

Example 2-11 *Route Table of the mclaury Router*

```
mclaury# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, 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 not set
I    192.168.10.0/24 [100/1200] via 172.16.10.5, 00:00:50, Ethernet0
     172.16.0.0/24 is subnetted, 4 subnets
I       172.16.33.0 [100/1200] via 172.16.10.5, 00:00:50, Ethernet0
I       172.16.34.0 [100/1200] via 172.16.10.5, 00:00:50, Ethernet0
C       172.16.10.0 is directly connected, Ethernet0
I       172.16.2.0 [100/1200] via 172.16.10.5, 00:00:50, Ethernet0
I    10.0.0.0/8 [100/1200] via 172.16.10.5, 00:00:50, Ethernet0
mclaury#
```

You can also use route maps to match a route's metric. This is the metric for the route as it appears in the route/forwarding table. If an OSPF route has an associated metric of 20, for instance, **match metric 20** is used to match this route. Table 2-5 lists the syntax used with the **match metric** command.

Table 2-5 **match metric** *Command*

Command	Description
match metric [**0-4294967295**]	Enter the metric value as it appears in the route/forwarding table of the router.

Using Figure 2-1 as a guide, Example 2-12 lists the route table of the clanton router followed by the route map configuration used to match the OSPF routes with a metric of 20. This example redistributes OSPF routes into EIGRP routes that have a metric of 20.

Example 2-12 *Demonstration of the* **match metric** *Route Map*

```
clanton# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
```

continues

Example 2-12 *Demonstration of the* **match metric** *Route Map (Continued)*

```
Gateway of last resort is not set
O E1 192.168.10.0/24 [110/20] via 172.16.10.10, 00:19:58, Ethernet0/0
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O E2    172.16.33.0/24 [110/10] via 172.16.10.10, 00:19:59, Ethernet0/0
O E2    172.16.34.0/24 [110/10] via 172.16.10.10, 00:19:59, Ethernet0/0
O E1    172.16.16.4/30 [110/20] via 172.16.10.10, 00:19:59, Ethernet0/0
O E2    172.16.16.0/30 [110/10] via 172.16.10.10, 00:19:59, Ethernet0/0
C       172.16.10.0/24 is directly connected, Ethernet0/0
O E2    172.16.2.0/24 [110/10] via 172.16.10.10, 00:19:59, Ethernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
O E1    10.10.3.0 [110/20] via 172.16.10.10, 00:19:59, Ethernet0/0

hostname clanton
!
<<<text omitted>>>
!
router ospf 7
 network 172.16.10.5 0.0.0.0 area 0
!
router eigrp 65002
 redistribute ospf 7 route-map match_metric_20
 network 172.16.0.0
 default-metric 10000 100 254 1 1500
!
ip classless
!
route-map match_metric_20 permit 10
 match metric 20
!
```

In the preceding example, the routes 10.10.3.0/24, 172.16.16.4/30, and 192.168.10.0/24 were redistributed into EIGRP.

The **match clns address** command is used in ISO CLNS routing much in the same way that it is used in IP routing. The **match clns address** command calls a CLNS address list and compares the address being testing against it. The **next-hop** and **route-source** keywords are used to call an OSI filter set during policy routing. Use the CLNS commands in the same manner as their IP counterparts. The syntax of the **match clns** command is as follows:

```
match clns {address [name]|next-hop [filter set]|route-source [filter set]}
```

Use the **match clns address** command to match routes that have a network address matching one or more in the specified OSI filter set.

The **next-hop** keyword is used to match routes that have a next-hop address matching one or more in the specified OSI filter set.

The **route-source** keyword is used to match routes that have been advertised by routers matching one or more in the specified OSI filter set.

The last **match** command discussed here is the **match length** command. This **match** statement is used primarily in policy routing when ACLs are insufficient for proper traffic distribution. The **match length** command enables you to match the Layer 3 packet length in bytes, including headers and trailers. You can use a route map such as this to send little interactive packets, such as Telnet traffic, one way, and large bulk-data transfers, such as a large FTP transfer, another way. Table 2-6 lists the syntax for the **match length** command.

Table 2-6 **match length** *Command*

Command	Description
Match length [*min_packet_length_0-2147483647*] [*max_packet_length_0-2147483647*]	Used to match the Layer 3 packet length in bytes with all associated headers and trailers included. You must enter the minimum and maximum packet length.

For an example of the **match length** command, see the later section "Configuring Policy-Based Routing (PBR)."

set Commands

The **set** commands are executed after a successful match has been made in the route map instance. The **set** command is optional and may be omitted. If you are using route maps on redistribution, or just to filter networks, for instance, there is no need to use a **set** command unless you want to *tag* or further influence the route. If no **match** statements are present in the route map instance, all **set** commands are executed for all routes. You may also use multiple **set** commands in each route map instance. The **set** commands discussed here are supported in Cisco IOS Software Release 12.2 and are listed in Table 2-7. The **set** commands have been divided into three categories: BGP-specific **set** commands, routing protocol/redistribution-specific **set** commands, and policy-routing specific **set** commands. The policy-routing specific **set** commands are covered in the upcoming section "Configuring Policy-Based Routing (PBR)."

Table 2-7 **set** *Commands*

set Command	Description
BGP-specific* set *commands	
as-path	Prepend string for a BGP AS_PATH attribute
community|extcommunity	Set BGP COMMUNITY attributes
comm-list	BGP community list for deletion
dampening	Set BGP route flap dampening parameters
local-preference	Set BGP LOCAL_PREF path attribute

continues

Table 2-7 **set** *Commands (Continued)*

set Command	Description
BGP-specific* set *commands (Continued)	
origin	Set BGP origin code
weight	Set BGP weight
Routing protocol/redistribution-specific* set *commands	
metric	Set metric value for destination routing protocol
metric-type	Type of metric for destination routing protocol
tag|automatic-tag	Tag value for destination routing protocol
Policy-routing specific set commands	
default	Set default routing information
interface	Set the Output interface, used in point-to-point links
ip	IP-specific information

BGP-Specific set Commands

The first **set** commands covered here are the ones related to BGP. This section discusses the syntax of the various **set** commands for BGP and their basic application. For more specific and detailed information on the application of the BGP-specific **set** commands, see Chapter 8, "Introduction to BGP-4 Configuration," and Chapter 9, "Advanced BGP Configuration."

The **set as-path** command is used in BGP to prepend one or more autonomous systems to the well-known mandatory transitive AS_PATH attribute. In BGP, this can be used to influence routing decisions. BGP views routes that have one or more autonomous systems prepended to the current AS_PATH attribute as less desirable, which can prove useful in a multihomed BGP network.

CAUTION The purpose of the **prepend** command is essentially to make the AS_PATH longer—thereby forming a less desirable path—not to completely change it. When using the **set as-path prepend** command in production environments, always use the same ASN that the route is from. If a different ASN is used, and that autonomous system is encountered by the advertised route, the receiving autonomous system/router will not accept the route. Modifying the AS_PATH by prepending a different autonomous system directly affects the inherent loop prevention provided by the AS_PATH attribute. Some Cisco IOS Software levels will not even enable you to enter an AS_PATH different from your own. For educational purposes, some of the examples in this text show the prepending of different autonomous systems; this is done to highlight the placement of the prepended autonomous system only.

Note an important difference in how the **prepend** command works with inbound and outbound route maps. When the **prepend** command is used on outbound route maps, the prepended autonomous system is added after the advertising router's autonomous system. This is because the prepended autonomous system will be in place before the route update is sent. When the update is sent, the advertising router's autonomous system is the first one on the list. If you prepend AS 10 10 to an outbound route map, for instance, and your router's autonomous system is 5, the receiving router/neighbor will have an AS_PATH of 5 10 10.

If you apply the **prepend** command on an inbound route map, the autonomous system that is prepended will actually precede the originating AS_PATH. This is because the autonomous system prepend is happening after the route has been received from its neighbor. If you prepend AS 10 10 on an inbound route map, and the router/neighbor you are receiving the route from has the AS_PATH of 5 500, for example, the AS_PATH to that route will be 10 10 5 500. The syntax for the **set as-path** command is as follows:

```
set as-path {prepend [as_path1|as_path2|as_path3]|[tag]}
```

Use the **set as-path** command in BGP networks to modify the AS_PATH attribute, by prepending one or more autonomous systems to it. You can use this command on inbound and outbound route maps.

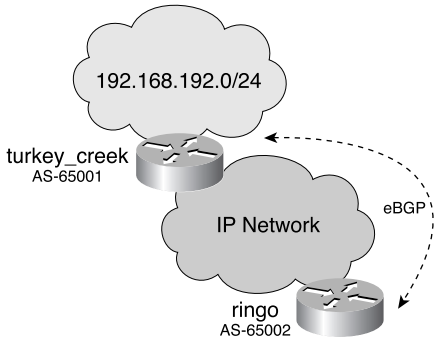
The **tag** keyword is used in BGP to recover the AS_PATH information from the tags of Interior Gateway Protocol (IGP)-redistributed routes.

