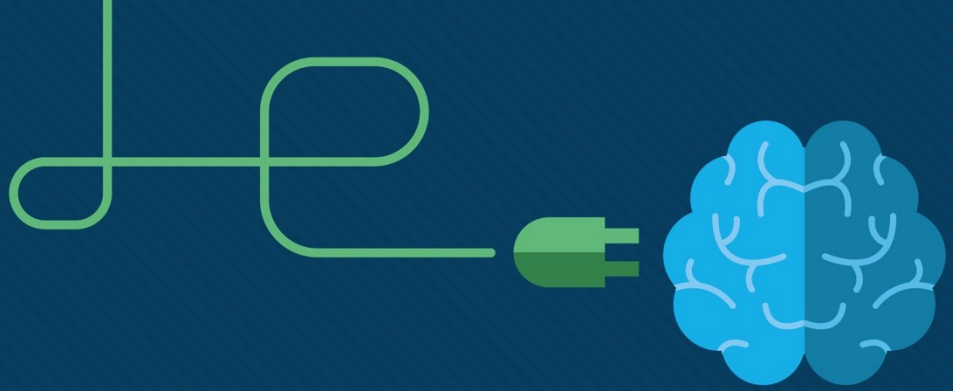


Chapter 12: Advanced BGP

Instructor Materials

CCNP Enterprise: Core Networking

Chapter 12 Content

This chapter covers the following content:

BGP Multihoming - This section reviews the methods of providing resiliency through redundant BGP connections, along with desired and undesired design considerations for Internet and MPLS connections (branch and data center).

Conditional Matching - This section provides an overview of how network prefixes can be conditionally matched with ACLs, prefix lists, and regular expressions.

Route Maps - This section explains the structure of a route map and how conditional matching and conditional actions can be combined to filter or manipulate routes.

BGP Route Filtering and Manipulation - This section expands on how conditional matching and route maps work by applying real-world use cases to demonstrate the filtering or manipulation of BGP routes.

BGP Communities - This section explains the BGP well-known mandatory path attribute and how it can be used to tag a prefix to have route policies applied by routers in the same autonomous system or in an external autonomous system.

Understanding BGP Path Selection - This section describes the logic used by BGP to identify the best path when multiple routes are installed in the BGP table.

BGP Multihoming

- An organization must incorporate redundancies in the network architecture to ensure that there are not any single points of failure (SPOF).
- The simplest method of providing redundancy is to provide a second circuit.
- Adding a second circuit and establishing a second BGP session is known as BGP multihoming.

Resiliency in Service Providers

By using a different SP, if one SP has problems in its network, network traffic can still flow across the other SP. In addition, adding more SPs means traffic can select an optimal path between devices due to the BGP best-path algorithm

Scenario 1: Same SP. This design accounts for link failures. However, a failure on either router or within SP1's network results in a network failure.

Scenario 2: Same SP. This design accounts for link failures, However, a failure on R1 or within SP1's network results in a network failure.

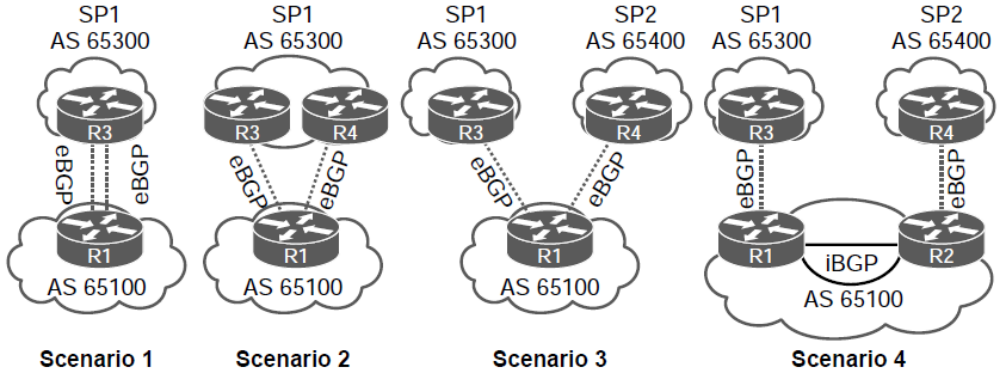


Figure 12-1 *Common BGP Multihoming Scenarios*

Scenario 3: Different SPs. This design accounts for link failures and failures in either SP's network, and it can optimize routing traffic. However, a failure on R1 results in a network failure.

Scenario 4: Different SPs. Design accounts for link failures and failures in either SP's network, and it can optimize routing traffic. Also it accounts for network failure on the interior Network by having R1 and R2 form an iBGP session with each other.

Internet Transit Routing

Using BGP to connect with more than one SP, runs the risk of the autonomous system (AS) becoming a transit AS.

A user that connects to SP3 (AS 300) routes through the enterprise network (AS 500) to reach a server that attaches to SP4 (AS 400) because the AS_Path is much shorter than going through SP1 and SP2's networks.

The AS 500 network is providing transit routing to everyone on the Internet, which can saturate AS 500's peering links.

Instead of using the default BGP routing policy transit routing can be avoided by applying outbound BGP route policies that only allow for local BGP routes to be advertised to other autonomous systems.

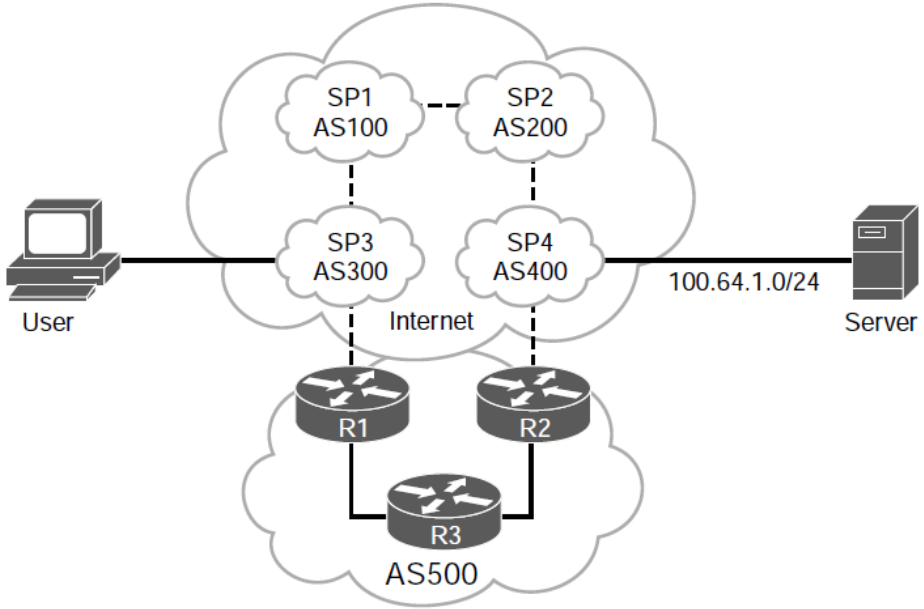


Figure 12-2 *Enterprise Transit Routing*

Branch Transit Routing

Figure 12-3 shows a multihomed design. All the routers are configured to prefer the MPLS SP2 transport over the MPLS SP1 transport (active/passive).

All the routers peer and advertise all the routes via eBGP to the SP routers. All the routers set the local preference for MPLS SP2 to a higher value to route traffic through it.