The **set as-path tag** command is used in BGP when doing redistribution to preserve a consistent and correct AS_PATH across an IGP. The Cisco BGP implementation automatically conveys AS_PATH in the form of a tag when redistributing BGP into an IGP. When redistributing IGP routes into BGP, however, AS_PATH information is lost. To recover the AS_PATH information from the tag of a redistributed IGP, use the **set as-path tag** command.

Practical Example: Setting the AS_PATH

The network model shown in Figure 2-3 has two routers running BGP between them. The turkey_creek router is in autonomous system 65001, and the ringo router is in autonomous system 65002. The router turkey_creek will advertise the network 192.168.192.0/24 via BGP. In this example, a route map will be used to prepend the AS_PATH with autonomous system 65001 2001 on outbound updates from the turkey_creek router.

Figure 2-3 *Route Map Practical Example: Setting the AS_PATH*



Example 2-13 lists the configuration to manipulate the AS_PATH attribute on the turkey_creek router.

Example 2-13 *BGP Configuration of the turkey_creek Router*

```
hostname turkey_creek
!
<<<text omitted>>>
!
router bgp 65001
 no synchronization
 network 192.168.192.0
 neighbor 172.16.100.10 remote-as 65002
 neighbor 172.16.100.10 ebgp-multihop 10
 neighbor 172.16.100.10 update-source Loopback20
 neighbor 172.16.100.10 route-map set_as out     ←Call route-map "set_as" for
                                                 outbound updates
!
route-map set_as permit 10
 set as-path prepend 65001 2001                  ←prepend AS-PATH with 65001 2001
!
```

You might be tempted to think that the AS_PATH for route 192.168.192.0/24 would read 65001 2001 65001; after all, the command says "prepend." Because this is an outbound route map, however, the prepended autonomous system will occur before the advertisement is sent. Therefore, the "prepended" AS_PATH will appear after the originating autonomous system to the downstream router. The AS_PATH on the downstream router, the ringo router, will read 65001 65001 2001. Example 2-14 demonstrates this by listing the output of the **show ip bgp** command on the ringo router.

Example 2-14 **show ip bgp** *Command on the ringo Router*

```
ringo# show ip bgp 192.168.192.0
BGP routing table entry for 192.168.192.0/24, version 4
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  65001 65001 2001
    172.16.200.10 (metric 1915392) from 172.16.200.10 (192.168.192.7)
      Origin IGP, metric 0, localpref 100, valid, external, best
ringo#
```

Example 2-15 applies the route map to inbound updates on the ringo router. The route map will append the AS_PATH 2001 65002 65001 to the routes from the turkey_creek router. Because this is an inbound route map, the final AS_PATH on the ringo router will read 2001 65002 65001 65001. On inbound route maps, the prepend functions like its name implies. Example 2-15 lists the relevant portions of the configuration of the ringo router, followed by the **show ip bgp** command.

Example 2-15 *Configuration of the ringo Router, and the* ***show ip bgp*** *Command*

```
Hostname ringo
!
<<<text omitted>>>
!
router bgp 65002
 no synchronization
 bgp log-neighbor-changes
 neighbor 172.16.200.10 remote-as 65001
 neighbor 172.16.200.10 ebgp-multihop 10
 neighbor 172.16.200.10 update-source Loopback20
 neighbor 172.16.200.10 route-map modify_as in      ←Route-map "modify_as" is called
!
route-map modify_as permit 10
 set as-path prepend 2001 65002 65001               ←Prepended AS
!
_____________________________________________________________
ringo# show ip bgp 192.168.192.0
BGP routing table entry for 192.168.192.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  2001 65002 65001 65001
    172.16.200.10 (metric 1915392) from 172.16.200.10 (192.168.192.7)
      Origin IGP, metric 0, localpref 100, valid, external, best
ringo#
```

The **set community** command is used in BGP to set various community attributes. As discussed in the later chapters on BGP, communities can be a powerful and efficient way to apply policies to a group of routes. The community is an optional transitive route attribute and communicated

among BGP peers. The **set community** command enables you to form community membership. After routes become members of a community, they can be assigned policies, such as "do not export this route to any E-BGP neighbors or advertise this route the Internet community." To send the community attribute in BGP, the **neighbor a.b.c.d send-community** command must be used.

The syntax for the **set community** command in Cisco IOS Software Release 12.2 is as follows:

```
set community {community-number_1-4294967200|AA|NN|no-export|no-advertise |internet
|local-AS [additive]}|none
```

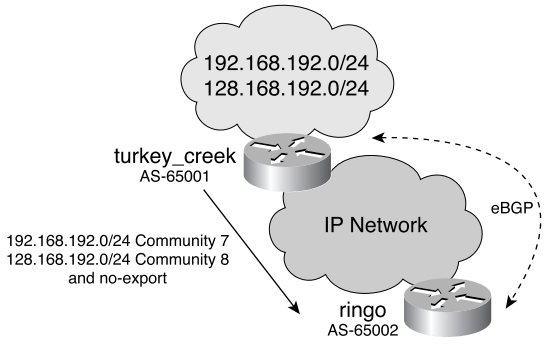
Use the **set community** command to designate or form communities from routes and to apply specific policies to those routes. The valid parameters and values are as follows:

- *Community number*—A valid number from 1 to 4,294,967,200; the routes will be designated to this community number.
- *AA:NN*—This format can also be used to designate communities. The *AA* is a 16-bit ASN between 1 and 65,535. *NN* is an arbitrary 16-bit number between 1 and 65,440.
- **Internet**—The Internet community. Advertise these routes to the Internet community and any router belonging to it.
- **no-export**—Do not advertise these routes to an E-BGP peer. Routes with this community are sent to peers in other subautonomous systems within a confederation.
- **local-as**—Do not advertise these routes to peers outside the local autonomous system. These routes will not be advertised to other autonomous systems or subautonomous systems if confederations are in place.
- **no-advertise**—Do not advertise these routes to any peer (internal or external). Used for I-BGP peers.
- **Additive**—(Optional) Adds the community to the already existing communities.
- **None**—Removes the COMMUNITY attribute from the prefixes that pass the route map.

Practical Example: Setting BGP Community Attributes

Consider the same network model from the preceding example, with the turkey_creek and the ringo routers running BGP between them. (See Figure 2-4.) The turkey_creek router is in AS 65001, and the ringo router is in AS 65002. The router turkey_creek will advertise the network 192.168.192.0/24 via BGP and place it in community 7. The turkey_creek router will also advertise another route, 128.168.192.0/24. The turkey_creek router will put this route in community 8 and set the COMMUNITY attribute to **no-export**. The **no-export** COMMUNITY attribute will instruct the ringo router not to advertise this route to E-BGP neighbors.

Figure 2-4 *Route Map Practical Example: Setting Communities*



Example 2-16 lists the route map used to accomplish this.

Example 2-16 *Route Map for Communities on turkey_creek*

```
Hostname turkey_creek
!
<<<text omitted>>>
!
router bgp 65001
 no synchronization
 network 128.168.192.0 mask 255.255.255.0
 network 192.168.192.0
 neighbor 172.16.100.10 remote-as 65002
 neighbor 172.16.100.10 ebgp-multihop 10
 neighbor 172.16.100.10 update-source Loopback20
 neighbor 172.16.100.10 send-community        ←send-community must be enabled
 neighbor 172.16.100.10 route-map set_communities out ←route-map "set_communities"
called
!
<<<text omitted>>>
!
access-list 10 permit 192.168.192.0 0.0.0.255     ←allow network 192.168.192.0/24 only
access-list 11 permit 128.168.192.0 0.0.0.255     ←allow network 128.168.192.0/24 only
!
route-map set_communities permit 100
 match ip address 10                       ←Match ip access-list 10 or 192.168.192.0/24
 set community 7                             ←set the community to 7
!
route-map set_communities permit 200
 match ip address 11                       ←Match ip access-list 11 or 128.168.192.0/24
 set community 8 no-export                  ←set the community to 8 and don't export to
                                             future E-BGP peers
```

By observing the routes on the ringo router, you can see that route 192.168.192.0/24 is in community 7. Route 128.168.192.0/24 is in community 8 with the **no-export** option set. Example 2-17 lists the output of the **show ip bgp** command on the ringo router.