The traffic between the sites uses the preferred SP network (MPLS SP2) in both directions. This simplifies troubleshooting when the traffic flow is symmetric (same path in both directions) as opposed to asymmetric forwarding (a different path for each direction). The path is considered deterministic when the flow between sites is predetermined and predictable.

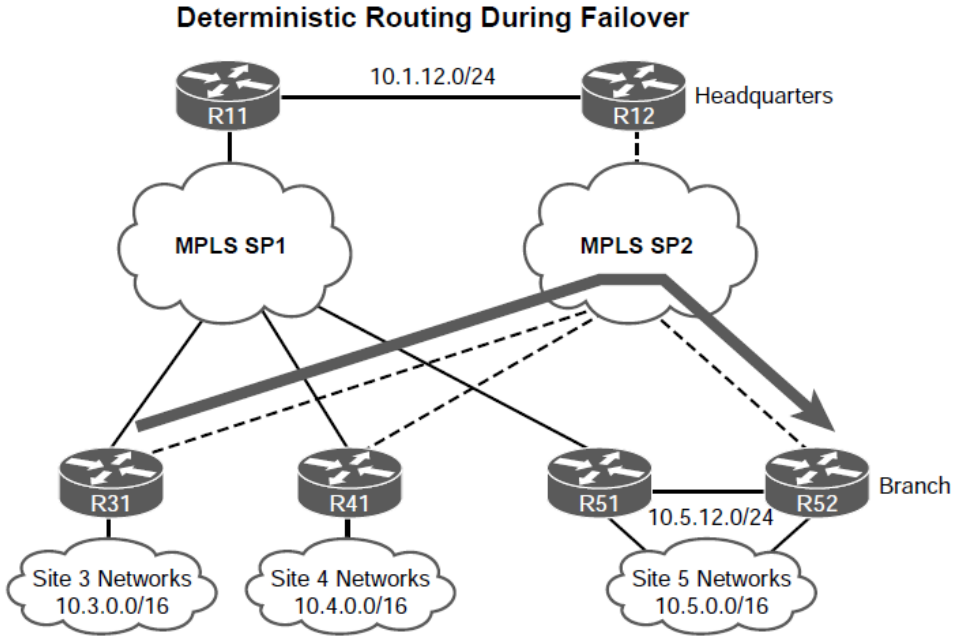


Figure 12-3 *Deterministic Routing*

Branch Transit Routing (Cont.)

Figure 12-4 shows a failure scenario with the R41 branch router providing transit connectivity between Site 3 and Site 5. Unplanned transit presents the following issues:

- The transit router's circuits can become oversaturated.
- The routing patterns can become unpredictable and nondeterministic. In this scenario, traffic from R31 may flow through R41, but the return traffic may take a different return path. This prevents deterministic routing, and complicates troubleshooting.

Multihomed environments should be configured so that branch routers cannot act as transit routers. Transit routing can be avoided by configuring outbound route filtering at each branch site. Branch sites do not advertise what they learn from the WAN but advertise only networks that face the LAN.

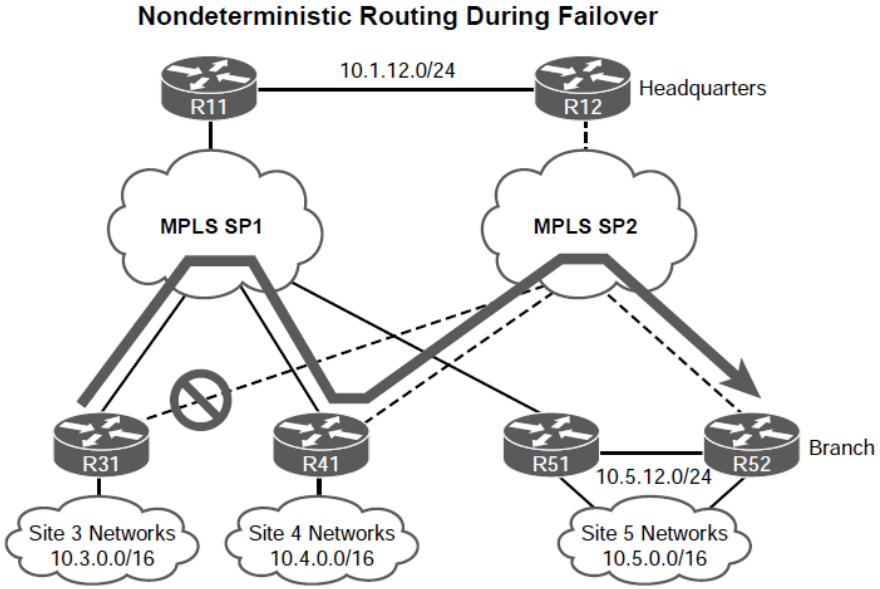


Figure 12-4 *Nondeterministic Routing During Failover*

Note: Transit routing at the data center or other planned locations is normal in enterprise designs as they have accounted for the bandwidth.

Conditional Matching

- Applying bulk changes to routes on a neighbor-by-neighbor basis (or interface-by-interface basis for IGPs) does not easily allow for tuning of the network.
- This section reviews some of the common techniques used to conditionally match a route—using access control lists (ACLs), prefix lists, regular expressions (regex), and AS path ACLs.

Access Control Lists

Originally, access control lists (ACLs) were intended to filter packets flowing in or out of a network interface, similar to a firewall. Today, ACLs provide packet classification for a variety of features, such as quality of service (QoS), or for identifying networks within routing protocols.

ACLs are composed of access control entries (ACEs), which are entries in the ACL that identify the action to be taken (permit or deny) and the relevant packet classification. ACE placement within an ACL is important, and unintended consequences may result from ACEs being out of order.

ACLs are classified into two categories Standard and Extended:

- **Standard ACLs -** Define packets based solely on the source network.
- **Extended ACLs -** Define packets based on source, destination, protocol, port, or a combination of other packet attributes.
- **Named ACLs -** provide relevance to the functionality of the ACL, can be used with standard or extended ACLs, and are generally preferred.

Note: This course is concerned with routing and limits the scope of ACLs to source, destination, and protocol.

Standard ACLs

Standard ACLS use a numbered entry 1–99, 1300–1999, or a named ACL. The following is the process for defining a standard ACL:

- **Step 1**. Define the ACL by using the command **ip access-list standard** *{acl-number | acl-name}* and placing the CLI in ACL configuration mode.
- **Step 2**. Configure the specific ACE entry with the command *[sequence]* **{permit | deny** *} source source-wildcard*. In lieu of using source source-wildcard, the keyword *any* replaces 0.0.0.0 0.0.0.0, and use of the *host* keyword refers to a /32 IP address so that the source-wildcard can be omitted.

Table 12-2 Standard ACL-to-Network Entries

ACE Entry	Networks
permit any	Permits all networks
permit 172.16.0.0 0.0.255.255	Permits all networks in the 172.16.0.0 range
permit host 192.168.1.1	Permits only the 192.168.1.1/32 network

Extended ACLs

Extended ACLs use a numbered entry 100–199, 2000–2699

The following is the process for defining an extended ACL:

- **Step 1.** Define the ACL by using the command **ip access-list extended** *{acl-number | acl-name}* and placing the CLI in ACL configuration mode.
- **Step 2.** Configure the specific ACE entry with the command *[sequence]* **{permit | deny}** *protocol source source-wildcard destination destination-wildcard*. The behavior for selecting a network prefix with an extended ACL varies depending on whether the protocol is an IGP (EIGRP, OSPF, or IS-IS) or BGP.