Example 2-17 *Routes with Communities Set on the ring Router*

```
ringo# show ip bgp 192.168.192.0
BGP routing table entry for 192.168.192.0/24, version 3
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  65001
    172.16.200.10 (metric 1915392) from 172.16.200.10 (192.168.192.7)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 7
ringo#
ringo#show ip bgp 128.168.192.0
BGP routing table entry for 128.168.192.0/24, version 2
Paths: (1 available, best #1, table Default-IP-Routing-Table, not advertised to
EBGP peer)
  Not advertised to any peer
  65001
    172.16.200.10 (metric 1915392) from 172.16.200.10 (192.168.192.7)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: 8 no-export
ringo#
```

Use the **set comm-list delete** command to remove the COMMUNITY attribute of an inbound or outbound route update. The syntax is as follows:

```
set comm-list {[standard | extended community list]} delete
```

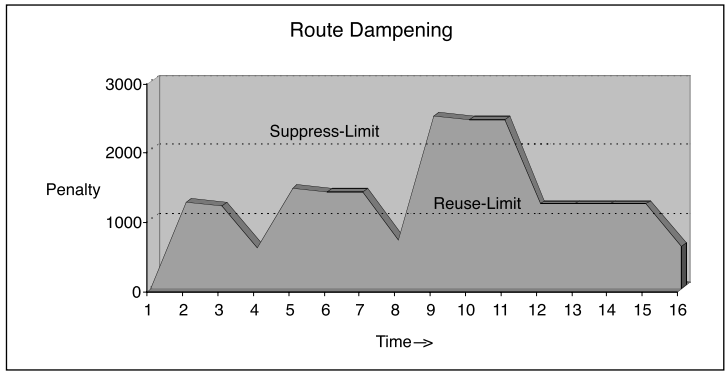
For more examples of the various **set communities** commands and how they function in BGP, see Chapters 7 through 9 on configuring BGP.

Another BGP-specific feature that you can set is dampening. Because of the long time it takes BGP networks to converge, an unstable route, or "route flapping," can have significant and detrimental impacts on large BGP networks. If a route goes down, a WITHDRAWN message is sent via BGP requesting all peers to remove that route from their tables. An instable route in your autonomous system will cause constant sending and withdrawing of messages to other autonomous systems. This effect multiplied by the hundreds or thousands of routes that may be in an autonomous system can negatively affect BGP.

Dampening allows routers to categorize routes as either well behaved or ill behaved. Obviously, a *well-behaved* route should be very stable over an extended period of time. On the other end of the spectrum, an *ill-behaved* route could be a route that is unstable, or flapping. When route dampening is enabled in BGP, with the BGP router command **bgp dampening**, the router will start a history file on how many times each route flaps. The route dampening feature will start to assign a penalty to a route each time it flaps. The penalties start to accumulate for each route, and when the penalty is greater than an arbitrary number called the *suppress value*, the route

will no longer be advertised. The route will remain suppressed until either the penalty falls below the reuse-limit or the max_suppress timer is exceeded. The penalty for a route can be decreased over time. The half-life timer is a timer, expressed in minutes, that must elapse before the penalty will be reduced by one-half. If the route remains stable, over time the penalty for that route will decrease. When the penalty is below another arbitrary timer called the reuse-limit, the route will be unsuppressed and advertised once again. Figure 2-5 illustrates the time and penalty relationship in route dampening.

Figure 2-5 *Route Dampening Timer Relationship*



Note that this type of route map is called from the BGP router command **bgp dampening** [**route-map** *route-map_name*]. A route map called on the **neighbor** statement will not work for route dampening.

The **set dampening** command in Cisco IOS Software Release 12.2 is as follows:

```
set dampening {half-life_1-45 reuse_1-20000 suppress_1-20000 max_suppress_time_1-255}
```

Use the **set dampening** command to influence how the router will react when it encounters unstable routes. The **half-life** parameter represents a time (in minutes) that must pass with a route being stable, after which the penalty value is reduced by half. The default is 15 minutes, and the valid range is 1 to 45 minutes.

The **reuse** parameter enables you to mark a point, or a reuse, point that allows the route to be advertised. When the penalty value falls below the reuse point, the route is unsuppressed and re-advertised. The default value is 750, and the valid ranges are 1 to 20,000.

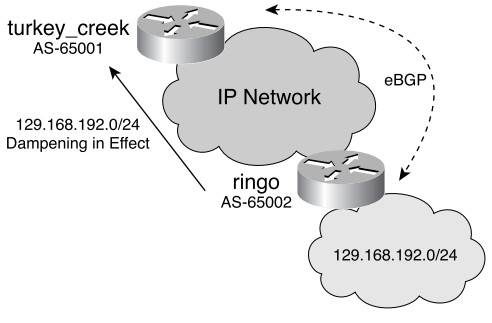
When the penalty exceeds the **suppress** parameter, the route is suppressed and no longer advertised. The valid range is from 1 to 20,000; the default is 2000.

The **max_suppress_time** is a value expressed in minutes that specifies how long a route should be suppressed by the dampening feature. The default value of this is 4 times the half-life timer, or 60 minutes. The valid range is from 1 to 255 minutes.

When a prefix is withdrawn, BGP considers the withdrawn prefix as a flap and increases the penalty by 1000. When BGP receives an attribute change, the penalty is increased by 500.

In Figure 2-6, the router ringo is advertising 129.168.192.0/24 to the turkey_creek router via BGP.

Figure 2-6 *Route Dampening*



The turkey_creek router has route dampening enabled, with a route map to apply the dampening to route 129.168.192.0/24. Use the **show ip bgp dampened-paths** command and the **show ip bgp a.b.c.d** command to view whether route dampening has occurred and to view what the current penalty count. Note that information related to the dampening does not appear until the route has actually flapped. Example 2-18 shows the penalty and dampening occurring on the turkey_creek router for route 129.168.192.0/24.

Example 2-18 *Verifying Dampening*

```
turkey_creek# show ip bgp dampened-paths
BGP table version is 9, local router ID is 192.168.192.7
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          From             Reuse    Path
*d 129.168.192.0/24 172.16.100.10    00:38:00 65002 i
turkey_creek#
turkey_creek# show ip bgp
BGP table version is 9, local router ID is 192.168.192.7
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
*> 128.168.192.0/24 0.0.0.0                  0         32768 i
*d 129.168.192.0/24 172.16.100.10            0             0 65002 i
*> 192.168.192.0    0.0.0.0                  0         32768 i
turkey_creek#
turkey_creek# show ip bgp 129.168.192.0
BGP routing table entry for 129.168.192.0/24, version 9
```

Example 2-18 *Verifying Dampening (Continued)*

```
Paths: (1 available, no best path)
  Not advertised to any peer
  65002, (suppressed due to dampening)
    172.16.100.10 (metric 2323456) from 172.16.100.10 (172.16.100.10)
      Origin IGP, metric 0, localpref 100, valid, external, ref 2
      Dampinfo: penalty 3717, flapped 4 times in 00:04:36, reuse in 00:37:50
turkey_creek#
```

Example 2-19 lists the BGP configuration for the preceding example and the associated route maps of the turkey_creek router.

Example 2-19 *Configuration of the turkey_creek Router*

```
hostname turkey_creek
!
<<<text omitted>>>
!
router bgp 65001
 no synchronization
 bgp dampening route-map set_dampening     ←Dampening enabled with route-map
 network 128.168.192.0 mask 255.255.255.0
 network 192.168.192.0
 neighbor 172.16.100.10 remote-as 65002
 neighbor 172.16.100.10 ebgp-multihop 10
 neighbor 172.16.100.10 update-source Loopback20
!
access-list 11 permit 129.168.192.0 0.0.0.255
!
route-map set_dampening permit 100
 match ip address 11                        ←Match network 129.168.192.0/24
 set dampening 20 1000 2000 80              ←Set dampening parameters
```

For more information and examples on route dampening, see BGP Chapters 7 through 9.

You can also use route maps in BGP to set the well-known discretionary LOCAL_PREF attribute. The LOCAL_PREF attribute is a numeric value ranging from 0 to 4,294,967,295, where the higher the value, the more preferred the route is. The default LOCAL_PREF value is 100. Table 2-8 lists the syntax used in setting the LOCAL_PREF attribute.

Table 2-8 **set local-preference** *Command in Cisco IOS Software Release 12.2*

Command	Description
set local-preference {0-4294967295}	Use the **set local-preference** command to set the LOCAL_PREF of a route. The valid range is from 0 to 4,294,967,295. The default value is 100.

Another BGP attribute that you can set with route maps is the well known mandatory transitive ORIGIN attribute. The ORIGIN attribute is a well-known mandatory attribute. The ORIGIN attribute, as the name states, specifies the origin of the route with respect to the autonomous system that originated it. BGP supports three different types of origin:

- **IGP(i)**—The network layer reachability information (NLRI) is internal to the originating autonomous system. This is a remote IGP system. The route originates from the network command.
- **EGP(e)**—The NLRI is learned via the EGP. This is a local EGP system. The route is redistributed from EGP.
- **Incomplete(?)**—The NLRI is learned from some other means. The route is redistributed from an IGP or static.