ACE Entry	Networks
permit any	Permits all networks
permit ip 172.16.0.0 0.0.255.255	Permits the network 172.16.0.0 range
permit ip host 92.168.1.1	Permits only 192.168.1.1/32

Extended ACL IGP Network Selection

When ACLS are used for IGP network selection, the source fields of the ACL are used to identify the network, and the destination fields identify the smallest prefix length allowed in the network range.

The table provides sample ACL entries and specifies the networks that would match with the extended ACL. Notice that the subtle difference in the destination wildcard for the 172.16.0.0 network affects the network ranges that are permitted in the second and third rows of the table.

Table 12-3 Extended ACL for IGP Route Selection

ACE Entry	Networks
permit ip any any	Permits all networks
permit ip host 172.16.0.0 host 255.240.0.0	Permits all networks in the 172.16.0.0/12 range
permit ip host 172.16.0.0 host 255.255.0.0	Permits all networks in the 172.16.0.0/16 range
permit host 192.168.1.1	Permits only the 192.168.1.1/32 network

Extended ACL BGP Network Selection

Extended ACLs react differently when matching BGP routes than when matching IGP routes.

The source fields match against the network portion of the route, and the destination fields match against the network mask.

permit *protocol source source-wildcard destination destination-wildcard*

Matches Networks **Matches Network Mask**

Table 12-4 Extended ACL for BGP Route Selection

Extended ACL	Matches These Networks
permit ip 10.0.0.0 0.0.0.0 255.255.0.0 0.0.0.0	Permits only the 10.0.0.0/16 network
permit ip 10.0.0.0 0.0.255.0 255.255.255.0 0.0.0.0	Permits any 10.0.x.0 network with a /24 prefix length
permit ip 172.16.0.0 0.0.255.255 255.255.255.0 0.0.0.255	Permits any 172.16.x.x network with a /24 network to /32 prefix length
permit ip 172.16.0.0 0.0.255.255 255.255.255.128 0.0.0.127	Permits any 172.16.x.x network with a /25 network to /32 prefix length

Prefix Match Specifications

A prefix list identifies a specific IP address, network, or network range and allows for the selection of multiple networks with a variety of prefix lengths by using a prefix match specification.

Many network engineers prefer this over the ACL network selection method.

A prefix match specification contains two parts: a high-order bit pattern and a high-order bit count, which determines the high-order bits in the bit pattern to be matched.

Some documentation refers to the high-order bit pattern as the address or network, and the high-order bit count as the mask length.

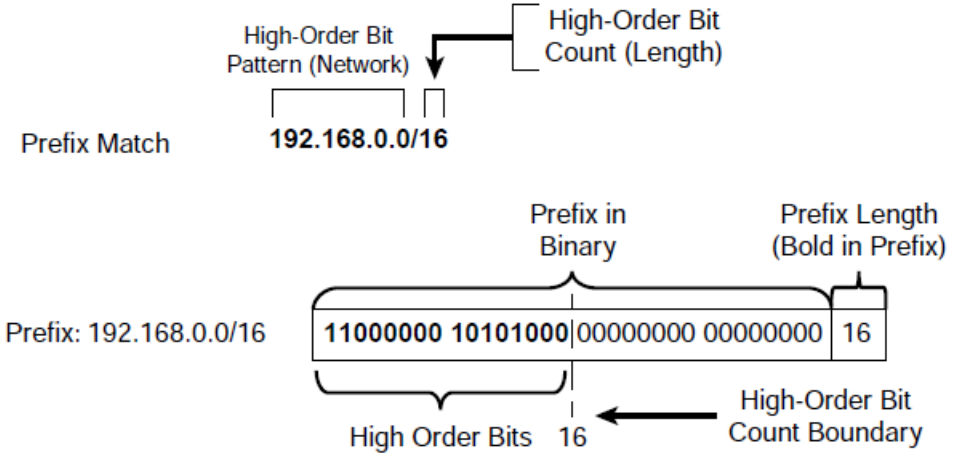


Figure 12-6 *Basic Prefix Match Pattern*

Prefix Matching with Length Parameters

The power of prefix matching comes in using matching length parameters to identify multiple networks with specific prefix lengths with one statement.

The matching length parameter options are:

- **le: Less than or equal to, <=**
- **ge: Greater than or equal to, >=**

Figure 12-7 demonstrates the prefix match specification with the high-order bit pattern 10.168.0.0 and high-order bit count 13. In this example, the matching length of the prefix must be greater than or equal to 24.

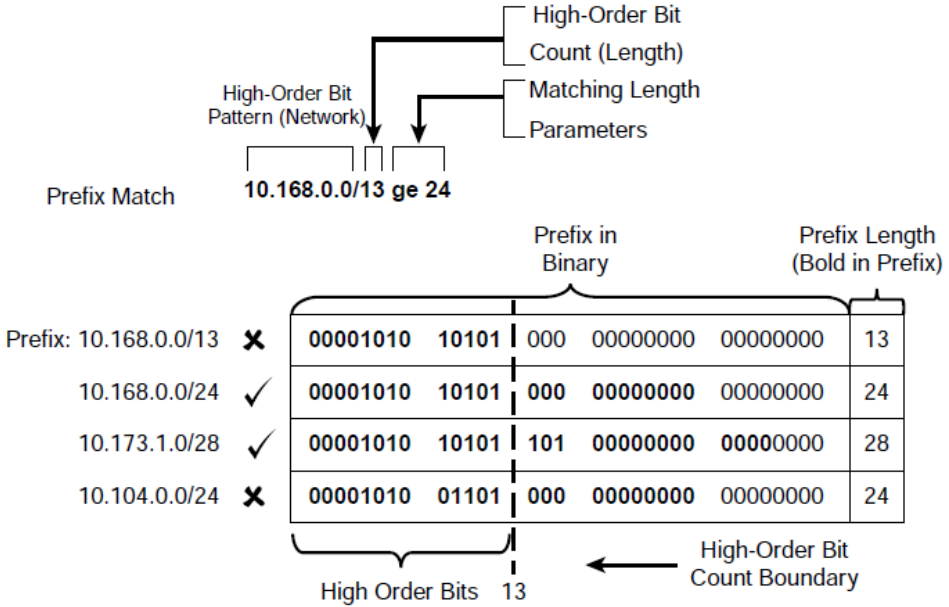


Figure 12-7 *Prefix Match Pattern with Matching Length Parameters*

Prefix Matching with Length Parameters (Cont.)

Figure 12-8 demonstrates a prefix match specification with the high-order bit pattern 10.0.0.0, high-order bit count 8, and matching length between 22 and 26.

The 10.0.0.0/8 prefix does not match because the prefix length is too short. The 10.0.0.0/24 network qualifies because the bit pattern matches, and the prefix length is between 22 and 26. The 10.0.0.0/30 prefix does not match because the bit pattern is too long. Any prefix that starts with 10 in the first octet and has a prefix length between 22 and 26 will match.

Matching to a specific prefix length that is higher than the high-order bit count requires that the ge-value and le-value match.