Table 2-9 lists the syntax used in setting the origin.

Table 2-9 **set origin** *Command in Cisco IOS Software Release 12.2*

Command	Description
set origin {**igp** | **egp** [*as_number*] | **incomplete**}	Use the **set origin** command to set the ORIGIN attribute of a route/routes. The valid origin types are IGP, EGP, and incomplete.

The final BGP-specific set command discussed here is the **set weight** command. The WEIGHT attribute is a Cisco proprietary feature used to measure a route's preference. The WEIGHT attribute is local to the router and does not get exchanged between routers; therefore it is only effective on inbound route maps. Use the WEIGHT attribute to influence routes from multiple service providers to a central location. Like LOCAL_PREF, assigning a higher weight to a route makes that route more preferred. The WEIGHT attribute also has the highest precedence of any BGP attribute. For more information on BGP, see Chapters 7 through 9. Table 2-10 lists the syntax used in setting the WEIGHT attribute.

Table 2-10 **set weight** *Command in Cisco IOS Software Release 12.2*

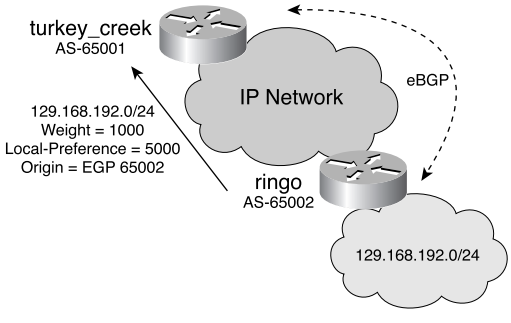
Command	Description
set weight {*0-65535*]	Use the **set weight** command to set the weight of a route/routes. The valid weight range is from 0 to 65,535, and the default weight of a route is 32,768.

Practical Example: Configuring BGP Attributes

This practical example uses the same network model as in the previous examples and sets the BGP attributes of LOCAL_PREF, WEIGHT, and ORIGIN. Figure 2-7 is the same network shown earlier. This example calls an inbound route map on the turkey_creek router. The route map set_attributes will set the following attributes: WEIGHT to 1000, LOCAL_PREF to 5000, and ORIGIN to be EGP from autonomous system 65002. In this example, the setting **local-**

preference is for education purposes only. Normally, **local-preference** would not be used or effective on E-BGP peers.

Figure 2-7 *Configuring BGP Attributes*



Example 2-20 lists the BGP and route map configuration to accomplish this on the turkey_creek router.

Example 2-20 *BGP Attribute Configuration*

```
hostname turkey_creek
!
<<<text omitted>>>
!
router bgp 65001
 no synchronization
 network 128.168.192.0 mask 255.255.255.0
 network 192.168.192.0
 neighbor 172.16.100.10 remote-as 65002
 neighbor 172.16.100.10 ebgp-multihop 10
 neighbor 172.16.100.10 update-source Loopback20
 neighbor 172.16.100.10 route-map set_attributes in ←call route-map "set_attributes"
!
route-map set_attributes permit 100
 set local-preference 5000      ←Set local-preference to 5000
 set weight 1000                ←Set weight to 1000
 set origin egp 65002           ←Set the ORIGIN to EGP in AS 65002
!                               ←*note with no match parameter all routes are
                                matched from the neighbor 172.16.100.10
```

To verify the effectiveness of the route map, use the **show ip bgp** command, as demonstrated in Example 2-21.

Example 2-21 *Verifying the Attributes*

```
turkey_creek# show ip bgp
BGP table version is 4, local router ID is 192.168.192.7
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
*> 128.168.192.0/24 0.0.0.0                  0         32768 i
*> 129.168.192.0/24 172.16.100.10            0   5000   1000 65002 e
*> 192.168.192.0    0.0.0.0                  0         32768 i
turkey_creek#
turkey_creek#
turkey_creek# show ip bgp 129.168.192.0
BGP routing table entry for 129.168.192.0/24, version 2
Paths: (1 available, best #1)
  Not advertised to any peer
  65002
    172.16.100.10 (metric 2323456) from 172.16.100.10 (172.16.100.10)
      Origin EGP, metric 0, localpref 5000, weight 1000, valid, external, best,
ref 2
turkey_creek#
```

Configuring Routing Protocol/Redistribution-Specific **set** Commands

The **set** commands covered next relate primarily to IGP routing protocols and are used mostly during route redistribution. The **set metric**, **set metric-type** and **set tag** commands can all be used to change the metric or the tag of a route during redistribution. As mentioned previously, the metrics and tags can also be matched and used for further route control during redistribution.

The most common use of the **set metric** command is to set the metric of the route for the destination routing protocol. If you are redistributing EIGRP routes into OSPF, for example, you can use a route map in conjunction with the **set metric** command to set the new OSPF metric. If you are redistributing into IGRP or EIGRP, the metric value you enter is the composite metric only. This differs slightly from setting the default metric or the metric on redistribution without a route map, where you would set all five submetrics. Another use of the **set metric** command is to set the BGP optional nontransitive MULTI_EXIT_DISC (MED) attribute. The syntax for the **set metric** command in Cisco IOS Software Release 12.2 is as follows:

set metric {[**-/+<***0-4294967295***>**]|*1-4294967295*]

The **+** and – keywords enable you to increase or decrease the current metric. To increase the metric by 10, for example, the command would be **set metric +10**. To set just the composite metric for EIGRP, the command is **set metric 4295**. For more information on IGP routing protocol metrics, refer to *CCIE Practical Studies, Volume I*. You can find more information on the BGP MED attribute in Chapters 7 through 9 of this book.

The **set metric-type** command is rather limited. It is used primarily in BGP, OSPF, and IS-IS. You can use it to set IS-IS external and internal metrics and OSPF type 1 and type 2 external metrics. The **set metric-type** command can also be used in BGP to use the IGP metric as the MED for BGP. The syntax for the **set metric-type** command in Cisco IOS Software Release 12.2 is as follows:

```
set metric-type [internal|external|type-1|type-2]
```

- **external**—IS-IS external metric.
- **internal**—Use the metric of the IGP as the MED for BGP. Also used for setting IS-IS internal metric.
- **type-1**—Use to match the OSPF type 1 metric.
- **type-2**—Use to match the OSPF external type 2 metric.

The final **set** command discussed in this section is the **set tag** command. The **set tag** command enables you to set the administrative tag of route. For IGPs, the tag value is usually set with a route map and the **set tag** command. In BGP, when you redistribute BGP into an IGP, the ASN of BGP is automatically put into the tag value. BGP does this to preserve the AS_PATH attribute across an IGP domain. For IGPs, the tag is an administrative value that certain routing protocols carry within the routing update. The tag value has no impact on routing decisions. Instead, it is used to mark routes or flag routes or to track the AS_PATH for BGP. The tag value may also be acted upon during a redistribution process. When the **automatic-tag** command is used with the BGP **table-map** command, the tag value includes the ASN and the origin. The syntax used to manipulate the tag value in Cisco IOS Software Release 12.2 is as follows:

```
set {tag [0-4294967295]|automatic-tag}
```

Use the **set tag** *value* command to set the tag value. Use the **set automatic-tag** command when redistributing an IGP into BGP to recover the tag value as an AS_PATH attribute.

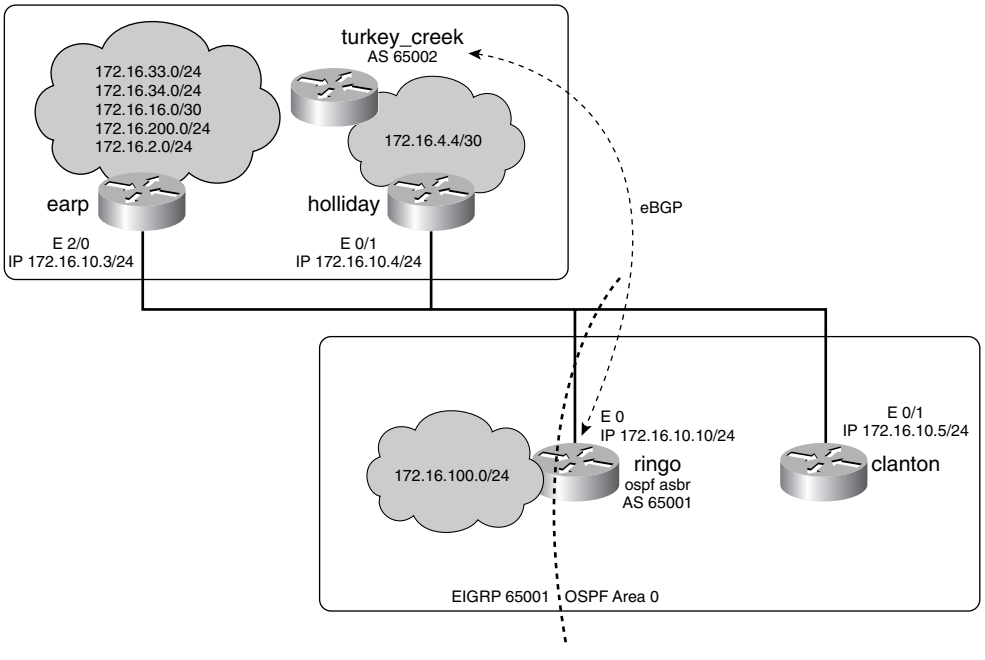
NOTE You can also use the tag value strictly for documentation purposes in an internetwork. If you have an OSPF domain, where RIP routes and EIGRP routes are redistributed, for example, you may want to tag the routes from EIGRP with a value of 100 and tag the routes from IGRP with a value of 110. When the OSPF database is viewed, it will be easy to determine the origin of specific routes. This can prove a handy documentation tool for troubleshooting route redistribution.