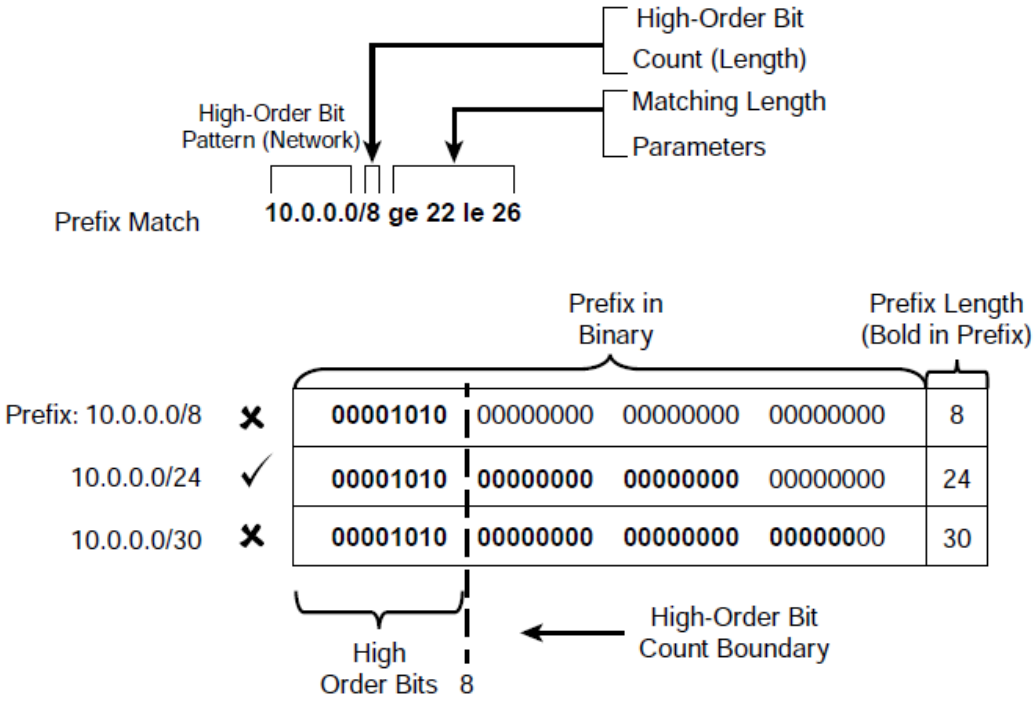


Figure 12-8 *Prefix Match with Ineligible Matched Prefixes*

Prefix Lists – IPv4

Prefix lists can contain multiple prefix matching specification entries that contain a permit or deny action.

Prefix lists process in sequential order in a top-down fashion, and the first prefix match processes with a permit or deny action.

Prefix lists are configured with the command ***ip prefix-list*** *prefix-listname [***seq** *sequence-number] {***permit** | **deny***} high-order-bit-pattern/high-order-bit-count [***ge** *ge-value] [***le** *le-value]*

The sequence number auto-increments by 5, based on the highest sequence number. The first entry is 5.

IOS and IOS XE require that the ge-value be greater than the high-order bit count and that the le-value be greater than or equal to the ge-value: **high-order bit count < ge-value <= le-value**

Example 12-1 *Sample Prefix List*

```
ip prefix-list RFC1918 seq 5 permit 192.168.0.0/13 ge 32
ip prefix-list RFC1918 seq 10 deny 0.0.0.0/0 ge 32
ip prefix-list RFC1918 seq 15 permit 10.0.0.0/7 ge 8
ip prefix-list RFC1918 seq 20 permit 172.16.0.0/11 ge 12
ip prefix-list RFC1918 seq 25 permit 192.168.0.0/15 ge 16
```

Prefix Lists – IPv6

The prefix matching logic works exactly the same for IPv6 networks as for IPv4 networks.

The most important thing to remember is that IPv6 networks are notated in hex and not in binary when identifying ranges. Ultimately, however, everything functions at the binary level.

IPv6 prefix lists are configured with the global configuration command **ipv6 prefix-list** *prefix-list-name [***seq** *sequence-number] {***permit** | **deny***} high-order-bit-pattern/highorder-bit-count [***ge** *ge-value] [***le** *le-value].*

Example 12-2 provides a sample prefix list named PRIVATE-IPV6.

Example 12-2 *Sample IPv6 Prefix List*

```
ipv6 prefix-list PRIVATE-IPV6 seq 5 permit 2001:2::/48 ge 48
ipv6 prefix-list PRIVATE-IPV6 seq 10 permit 2001:db8::/32 ge 32
```

Regular Expressions (regex)

There may be times when conditionally matching on network prefixes may be too complicated, and identifying all routes from a specific organization is preferred. In such a case, path selection can be made by using a BGP AS_Path.

Regular expressions (regex) are used to parse through the large number of available ASNs (4,294,967,295). The BGP table can be parsed with regex by using the command **show bgp** *afi safi* **regexp** *regex-pattern*.

^ indicates the start of a string, **$** indicates the end of a string, **_** matches a space, **+** matches one or more instance, **?** matches one or no instances, **[ ]** match a single character or nesting within a range, **-** is for a range

Table 12-6 Common BGP Regular Expressions

Regular Expression	Meaning
^$	Local originating routes
permit ^200_	Only routes from neighbor AS 200
permit _200$	Only routes originating from AS 200
permit _200_	Only routes that pass through AS 200
permit ^[0-9]+ [0-9]+ [0-9]+?	Routes with three or fewer AS_Path entries

Route Maps

- Route maps can filter networks much the same way as ACLs, but they also provide additional capability through the addition or modification of network attributes.
- To influence a routing protocol, a route map must be referenced from the routing protocol.
- Route maps are critical to BGP because they are the main component in modifying a unique routing policy on a neighbor-by-neighbor basis.

Route Map Components and Syntax

A route map has four components:

- **Sequence number -** Dictates the processing order of the route map.
- **Conditional matching criteria -** Identifies prefix characteristics (network, BGP path attribute, next hop, …)
- **Processing action -** Permits or denies the prefix.
- **Optional action -** Allows for manipulations, depending on how the route map is referenced on the router. Actions can include modification, addition, or removal of route characteristics.

A route map uses the command syntax: **route-map** *route-map-name* ***[*permit | deny*]*** ***[****sequence-number****]***

- If a processing action is not provided, the default value permit is used.
- If a sequence number is not provided, the sequence number is incremented by 10 automatically.
- If a matching statement is not included, an implied all prefixes is associated with the statement.
- Processing within a route map stops after all optional actions have processed (if configured) after matching a conditional matching criterion.

Route Map Components and Syntax (example)

Example 12-3 provides a sample route map to demonstrate the four components of a route map.

The conditional matching criterion is based on network ranges specified in an ACL.

Comments have been added to this example to explain the behavior of the route map in each sequence.

Example 12-3 *Sample Route map*

```
route-map EXAMPLE permit 10
 match ip address ACL-ONE
! Prefixes that match ACL-ONE are permitted. Route-map completes processing upon a match

route-map EXAMPLE deny 20
 match ip address ACL-TWO
! Prefixes that match ACL-TWO are denied. Route-map completes processing upon a match

route-map EXAMPLE permit 30
 match ip address ACL-THREE
 set metric 20
! Prefixes that match ACL-THREE are permitted and modify the metric. Route-map completes
! processing upon a match

route-map EXAMPLE permit 40
! Because a matching criteria was not specified, all other prefixes are permitted
! If this sequence was not configured, all other prefixes would drop because of the
! implicit deny  for all route-maps
```

Route Map Conditional Matching

Command syntax for common methods for conditionally matching prefixes and their usage.