Tags are supported in RIPv2, OSPF, Integrated IS-IS, EIGRP, BGP, and CLNS. IGRP and RIPv1 do not support tags. To view tags, use the **show ip eigrp topology** *ip_address subnet_mask* command and the **show ip ospf database** command for EIGRP and OSPF, respectively. You can also view the tag value in other routing protocols by using the extended **show ip route** command, **show ip route** *ip_address*.

Practical Example: Setting Route Tags and Metric Types

In the internetwork model in Figure 2-8, the routers turkey_creek, earp, holliday, and ringo are running EIGRP. The ringo router also has a BGP peer to the turkey_creek router and is running OSPF and to the clanton router.

Figure 2-8 *Route Tagging and Metric Setting*



To demonstrate route tagging and metric setting, the following example writes a route map on the ringo router. The route map will be used on the ringo router when redistributing EIGRP routes into OSPF. The route map will first tag the routes from the earp router, 172.16.10.3, with a tag of 3. Next, the route map will tag all other routes with a tag of 500 while making these routes OSPF external type 1 routes. Example 2-22 lists the configuration to accomplish this on the ringo router.

Example 2-22 *Configuration of the ringo Router*

```
hostname ringo
!
<<<text omitted>>>
!
router eigrp 65001
 redistribute bgp 65002
 network 172.16.0.0
```

Example 2-22 *Configuration of the ringo Router (Continued)*

```
 network 192.168.10.0
 default-metric 10000 1000 254 1 1500
 no auto-summary
 eigrp log-neighbor-changes
!
router ospf 7
 log-adjacency-changes
 redistribute eigrp 65001 subnets route-map set_tag3 ←Redistribute and call route-map
 redistribute bgp 65002
 network 172.16.10.10 0.0.0.0 area 0
 default-metric 10
!
router bgp 65002
 no synchronization
 bgp log-neighbor-changes
 neighbor 172.16.200.10 remote-as 65001
 neighbor 172.16.200.10 ebgp-multihop 10
 neighbor 172.16.200.10 update-source Loopback20
!
access-list 5 permit 172.16.10.3     ←Match routes from 172.16.10.3
access-list 50 permit any            ←Match all routes
!
route-map set_tag3 permit 100
 match ip route-source 5             ←Match routes from 172.16.10.3
 set tag 3                           ←Set the TAG value to 3
!
route-map set_tag3 permit 200
 match ip address 50                 ←Match all other routes
 set metric-type type-1              ←Set the OSPF metric to External Type-1
 set tag 500                         ←Set the TAG value to 500
!
```

By observing the route table of the ringo router followed by the OSPF database, you can see the effects of the route maps, as shown in Example 2-23.

Example 2-23 *Route Map Effects on the ringo Router*

```
ringo# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, 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 not set
B    192.168.192.0/24 [20/0] via 172.16.200.10, 01:07:04
     172.16.0.0/16 is variably subnetted, 8 subnets, 2 masks
D       172.16.200.0/24 [90/1915392] via 172.16.10.3, 01:07:08, Ethernet0
```

continues

Example 2-23 *Route Map Effects on the ringo Router (Continued)*

```
D       172.16.33.0/24 [90/1812992] via 172.16.10.3, 01:07:08, Ethernet0
D       172.16.34.0/24 [90/1812992] via 172.16.10.3, 01:07:08, Ethernet0
D       172.16.16.4/30 [90/2195456] via 172.16.10.4, 01:07:08, Ethernet0
D       172.16.16.0/30 [90/1787392] via 172.16.10.3, 01:07:08, Ethernet0
C       172.16.10.0/24 is directly connected, Ethernet0
D       172.16.2.0/24 [90/284160] via 172.16.10.3, 01:07:09, Ethernet0
C       172.16.100.0/24 is directly connected, Loopback20
ringo#
ringo# show ip ospf database
            OSPF Router with ID (172.16.100.10) (Process ID 7)
                Router Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum Link count
172.16.10.5     172.16.10.5     1151        0x80000015 0x4E2    1
172.16.100.10   172.16.100.10   1875        0x80000003 0xC969   1
                Net Link States (Area 0)
Link ID         ADV Router      Age         Seq#       Checksum
172.16.10.5     172.16.10.5     1151        0x80000003 0x1693
                Type-5 AS External Link States
Link ID         ADV Router      Age         Seq#       Checksum Tag
172.16.2.0      172.16.100.10   1875        0x80000002 0x8E2E   3
172.16.16.0     172.16.100.10   1875        0x80000002 0xE1CF   3
172.16.16.4     172.16.100.10   1875        0x80000002 0x4AF0   500
172.16.33.0     172.16.100.10   1875        0x80000002 0x3865   3
172.16.34.0     172.16.100.10   1875        0x80000002 0x2D6F   3
172.16.100.0    172.16.100.10   1875        0x80000002 0xE403   500
172.16.200.0    172.16.100.10   1875        0x80000002 0x4F1    3
192.168.192.0   172.16.100.10   1876        0x80000002 0x4A22   65001
ringo#
```

Notice that at the end of the OSPF database is the BGP route 192.168.192.0/24. This route has a tag of 65001 because BGP will try to preserve the AS_PATH attribute when redistributing BGP into an IGP that supports tags. BGP will use a tag value equal to its autonomous system ID.

You can also see the effects of the route map on the clanton router. Example 2-24 lists the route table of the clanton router, highlighting the different OSPF route types. Notice how the 172.16.16.4/30 and 172.16.100.0/24 routes are not set as default OSPF external type 2 routes, but are external type 1 routes. This is due to the **set route-type type-1** command in the route map on the ringo router.

Example 2-24 *Route Table of the clanton Router*

```
clanton# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
```

Example 2-24 *Route Table of the clanton Router (Continued)*

```
Gateway of last resort is not set
O E2 192.168.192.0/24 [110/10] via 172.16.10.10, 01:00:14, Ethernet0/0
     172.16.0.0/16 is variably subnetted, 8 subnets, 2 masks
O E2    172.16.200.0/24 [110/10] via 172.16.10.10, 01:00:14, Ethernet0/0
O E2    172.16.33.0/24 [110/10] via 172.16.10.10, 01:00:14, Ethernet0/0
O E2    172.16.34.0/24 [110/10] via 172.16.10.10, 01:00:14, Ethernet0/0
O E1    172.16.16.4/30 [110/20] via 172.16.10.10, 01:00:14, Ethernet0/0
O E2    172.16.16.0/30 [110/10] via 172.16.10.10, 01:00:14, Ethernet0/0
C       172.16.10.0/24 is directly connected, Ethernet0/0
O E2    172.16.2.0/24 [110/10] via 172.16.10.10, 01:00:15, Ethernet0/0
O E1    172.16.100.0/24 [110/20] via 172.16.10.10, 01:00:15, Ethernet0/0
clanton#
```

Route Maps and Policy-Based Routing

Sometimes in the modern internetwork, the forwarding decisions of a router need to be more complex than the decision information offered by the routing protocols and route table. Routers for the most part base their forwarding decisions on the destination address of packet. Policy-based routing enables the network engineer to configure policies that selectively cause packets to take paths that differ from the next-hop path specified by the route table. This section discusses the benefits and configuration of policy-based routing.

Policy-based routing offers the following benefits:

- **Forwarding decision not based on the destination address**—Policy routing enables the network engineer to define a path based on attributes of a packet, source/destination IP address, application port, and packet lengths, and to forward them according to a different policy. Policy routing can be configured to set the packet's next hop or the packet's default next hop/interface. Policy routing may also be used to route the packet to the null interface, essentially discarding them.
- **Quality of service (QoS)**—Route maps and PBR can provide QoS by enabling you to set the type of service (ToS) values and the IP precedence values in the IP header. QoS configuration is performed on the edge routers. This improves performance by preventing additional configuration on the core devices.
- **Cost saving by using alternative paths**—IP traffic can be manipulated with PBR, for instance, traffic such as large bulky batch file transfers can be sent over low-cost, low-bandwidth links, whereas more time-sensitive, user-interactive traffic is sent over higher-cost and higher-speed links.
- **Multiple and unequal path load sharing based on traffic characteristics**—Policy routing can be used to load balance traffic across multiple and unequal paths based on traffic characteristics versus the route cost.

Assuming that PBR is enabled and configured on the router and interface, PBR operates in the following manner:

Step 1 All packets *received* on a PBR-enabled interface are considered for policy routing. Each packet *received* on that interface is passed through an associated route map.

Step 2 The **match** commands are called by the route map; if all **match** commands are met, the route map is marked as a **permit** or **deny**, and no further route maps instances are executed. If a **match** statement is not present, the route map and any **set** commands apply to all packets.

Step 3 If the route map has a **permit** statement, all **set** commands are applied and the packet is forwarded according to the new policy. You can use multiple **set** commands in a single route map instance. Table 2-7 lists the **set** commands that are specific to PBR. If you use multiple **set** commands in conjunction with one another, they are applied in the same order as follows:

```
set ip {precedence [value_0-7 | name] | tos [value_0-8 | name]}
set ip next-hop ip_address
set interface interface_name
set ip default next-hop ip_address
set default interface interface_name
```

Each of these commands is covered in further detail later in this section.

Step 4 If the route map has a **deny** statement, normal forwarding is used, as specified in the route/forwarding table. The **set** statements will not be applied to the packet.

Step 5 At the end of all the route map instances, an implicit route map will deny all packets. If the packet has not found a match in the previous route map instances, the packet will hit the implicit deny route map instance. When this occurs, the packet will be forwarded by the router following the normal route table.

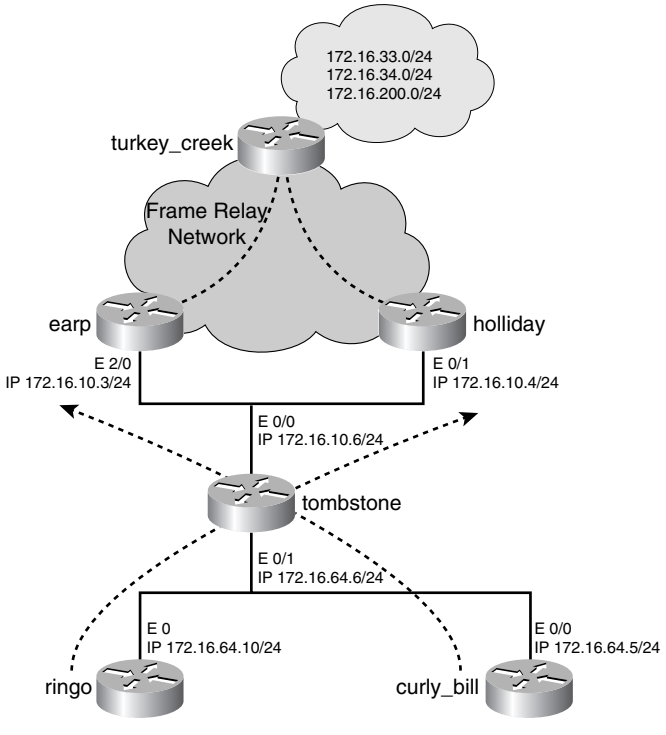
NOTE Policy routing only works on inbound packets; therefore, it must be applied to the incoming traffic or to the interface receiving the traffic to be policy routed. To policy route local traffic, you must have local policy routing enabled.

Practical Example: Policy-Based Routing

This section examines how you may use policy routing to control traffic in the internetwork. In the network model depicted in Figure 2-9, a policy route exists on the tombstone router to control traffic from the ringo and curly_bill routers. The policy states that all IP traffic from the

ringo router will be forwarded to holliday, whereas all IP traffic from the curly_bill router will be forwarded to earp. All other IP traffic will be handled by the normal routing procedure.

Figure 2-9 *Policy-Based Routing*



To control the traffic from the ringo and curly_bill routers, this example uses policy routing and route maps on the tombstone router. Policy routing will be enabled on the E0/1 interface of the tombstone router. This is the inbound interface, or the interface that will be receiving traffic from the ringo and curly_bill routers. The route map used in this model, policy_1, will have two route map instances. One will match packets from the ringo router, 172.16.64.10, and set the next hop to be 172.16.10.4, the holliday router. The other route map instance will match packets from the curly_bill router, 172.16.64.5, and set the next hop to be 172.16.10.3, the earp router.

The route/forwarding table on the tombstone router shows that there are two paths to the routes 172.16.33.0/24, 172.16.34.0/24, and 172.16.200.0/24 that reside on the turkey_creek router. One path passes through the earp router, whereas the other one passes through the holliday router. Example 2-25 lists the route table of the tombstone router.

Example 2-25 *Route Table of the tombstone Router*

```
tombstone# show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       i - IS-IS, 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 not set
     172.16.0.0/16 is variably subnetted, 9 subnets, 2 masks
D       172.16.200.0/24 [90/40665600] via 172.16.10.3, 03:58:24, Ethernet0/0
                        [90/40665600] via 172.16.10.4, 03:58:24, Ethernet0/0
D       172.16.33.0/24 [90/40563200] via 172.16.10.3, 03:58:24, Ethernet0/0
                       [90/40563200] via 172.16.10.4, 03:58:24, Ethernet0/0
D       172.16.34.0/24 [90/40563200] via 172.16.10.3, 03:58:24, Ethernet0/0
                       [90/40563200] via 172.16.10.4, 03:58:24, Ethernet0/0
D       172.16.16.4/30 [90/40537600] via 172.16.10.4, 03:59:03, Ethernet0/0
D       172.16.16.0/30 [90/40537600] via 172.16.10.3, 04:56:26, Ethernet0/0
C       172.16.10.0/24 is directly connected, Ethernet0/0
D       172.16.2.0/24 [90/284160] via 172.16.10.3, 03:59:03, Ethernet0/0
D       172.16.100.0/24 [90/409600] via 172.16.64.10, 03:49:42, Ethernet0/1
C       172.16.64.0/24 is directly connected, Ethernet0/1
tombstone#
```

By issuing an extended **traceroute** command on the tombstone router from the address 172.16.64.6 to 172.16.200.10, you can see that EIGRP is using load sharing between the earp and holliday routers. Policy routing will override this process by sending IP traffic from the ringo router to holliday, and IP traffic from curly_bill to earp, as shown in Example 2-26.

Example 2-26 *Extended Trace on the tombstone Router*

```
tombstone# traceroute
Protocol [ip]:
Target IP address: 172.16.200.10
Source address: 172.16.64.6
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]: 4
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 172.16.200.10
  1 172.16.10.4 0 msec
    172.16.10.3 0 msec
```

Example 2-26 *Extended Trace on the tombstone Router (Continued)*

```
    172.16.10.4 0 msec
    172.16.10.3 0 msec
  2 172.16.16.5 8 msec
    172.16.16.1 12 msec
    172.16.16.5 8 msec
    172.16.16.1 12 msec
tombstone#
```

The configuration needed for PBR on the tombstone router is listed in the next example, Example 2-27.

Example 2-27 *Policy-Based Routing Configuration on tombstone*

```
hostname tombstone
!
interface Ethernet0/0
 ip address 172.16.10.6 255.255.255.0
!
interface Ethernet0/1
 ip address 172.16.64.6 255.255.255.0
 ip route-cache policy         ←Optional fast switching for policy routing
 ip policy route-map policy_1  ←Call route-map "policy_1" for policy routing
!
router eigrp 65001
 network 172.16.0.0
 no auto-summary
!
access-list 100 permit ip host 172.16.64.10 any     ←match packets from 172.16.64.10
access-list 101 permit ip host 172.16.64.5 any      ←match packets from 172.16.64.5
!
route-map policy_1 permit 100                       ←route-map "policy_1"
 match ip address 100                               ←call ACL 100 for match criteria
 set ip next-hop 172.16.10.4                        ←set IP next hop to holliday
!
route-map policy_1 permit 200                       ←next route map instance
 match ip address 101                               ←call ACL 101 for match criteria
 set ip next-hop 172.16.10.3                       ←set IP next hop to the earp router
!
```

To test the new policy, issue the **traceroute** commands on the ringo and curly_bill routers to the IP address of 172.16.200.10, which resides on the turkey_creek router. The **traceroute** from the ringo router will show that packets pass to the tombstone router, and then to holliday, and finally to turkey_creek. Example 2-28 demonstrates the **traceroute** command on the ringo router with PBR enabled.