Table 12-7 Conditional Match Options

Match Command	Description
match as-path *acl-number*	Selects prefixes based on a regex query to isolate the ASN in the BGP path attribute (PA) AS path. The AS path ACLs are numbered 1-500. This command allows for multiple match variables.
match ip address {*acl-number* | *acl-name*}	Selects prefixes based on network selection criteria defined in the ACL. This command allows for multiple match variables.
match ip address prefix-list *prefix-list-name*	Selects prefixes based on prefix selection criteria. This command allows for multiple match variables.
match local-preference *local-preference*	Selects prefixes based on the BGP attribute local preference. This command allows for multiple match variables.
match metric {*1-4294967295* | external *1-4294967295}*[+- *deviation*]	Selects prefixes based on a metric that can be exact, a range, or within acceptable deviation.
match tag *tag-value*	Selects prefixes based on a numeric tag (0 to 4294967295) that was set by another router. This command allows for multiple match variables.

Route Map Multiple Match Variables and Options

If there are multiple variables (ACLs, prefix lists, tags, and so on) configured for a specific route map sequence, only one variable must match for the prefix to qualify. The Boolean logic uses an OR operator for this configuration.

In Example 12-4, sequence 10 requires that a prefix pass ACL-ONE or ACL-TWO. Notice that sequence 20 does not have a match statement, so all prefixes that are not passed in sequence 10 will qualify and are denied.

If there are multiple match options configured for a specific route map sequence, both match options must be met for the prefix to qualify for that sequence. The Boolean logic uses an AND operator for this configuration.

Example 12-4 *Multiple Match Variables Route Map Example*

```
route-map EXAMPLE permit 10
 match ip address ACL-ONE ACL-TWO
!
route-map EXAMPLE deny 20
```

Example 12-5 *Multiple Match Options Route Map Example*

```
route-map EXAMPLE permit 10
 match ip address ACL-ONE
 match metric 550 +- 50
```

In Example 12-5, sequence 10 requires that the prefix match ACL-ONE and that the metric be a value between 500 and 600. If the prefix does not qualify for both match options, the prefix does not qualify for sequence 10 and is denied because another sequence does not exist with a permit action.

Route Map Complex Matching Problems

Route maps process using an order of evaluation: the sequence, conditional match criteria, processing action, and optional action in that order. Any deny statements in the match component are isolated and excluded from the route map sequence action.

The prefix 172.16.1.0/24 is denied by ACL-ONE, which implies that there is not a match in sequences 10 and 20; therefore, the processing action (permit or deny) is not needed. Sequence 30 does not contain a match clause, so any remaining routes are permitted. The prefix 172.16.1.0/24 would pass on sequence 30 with the metric set to 20. The prefix 172.16.2.0/24 would match ACL-ONE and would pass in sequence 10.

Example 12-6 *Complex Matching Route Maps*

```
ip access-list standard ACL-ONE
 deny   172.16.1.0 0.0.0.255
 permit 172.16.0.0 0.0.255.255

route-map EXAMPLE permit 10
 match ip address ACL-ONE
!
route-map EXAMPLE deny 20
 match ip address ACL-ONE
!
route-map EXAMPLE permit 30
 set metric 20
```

Route Map Optional Actions

In addition to permitting the prefix to pass, route maps can modify route attributes. The table provides a brief overview of the most popular attribute modifications.

Table 12-8 Route Map Set Actions

Match Command	Description
set as-path prepend {*as-number-pattern* | last-as *1-10*}	Prepends the AS path for the network prefix with the pattern specified or from multiple iterations from a neighboring AS.
set ip next-hop { *ip-address* | peer-address | self }	Sets the next-hop IP address for any matching prefix. BGP dynamic manipulation uses the peer-address or self keywords.
set local-preference *0-4294967295*	Sets the BGP PA local preference.
set metric {+*value* | -*value* | *value*} (where value parameters are 0–4294967295)	Modifies the existing metric or sets the metric for a route.
set origin {igp | incomplete}	Sets the BGP PA origin.
set tag *tag-value*	Sets a numeric tag (0–4294967295) for identification of networks by other routers
set weight *0-65535*	Set the BGP PA weight.

Route Map continue Keyword

The route map is processed in order, and upon the first match, it executes the processing action, performs any optional action (if feasible), and stops processing. This prevents multiple route map sequences from processing.

Adding the keyword **continue** to a route map allows the route map to continue processing other route map sequences.

Example 12-7: The network prefix 192.168.1.1 matches in sequences 10, 20, and 30. Because the keyword continue was added to sequence 10, sequence 20 processes, but sequence 30 does not because a continue command was not present in sequence 20. The 192.168.1.1 prefix is permitted, and it is modified so that the metric is 20, with the next-hop address 10.12.1.1.

Example 12-7 *Route Map with the* continue *Keyword*

```
ip access-list standard ACL-ONE
 permit 192.168.1.1 0.0.0.0
 permit 172.16.0.0 0.0.255.255
 !
ip access-list standard ACL-TWO
 permit 192.168.1.1 0.0.0.0
 permit 172.31.0.0 0.0.255.255
!
route-map EXAMPLE permit 10
 match ip address ACL-ONE
 set metric 20
 continue
!
route-map EXAMPLE permit 20
 match ip address ACL-TWO
 set ip next-hop 10.12.1.1
!
route-map EXAMPLE permit 30
 set ip next-hop 10.13.1.3
```

Note: The continue command is not commonly used because it adds complexity when troubleshooting route maps.

BGP Route Filtering and Manipulation

- Route filtering is a method of selectively identifying routes that are advertised or received from neighbor routers.
- Route filtering may be used to manipulate traffic flows, reduce memory utilization, or improve security.

BGP Route Filtering Concepts

Route filtering selectively identifies routes that are advertised or received from neighbor routers. Route filtering may be used to manipulate traffic flows, reduce memory utilization, or improve security.

ISPs commonly deploy route filters on BGP peerings to customers. Ensuring that only the customer routes are allowed over the peering link prevents the customer from accidentally becoming a transit AS on the internet.

IOS XE has four methods of filtering routes inbound or outbound for a specific BGP peer:

- **Distribute list -** This filters network prefixes based on a standard or extended ACL. An implicit deny implied for any prefix not permitted.
- **Prefix list -** The prefix-matching specifications permit or deny network prefixes in a top-down fashion. An implicit deny for any prefix not permitted.
- **AS path ACL/filtering -** A list of regex commands allow for the permit or deny of a network prefix based on the current AS path values. An implicit deny for any prefix not permitted.
- **Route maps -** These provide a method of conditional matching on a variety of prefix attributes and taking a variety of actions. Actions could be a simple permit or deny; or could include the modification of BGP path attributes. An implicit deny for any prefix that is not permitted.

BGP Route Filtering Concepts (begin routing table reference)

The following slides explain each of the route filtering techniques in more detail. Imagine a simple scenario with R1 (AS 65100) that has a single eBGP peering with R2 (AS 65200), which then may peer with other autonomous systems (such as AS 65300). The relevant portion of the topology is that R1 peers with R2 and focuses on R1's BGP table, as shown in Example 12-8, with an emphasis on the network prefix and the AS path.

Example 12-8 *Reference BGP Table*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.3.3.0/24      10.12.1.2               33             0 65200 65300 3003 ?
 *   10.12.1.0/24     10.12.1.2               22             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2              333             0 65200 ?
 *>  100.64.2.0/25    10.12.1.2               22             0 65200 ?
 *>  100.64.2.192/26  10.12.1.2               22             0 65200 ?
 *>  100.64.3.0/25    10.12.1.2               22             0 65200 65300 300 ?
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   10.12.1.2               22             0 65200 ?
 *>  192.168.3.3/32   10.12.1.2             3333             0 65200 65300 ?
```

BGP Distribute List Filtering

Distribute lists allow the filtering of network prefixes on a neighbor-by-neighbor basis, using standard or extended ACLs. Configuring a distribute list requires using the BGP address family configuration command **neighbor ip-address distribute-list {*acl-number* | *acl-name*} {in|out}.**

Remember that extended ACLs for BGP use the source fields to match the network portion and the destination fields to match against the network mask.

Example 12-9 *BGP Distribute List Configuration*

```
R1
ip access-list extended ACL-ALLOW
 permit ip 192.168.0.0 0.0.255.255 host 255.255.255.255
 permit ip 100.64.0.0 0.0.255.0 host 255.255.255.128
!
router bgp 65100
 address-family ipv4
  neighbor 10.12.1.2 distribute-list ACL-ALLOW in
```

BGP Distribute List Filtering (routing table result)

Example 12-10 displays the routing table of R1 after BGP distribute list filtering.

Two local routes are injected into the BGP table by R1 (10.12.1.0/24 and 192.168.1.1/32). The two loopback networks from R2 (AS 65200) and R3 (AS 65300) are allowed because they are within the first ACL-ALLOW entry, and two of the networks in the 100.64.x.0 pattern (100.64.2.0/25 and 100.64.3.0/25) are accepted. The 100.64.2.192/26 network is rejected because the prefix length does not match the second ACL-ALLOW entry.

Example 12-10 *Viewing Routes Filtered by BGP Distribute List*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.12.1.0/24     0.0.0.0                  0         32768 ?
 *>  100.64.2.0/25    10.12.1.2               22             0 65200 ?
 *>  100.64.3.0/25    10.12.1.2               22             0 65200 65300 300 ?
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   10.12.1.2               22             0 65200 ?
 *>  192.168.3.3/32   10.12.1.2             3333             0 65200 65300 ?
```

BGP Prefix List Filtering and Routing Table Result

Prefix lists allow the filtering of network prefixes on a neighbor-by-neighbor basis, using a prefix list. Configuring a prefix list involves using the BGP address family configuration command **neighbor** *ip-address* **prefix-list** *prefix-list-name* **{in | out}.**

Example 12-11 demonstrates the use of a prefix list filter to allow only routes within the RFC 1918 space. The prefix-list is applied on R1's peering to R2 (AS 65200).

The BGP table can be examined on R1, as shown in Example 12-12. Notice that the 100.64.2.0/25, 100.64.2.192/26, and 100.64.3.0/25 networks were filtered as they did not fall within the prefix list matching criteria.

Example 12-11 *Prefix List Filtering Configuration*

```
R1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)# ip prefix-list RFC1918 seq 5 permit 192.168.0.0/13 ge 32
R1(config)# ip prefix-list RFC1918 seq 10 deny 0.0.0.0/0 ge 32
R1(config)# ip prefix-list RFC1918 seq 15 permit 10.0.0.0/7 ge 8
R1(config)# ip prefix-list RFC1918 seq 20 permit 172.16.0.0/11 ge 12
R1(config)# ip prefix-list RFC1918 seq 25 permit 192.168.0.0/15 ge 16
R1(config)# router bgp 65100
R1(config-router)# address-family ipv4 unicast
R1(config-router-af)# neighbor 10.12.1.2 prefix-list RFC1918 in
```

Example 12-12 *Verification of Filtering with a BGP Prefix List*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.3.3.0/24      10.12.1.2               33             0 65200 65300 3003 ?
 *   10.12.1.0/24     10.12.1.2               22             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2              333             0 65200 ?
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   10.12.1.2               22             0 65200 ?
 *>  192.168.3.3/32   10.12.1.2             3333             0 65200 65300 ?
```

BGP AS Path ACL Filtering

Selecting routes from a BGP neighbor by using the AS path requires the definition of an AS path access control list (AS path ACL).

Regular expressions, introduced earlier in this chapter, are a component of AS_Path filtering.

Example 12-13 shows the routes that R2 (AS 65200) is advertising toward R1 (AS 65100).

Example 12-13 *AS Path Access List Configuration*

```
R2# show bgp ipv4 unicast neighbors 10.12.1.1 advertised-routes | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.3.3.0/24      10.23.1.3                33             0 65300 3003 ?
 *>  10.12.1.0/24     0.0.0.0                   0         32768 ?
 *>  10.23.1.0/24     0.0.0.0                   0         32768 ?
 *>  100.64.2.0/25    0.0.0.0                   0         32768 ?
 *>  100.64.2.192/26  0.0.0.0                   0         32768 ?
 *>  100.64.3.0/25    10.23.1.3                 3             0 65300 300 ?
 *>  192.168.2.2/32   0.0.0.0                   0         32768 ?
 *>  192.168.3.3/32   10.23.1.3                333             0 65300 ?

Total number of prefixes 8
```

R2 is advertising the routes learned from R3 (AS 65300) to R1. In essence, R2 provides transit connectivity between the autonomous systems. If this were an Internet connection and R2 were an enterprise, it would not want to advertise routes learned from other ASs.

Using an AS path access list to restrict the advertisement of only AS 65200 routes is recommended.

BGP AS Path ACL Filtering (Cont.)

IOS supports up to 500 AS path ACLs and uses the command **ip as-path access-list** *acl-number* **{deny | permit}** *regex-query* for creating an AS path ACL. The ACL is then applied with the command **neighbor** *ip-address* **filter-list** *acl-number* **{in|out}.**

Example 12-14 shows the configuration on R2 using an AS path ACL to restrict traffic to only locally originated traffic, using the regex pattern ^$ to ensure completeness, the AS path ACL is applied on all eBGP neighborships.

Example 12-14 *AS Path Access List Configuration*

```
R2
ip as-path access-list 1 permit ^$
!
router bgp 65200
 address-family ipv4 unicast
  neighbor 10.12.1.1 filter-list 1 out
  neighbor 10.23.1.3 filter-list 1 out
```

Example 12-15 *Verification of Local Route Advertisements with an AS Path ACL*

```
R2# show bgp ipv4 unicast neighbors 10.12.1.1 advertised-routes | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.12.1.0/24     0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     0.0.0.0                  0         32768 ?
 *>  100.64.2.0/25    0.0.0.0                  0         32768 ?
 *>  100.64.2.192/26  0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   0.0.0.0                  0         32768 ?