Example 2-28 traceroute *Performed on the ringo Router*

```
ringo# traceroute 172.16.200.10
Type escape sequence to abort.
Tracing the route to 172.16.200.10
  1 172.16.64.6 4 msec 4 msec 4 msec
  2 172.16.10.4 8 msec 4 msec 4 msec
  3 172.16.16.5 20 msec 8 msec *
ringo#
```

To test the new policy for the curly_bill router, issue the **traceroute** command on the curly_bill router to the IP address of 172.16.200.10. The packets will pass to the tombstone router, and then to earp, and finally to turkey_creek. Example 2-29 demonstrates the **traceroute** command on the curly_bill router.

Example 2-29 traceroute *Performed on the curly_bill Router*

```
curly_bill# traceroute 172.16.200.10
Type escape sequence to abort.
Tracing the route to 172.16.200.10
  1 172.16.64.6 4 msec 4 msec 4 msec
  2 172.16.10.3 4 msec 4 msec 0 msec
  3 172.16.16.1 12 msec 9 msec *
curly_bill#
```

CAUTION Whenever implementing policy routing, take care to consider the applications running on the network and the forward and return paths of the network traffic. In models such as this preceding example, you could implement policy routing on the turkey_creek router to avoid asymmetrical routing. *Asymmetrical routing* refers to when IP packets are forwarded along one path toward a destination, but follow a different path back, which can lead to problems with some applications, such as multicast.

Configuring Policy-Based Routing (PBR)

You can configure PBR by following these steps. Some of the steps may be omitted depending on your application for PBR.

Step 1 Define and configure the route map needed for the policy. This is accomplished with the **route-map** command, as discussed previously.

Step 2 Define and configure the **match** statements the route map will use. The most common **match** statements used are the following:

```
match ip address [access-list number]
```

The **match ip address** is used to call a standard, extended, or expanded-range ACL.

```
match length [min_packet_length_0-2147483647] [max_packet_length_0-2147483647]
```

The **match length** is used to match the Layer 3 packet length, in bytes, with all associated headers and trailers included. You must enter the minimum and maximum packet length. Use the **match length** command to policy route traffic based on packet size. You can deploy this to route traffic with large or small packet sizes to specific areas of the network.

Step 3 Configure and define the new routing policy with **set** commands. Multiple **set** commands may be used; if multiple commands are used, they are executed in the following order:

```
set ip {precedence [value_0-7 | name] | tos [value_0-8 | name]}
set ip next-hop ip_address
set interface interface_name
set ip default next-hop ip_address
set default interface interface_name
Set ip precedence {[1-7]|[routine|critical|flash|flash-
override|immediate|internet|network|priority]}
```

By setting the precedence, you are manipulating the first 3 bits, bits 0 through 2, of the 8-bit ToS field in the IP header. Earlier texts on TCP/IP state that this field is unused and ignored by routers, except for some routing protocols. This may have been true in the past; however, with the advent of Voice over IP and newer QoS features, the Precedence field is finding new life and meaning. IP precedence becomes a factor during periods of congestion on an interface. By default, Cisco routers do not manipulate the precedence value in the IP header; it remains at its original setting as when it arrived at the router. When Weighted Fair Queuing (WFQ) is enabled and the precedence bits are set, the packets are ordered for transmission according to the precedence value. The higher the precedence value, the higher its place in the queue for transmission. For the router to act on precedence, the link must be congested, and queuing must be enabled; otherwise, the packets are transmitted in first in, first out (FIFO) order. When setting precedence, you may use the numeric value of the precedence or the name of the precedence. Precedence should be set such that downstream IP devices can take advantage of the settings you use. Table 2-11 lists the valid names values for the **set precedence** command.

For detailed information about the **set precedence** command, see Chapter 5, "Integrated and Differentiated Services," and Chapter 6, "QoS – Rate Limiting and Queuing Traffic."

Table 2-11 **set precedence** *Commands in CISCO IOS Software Release 12.2*

Command	Function
routine	Set routine precedence (value = 0)
priority	Set priority precedence (value = 1)
immediate	Set immediate precedence (value = 2)
flash	Set Flash precedence (value = 3)
flash-override	Set Flash override precedence (value = 4)
critical	Set critical precedence (value = 5)
internet	Set internetwork control precedence (value = 6)
network	Set network control precedence (value = 7)

NOTE For a router's queuing mechanisms to act on the precedence bits, the following two conditions must be met:

- The outbound link must be congested.
- The outbound link must be configured for WFQ or Weighted Random Early Detection (WRED).

```
Set ip tos {[1-15]|[normal|min-delay|max-throughput|max-reliability|min-monetary-
cost|priority]}
```

The **set ip tos** command enables you to set bits 3 through 6 in the IP header's 8-bit ToS field. The ToS bits are composed of 4 bits. These bits are referred to as the following:

- **D bit (bit 3)**—Normal = off, low delay = on
- **T bit (bit 4)**—Normal = off, high throughput = on
- **R bit (bit 5)**—Normal = off, high reliability = on
- **C bit (bit 6)**—Unused in Cisco Routers. RFC 1349 calls it the *minimize monetary cost*. Some TCP/IP implementations ignore this bit or implement it differently.

Bit 7 in the ToS field is currently unused and is set to 0. If all 4 bits are set to 0, it implies normal service.

Table 2-12 lists the recommended guidelines for setting ToS by protocol type.

Table 2-12 *Recommended ToS Values by Protocol*

Protocol	min-delay	max-throughput	max-reliability	min-monetary-cost
Telnet/Rlogin	1	0	0	0
HTTP	1	0	0	0
FTP control	1	0	0	0
FTP data	0	1	0	0
Any bulk data	0	1	0	0
TFTP	1	0	0	0
SMTP commands	1	0	0	0
SMTP data phase	0	1	0	0
DNS UDP query	1	0	0	0
DNS TCP query	0	0	0	0
DNS zone xfer	0	1	0	0
ICMP	0	0	0	0
IGPs	0	0	1	0
SNMP	0	0	1	0
BOOTP	0	0	0	0
NNTP	0	0	0	1

NOTE Cisco IOS Software considers the precedence bits of the ToS field if there is traffic that is queued in WFQ, WRED, or Weighted Round Robin (WRR). The precedence bits are not considered when policy routing, Priority Queuing (PQ), Custom Queuing (CQ), or Class-Based Weighted Fair Queuing (CBWFQ) are configured.

```
set ip next-hop {ip_address}
```

Use this command to set IP address of the next-hop router to which the packet will be forwarded. The IP address used must be an adjacent router.

```
set interface {interface_name}
```

Use this command to set the output interface for the matched packet.

```
set ip default next-hop {ip_address}
```

This command is used like the **ip next-hop** command. It specifies which IP address to forward packets to if there is not an explicit route to the destination in the route table. Think of this command as a default route to use for policy routing. The next-hop address must be an adjacent router.

```
set default interface {interface_name}
```

This command functions much like the **ip default next-hop** command; it specifies which interface to forward a matched packet to if there is not an explicit route to the destination. Used on point-to-point links.

NOTE The **set ip next-hop** and **set ip default next-hop** commands are similar but function differently. The **set ip next-hop** command causes the router to use policy routing first and then use the route table. The **set ip default next-hop** command causes the router to use the route table first and then policy route to the specified default next hop.

Step 4 (Optional) Define and configure any ACLs that will be used with the new routing policy. With extended ACLs, for example, you can use policy to forward traffic based on traffic type (for instance, traffic one way, and FTP traffic another). You can also use ACLs to route traffic from specific addresses. When you use standard ACLs, policy routing compares the source IP address in the packet to the ACL.

Step 5 Configure policy routing on the inbound interface. To configure policy routing for an interface, use the following interface command:

```
router(config-if)# ip policy route-map route-map_name
```

Step 6 (Optional) Enable fast switching for PBR. In Cisco IOS Software Release 12.0, PBR can be fast switched. Prior to Cisco IOS Software Release 12.0, PBR could only be processed switched. In a process-switched environment, the switching rate is approximately 1000 to 10,000 packets per second. This speed was not considered fast enough for many applications. You can enable fast switching of PBR with the following interface command:

```
router(config-if)# ip route-cache policy
```

PBR must be configured before you configure fast-switched PBR. Fast-switched PBR does not support the **set ip default next-hop** and **set default interface** commands. The **set interface** command is supported over point-to-point links or with a static route cache entry equal to the interface specified in the **set interface** command.