Total number of prefixes 5
```

Example 12-15 displays the routes being advertised to R1. Notice that all the routes do not have an AS path, confirming that only locally originating routes are being advertised externally.

BGP Route Map Filtering

Route maps provide additional functionality over pure filtering. Route maps can manipulate BGP path attributes as well. Route maps are applied on a BGP neighbor for routes that are advertised or received. A different route map can be used for each direction. The route map is associated with the BGP neighbor under the specific address family, with the command **neighbor** *ip-address* **route-map** *route-map-name* **{in|out}.**

Example 12-16 shows the BGP routing table of R1, which is used here to demonstrate the power of a route map.

Example 12-16 *BGP Table Before Applying a Route Map*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      0.0.0.0                  0         32768 ?
 *>  10.3.3.0/24      10.12.1.2               33             0 65200 65300 3003 ?
 *   10.12.1.0/24     10.12.1.2               22             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2              333             0 65200 ?
 *>  100.64.2.0/25    10.12.1.2               22             0 65200 ?
 *>  100.64.2.192/26  10.12.1.2               22             0 65200 ?
 *>  100.64.3.0/25    10.12.1.2               22             0 65200 65300 300 ?
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   10.12.1.2               22             0 65200 ?
 *>  192.168.3.3/32   10.12.1.2             3333             0 65200 65300 ?
```

BGP Route Map Filtering (Cont.)

This route map consists of four steps:

1. Deny any routes that are in the 192.168.0.0/16 network by using a prefix list.

2. Match any routes originating from AS 65200 that are within the 100.64.0.0/10 network range and set the BGP local preference to 222.

3. Match any routes originating from AS 65200 that did not match step 2 and set the BGP weight to 65200.

4. Permit all other routes to process.

Example 12-17 demonstrates R1's configuration, where multiple prefix lists are referenced along with an AS path ACL.

Example 12-17 *R1's Route Map Configuration for Inbound AS 65200 Routes*

```
R1
ip prefix-list FIRST-RFC1918 permit  192.168.0.0/15 ge 16
ip as-path access-list 1 permit _65200$
ip prefix-list SECOND-CGNAT permit 100.64.0.0/10 ge 11
!
route-map AS65200IN deny 10
 description Deny any RFC1918 networks via Prefix List Matching
 match ip address prefix-list FIRST-RFC1918
!
route-map AS65200IN permit 20
 description Change local preference for AS65200 originate route in 100.64.x.x/10
 match ip address prefix-list SECOND-CGNAT
 match as-path 1
 set local-preference 222
!
route-map AS65200IN permit 30
 description Change the weight for AS65200 originate routes
 match as-path 1
 set weight 65200
!
route-map AS65200IN permit 40
 description Permit all other routes un-modified
!
router bgp 65100
 address-family ipv4 unicast
  neighbor 10.12.1.1 route-map AS65200IN in
```

BGP Route Map Filtering (routing table result)

Example 12-18 displays R1's BGP routing table. The following actions have occurred:

- The 192.168.2.2/32 and 192.168.3.3/32 routes were discarded. The 192.168.1.1/32 route is a locally generated route.
- The 100.64.2.0/25 and 100.64.2.192/26 networks had the local preference modified to 222 because they originated from AS 65200 and are within the 100.64.0.0/10 network range.
- The 10.12.1.0/24 and 10.23.1.0/24 routes from R2 were assigned the locally significant BGP attribute weight 65200.
- All other routes were received and not modified.

Example 12-18 *Verifying Changes from R1's Route Map to AS 65200*

```
R1# show bgp ipv4 unicast | b Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      0.0.0.0                  0         32768 ?
 *>  10.3.3.0/24      10.12.1.2               33             0 65200 65300 3003 ?
 r>  10.12.1.0/24     10.12.1.2               22         65200 65200 ?
 r                    0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2              333         65200 65200 ?
 *>  100.64.2.0/25    10.12.1.2               22    222      0 65200 ?
 *>  100.64.2.192/26  10.12.1.2               22    222      0 65200 ?
 *>  100.64.3.0/25    10.12.1.2               22             0 65200 65300 300 ?
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
```

It is considered a best practice to use a different route policy for inbound and outbound prefixes for each BGP neighbor.

Clearing BGP Connections

Depending on the change to the BGP route manipulation technique, a BGP session may need to be refreshed in order to take effect.

BGP supports two methods of clearing a BGP session:

1. Hard reset tears down the BGP session, removes BGP routes from the peer, and is the most disruptive.
2. Soft reset invalidates the BGP cache and requests a full advertisement from its BGP peer.

Routers initiate a hard reset with the command: **clear ip bgp** *ip-address* **[soft]** and a soft reset by using the optional **soft** keyword. All of a router's BGP sessions can be cleared by using an asterisk * in lieu of the peer's IP address.

Soft resets can be performed for a specific address family with the command **clear bgp** *afi safi* **{***ip-address***|*} soft [in | out]**

Soft resets reduce the number of routes that must be exchanged if multiple address families are configured with a single BGP peer.

BGP Communities

- BGP communities provide additional capability for tagging routes and for modifying BGP routing policy on upstream and downstream routers.
- BGP communities can be appended, removed, or modified selectively on each attribute as a route travels from router to router.

BGP Communities

BGP communities are an optional transitive BGP attribute that can traverse from AS to AS.

A BGP community is a 32-bit number that can be included with a route.

A BGP community can be displayed as a full 32-bit number (0–4,294,967,295) or as two 16-bit numbers (0–65535):(0–65535), commonly referred to as new format.

Private BGP communities follow a particular convention where the first 16 bits represent the AS of the community origination, and the second 16 bits represent a pattern defined by the originating AS.

In 2006, RFC 4360 expanded BGP communities' capabilities by providing an extended format. Extended BGP communities provide structure for various classes of information and are commonly used for VPN services. RFC 8092 provides support for communities larger than 32 bits (which are beyond the scope of this course).

Well-Known Communities

Well-known communities are implemented by all routers that are capable of sending/receiving BGP communities.

Three common well-known communities:

Internet: This is a standardized community for identifying routes that should be advertised on the internet. In larger networks that deploy BGP into the core, advertised routes should be advertised to the Internet and should have this community set. This allows for the edge BGP routers to only allow the advertisement of BGP routes with the internet community to the Internet. Filtering is not automatic but can be done with an outbound route map.

No_Advertise: Routes with this community should not be advertised to any BGP peer (iBGP or eBGP).

No_Export: When a route with this community is received, the route is not advertised to any eBGP peer. Routes with this community can be advertised to iBGP peers.

Enabling BGP Community Support

IOS and IOS XE routers do not advertise BGP communities to peers by default. Communities are enabled on a neighbor-by-neighbor basis with the BGP address family configuration command. If a keyword is not specified, standard communities are sent by default.:

neighbor *ip-address* **send-community [standard | extended | both]**

IOS XE nodes can display communities in new format, with the global configuration command:

ip bgp-community new-format

Example 12-19 displays the BGP community in decimal format first, followed by the new format.

Example 12-19 *BGP Community Formats*

```
! Decimal Format
R3# show bgp 192.168.1.1
! Output omitted for brevity
BGP routing table entry for 192.168.1.1/32, version 6
Community: 6553602 6577023
```

```
! New-Format
R3# show bgp 192.168.1.1
! Output omitted for brevity
BGP routing table entry for 192.168.1.1/32, version 6
Community: 100:2 100:23423
```

BGP Community List - Conditional Matching

Conditionally matching requires the creation of a community list with a similar structure to an ACL.

Standard community lists are numbered 1 to 99 and match either well-known communities or a private community number (as-number:16-bit-number).

Expanded community lists are numbered 100 to 500 and use regex patterns.

The configuration syntax for a community list is: **ip community-list {*1-500* | standard *list-name* | expanded *list-name*} {permit | deny} *community-pattern***

Example 12-23 creates a BGP community list 100 that matches on the community 333:333. Then it is used in the first sequence of *route-map* COMMUNITY-CHECK, which denies any routes with that community. The second route map sequence allows for all other BGP routes and sets the BGP weight (locally significant) to 111. The route map is then applied on routes advertised from R2 toward R1.