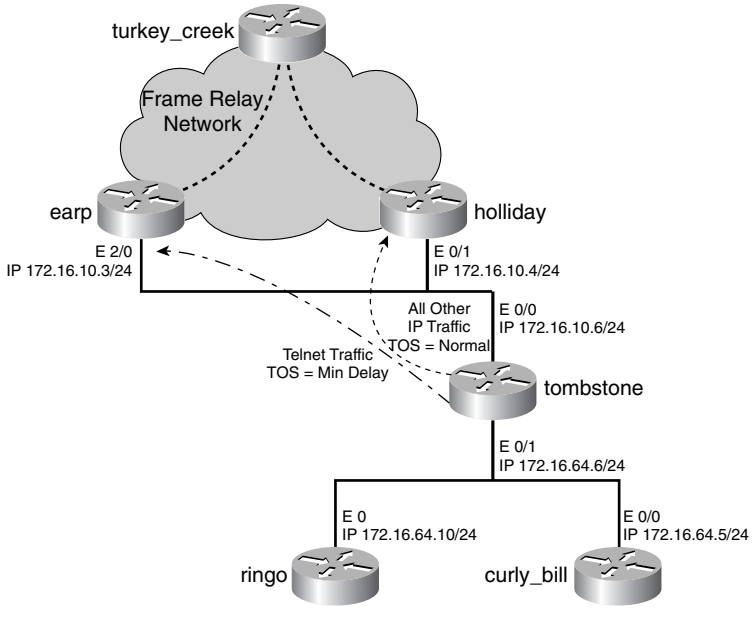
Step 7 (Optional) Configure local PBR. Packets generated by the router are not policy routed. If you want to policy route traffic generated by the router, you must enable it. To enable local PBR, use the following global configuration command.

```
router(config)# ip local policy route-map route-map_name
```

Practical Example: Configuring PBR and Setting ToS

In this section, you apply a couple of these concepts to a practical example in policy routing. For the network depicted in Figure 2-10, create a policy route that will forward Telnet traffic to the earp router, 172.16.10.3, while setting the ToS bit to minimum delay. All other IP traffic will be forwarded to the holliday router, 172.16.10.4.

Figure 2-10 *Policy-Based Routing*



Following the multistep process for configuring PBR, Steps 1 through 3 call for you to first configure the route map with the necessary **match** and **set** commands. The route map will call an ACL that matches Telnet traffic, and the **set** command will set the IP next hop to be the IP address of the earp router. Table 2-12 specifies that Telnet traffic should have the ToS set to **min-delay**; therefore, the route map will set this bit for Telnet traffic in the ToS value to **min-delay**. Another route map instance will be used to match all other traffic and forward it to the holliday

router. Because the route map instance will match all traffic, there is no need to include a **match** command. Example 2-30 lists the route map configuration on the tombstone router to accomplish this.

Example 2-30 *Route Map Configuration on the tombstone Router*

```
route-map policy_2 permit 100
 match ip address 101                 ←Call access-list 101
 set ip next-hop 172.16.10.3          ←Set the next hop to 172.16.10.3/earp
 set ip tos min-delay                 ←Set the TOS to min-delay
!
route-map policy_2 permit 200
 set ip next-hop 172.16.10.4         ←Match all routes and set the next hop
                                     ←to 172.16.10.4/holliday
```

Now you must configure any ACLs the route map will need. In this case, configure a single ACL to match TCP telnet traffic from any IP address. The ACL you will use resembles the following:

```
access-list 101 permit tcp any any eq telnet
```

There is no need to write an ACL to catch all the regular traffic. As discussed earlier, the absence of a **match** statement, such as in the second route map instance, will match all routes or all packets.

The last two steps call for you to apply the policy route to an interface and to enable fast switching for PBR. This is accomplished with the **interface** commands **ip policy route-map** and **ip route-cache policy**. In this model, you will enable PBR on the E0/1 interface of the tombstone router. With PBR enabled on the E0/1 interface, all Telnet traffic will be forwarded to the earp router, whereas all other IP traffic will be forwarded to the holliday router. Example 2-31 lists the complete PBR configuration of the tombstone router.

Example 2-31 *PBR Configuration on the tombstone Router*

```
hostname tombstone
!
interface Ethernet0/0
 ip address 172.16.10.6 255.255.255.0
!
interface Ethernet0/1
 ip address 172.16.64.6 255.255.255.0
 ip route-cache policy                      ←enable PBR fast-switching
 ip policy route-map policy_2               ←Call route-map "policy_2" for PBR
!
router eigrp 65001
 network 172.16.0.0
 no auto-summary
 no eigrp log-neighbor-changes
```

Example 2-31 *PBR Configuration on the tombstone Router (Continued)*

```
!
access-list 101 permit tcp any any eq telnet ←Match Telnet traffic
!
priority-list 1 protocol ip high             ←Priority queuing for TOS enforcement
priority-list 1 default low
!
route-map policy_2 permit 100
 match ip address 101                        ←call access-list 101 and match Telnet
 set ip next-hop 172.16.10.3                 ←Set the next hop to earp/172.16.10.3
 set ip tos min-delay                        ←Set TOS min-delay bit
!
route-map policy_2 permit 200                ←Match all other traffic
 set ip next-hop 172.16.10.4                ←Set the next hop to holliday/172.16.10.4
!
```

In this model, because you are setting ToS values, you need to configure WRED or WFQ on the outbound interface. WFQ is not the default queuing method on Ethernet interfaces. It is the default queuing method on serial interfaces with 2.048 Mbps or less of bandwidth. This portion of the configuration is not present in this example. For more information on configuring WRED and WFQ, see Chapters 5 and 6.

Big Show for Route Maps

CCIE Practical Studies, Volume I introduced what was called the *Big Show* and *Big D*. These terms were used because the discussion focused on only a select few **show** and **debug** commands considered most useful.

The Big Show and Big D commands for route maps are rather limited in their use. The best way to test the functionality of route maps and policy routing is to actually see how they are performing by viewing the route table and using **traceroute** commands. The **show** commands offered by Cisco are very good at showing where the route map is applied and the logical order in which it is operated. The Big Show commands discussed here are as follows:

- **show route-map**
- **show ip policy**
- **show ip cache policy**

The **show route-map** command enables you to determine the logical order and execution of the route map. If PBR is enabled, the command also shows the number of matches and the number of bytes that were policy routed. Working from the previous network models, Example 2-32 demonstrates the **show route-map** command on the tombstone router.

Example 2-32 show route-map *Command on the tombstone Router*

```
tombstone# show route-map
route-map policy_2, permit, sequence 100
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 172.16.10.3
    ip tos min-delay
  Policy routing matches: 264 packets, 15852 bytes
route-map policy_2, permit, sequence 200
  Match clauses:
  Set clauses:
    ip next-hop 172.16.10.4
  Policy routing matches: 60 packets, 4478 bytes
route-map policy_1, permit, sequence 100
  Match clauses:
    ip address (access-lists): 100
  Set clauses:
    ip next-hop 172.16.10.4
    ip tos max-throughput
  Policy routing matches: 85 packets, 6880 bytes
route-map policy_1, permit, sequence 200
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 172.16.10.3
  Policy routing matches: 43 packets, 3318 bytes
tombstone#
```

Use the **show ip policy** command to verify which interfaces have PBR enabled and which route map they are currently using for PBR. Example 2-33 demonstrates the **show ip policy** command on the tombstone router.

Example 2-33 show ip policy *Command on the tombstone Router*

```
tombstone# show ip policy
Interface      Route map
Ethernet0/1    policy_2
```

You can use the **show ip cache policy** command to verify whether fast switching is enabled for policy routing. This command shows the policy type, the route map in use, and the age of the cache entries. If the policy is a next-hop policy, the next hop also displays. Example 2-34 lists the output of the **show ip cache policy** command on the tombstone router.

Example 2-34 show ip cache policy *Command on the tombstone Router*

```
tombstone# show ip cache policy
Total adds 4, total deletes 2
Type Routemap/sequence      Age       Interface              Next Hop
NH   policy_2/100            00:38:27  Ethernet0/0            172.16.10.3
NH   policy_2/200            00:43:56  Ethernet0/0            172.16.10.4
tombstone#
```

Lab 3: Configuring Complex Route Maps and Using Tags—Part I

Practical Scenario

Route maps are one of most powerful features you can use on a router. You can use them during redistribution, in PBR, in BGP, and in many other scenarios. This lab gives you practice in configuring complex route maps that will be used during redistribution. You then practice setting and using route tags.

Lab Exercise

GameNetworks.com is an upstart company focusing on providing WAN and LAN connectivity for console games. GameNetworks.com enables its customers to play the latest and greatest console games online through its private network. GameNetworks.com has two new locations in Wisconsin and California. Your task is to configure an IP network using the following strict design guidelines:

- Configure the GameNetworks.com IP network as depicted in Figure 2-11. Use EIGRP as the routing protocol and 2002 as the autonomous system ID on the wisconsin_x, unreal, and halo routers. Use EIGRP as the routing protocol on the california_x router and the gamenet router; the autonomous system of this router will be 65001.
- Join the EIGRP routing domains with OSPF on the gamenet and wisconsin_x routers.
- Configure the Frame Relay network as depicted in Figure 2-11.
- Configure all IP addresses as depicted in Figure 2-11.
- Use the “Lab Objectives” section for configuration specifics.