Example 12-23 *Conditionally Matching BGP Communities*

```
R1
ip community-list 100 permit 333:333
!
route-map COMMUNITY-CHECK deny 10
 description Block Routes with Community 333:333 in it
 match community 100
route-map COMMUNITY-CHECK permit 20
 description Allow routes with either community in it
 set weight 111
!
router bgp 65100
 address-family ipv4 unicast
  neighbor 10.12.1.2 route-map COMMUNITY-CHECK in
```

BGP Community List - Conditional Matching (routing table result)

Example 12-24 shows the BGP table after the route map has been applied to the neighbor. The 10.23.1.0/24 network prefix was discarded, and all the other routes learned from AS 65200 had the BGP weight set to 111.

Example 12-24 *R1's BGP Table After Applying the Route Map*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      0.0.0.0                  0         32768 ?
 *   10.12.1.0/24     10.12.1.2               22           111 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   10.12.1.2               22           111 65200 ?
 *>  192.168.3.3/32   10.12.1.2             3333           111 65200 65300 ?
```

Setting Private BGP Communities

A private BGP community is set in a route map with the command **set community *bgp-community* [additive]**.

By default, when setting a community, any existing communities are over-written but can be preserved by using the optional **additive** keyword.

Example 12-25 shows the BGP table entries for the 10.23.1.0/24 network, which has the 333:333 and 65300:333 BGP communities. The 10.3.3.0/24 network has the 65300:300 community.

Example 12-25 *Viewing the BGP Communities for Two Network Prefixes*

```
R1# show bgp ipv4 unicast 10.23.1.0/24
! Output omitted for brevity
BGP routing table entry for 10.23.1.0/24, version 15
  65200
    10.12.1.2 from 10.12.1.2 (192.168.2.2)
      Origin incomplete, metric 333, localpref 100, valid, external, best
      Community: 333:333 65300:333
```

```
R1# show bgp ipv4 unicast 10.3.3.0/24
! Output omitted for brevity
BGP routing table entry for 10.3.3.0/24, version 12
  65200 65300 3003
    10.12.1.2 from 10.12.1.2 (192.168.2.2)
      Origin incomplete, metric 33, localpref 100, valid, external, best
      Community: 65300:300
```

Setting Private BGP Communities (cont.)

Example 12-26 shows the configuration where the BGP community is set to the 10.23.1.0/24 network. The additive keyword is not used, so the previous community values 333:333 and 65300:333 are overwritten with the 10:23 community. The 10.3.3.0/24 network has the communities 3:0, 3:3, and 10:10 added to the existing communities. The route map is then associated to R2 (AS 65200).

Example 12-27 shows that after the route map has been applied and the routes have been refreshed, the path attributes can be examined. As anticipated, the previous BGP communities were removed for the 10.23.1.0/24 network but were maintained for the 10.3.3.0/24 network.

Example 12-26 *Setting Private BGP Community Configuration*

```
ip prefix-list PREFIX10.23.1.0 seq 5 permit 10.23.1.0/24
ip prefix-list PREFIX10.3.3.0 seq 5 permit 10.3.3.0/24
!
route-map SET-COMMUNITY permit 10
 match ip address prefix-list PREFIX10.23.1.0
 set community 10:23
route-map SET-COMMUNITY permit 20
 match ip address prefix-list PREFIX10.3.3.0
 set community 3:0 3:3 10:10 additive
route-map SET-COMMUNITY permit 30
!
router bgp 65100
 address-family ipv4
  neighbor 10.12.1.2 route-map SET-COMMUNITY in
```

Example 12-27 *Verifying BGP Community Changes*

```
R1# show bgp ipv4 unicast 10.23.1.0/24
! Output omitted for brevity
BGP routing table entry for 10.23.1.0/24, version 22
  65200
    10.12.1.2 from 10.12.1.2 (192.168.2.2)
      Origin incomplete, metric 333, localpref 100, valid, external, best
      Community: 10:23
```

```
R1# show bgp ipv4 unicast 10.3.3.0/24
BGP routing table entry for 10.3.3.0/24, version 20
  65200 65300 3003
    10.12.1.2 from 10.12.1.2 (192.168.2.2)
      Origin incomplete, metric 33, localpref 100, valid, external, best
      Community: 3:0 3:3 10:10 65300:300
```

Understanding BGP Path Selection

- The BGP best-path selection algorithm influences how traffic enters or leaves an AS.
- Some router configurations modify the BGP attributes to influence inbound traffic, outbound traffic, or inbound and outbound traffic, depending on the network design requirements.
- This section explains the logic used by a router that uses BGP when forwarding packets.

Routing Path Selections Using Longest Match

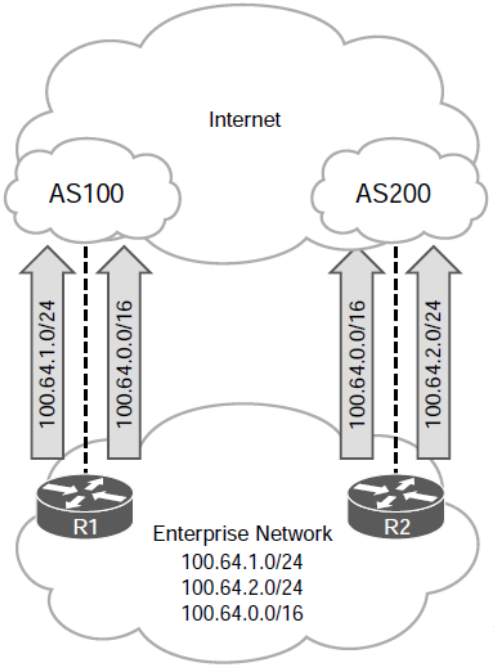
Routers always select the path by examining the prefix length of a network entry. The path selected is chosen where the longest prefix length is always preferred.

This logic can be used to influence path selection in BGP. Assume that an organization owns the 100.64.0.0/16 network range but only needs to advertise two subnets (100.64.1.0/24 and 100.64.2.0/24). It could advertise both prefixes (100.64.1.0/24 and 100.64.2.0/24) from all its routers, but how can it distribute the load for each subnet if all traffic comes in on one router (such as R1)?

The organization could modify various BGP path attributes (PAs) that are advertised externally, but an SP could have a BGP routing policy that ignores path attributes, resulting in random receipt of network traffic.

A way that guarantees that paths are selected deterministically outside the organization is to advertise a summary prefix (100.64.0.0/16) out both routers. Then the organization can advertise a longer matching prefix out the router that should receive network traffic for that prefix.

The Figure shows R1 advertising the 100.64.1.0/24 prefix, R2 advertising the 100.64.2.0/24 prefix, and both routers advertising the 100.64.0.0/16 summary prefix.



BGP Best-Path Algorithm

In BGP, route advertisements consist of Network Layer Reachability Information (NLRI) and path attributes (PAs). The NLRI consists of the network prefix and prefix length, and the BGP attributes such as AS_Path, origin, and so on are stored in the PAs.

A BGP route may contain multiple paths to the same destination network. Every path's attributes impact the desirability of the route. A BGP router advertises only the best path to the neighboring routers.

The best path is installed in the RIB. If the best path is no longer available, the router can use the existing paths to identify a new best path. BGP recalculates the best path for a prefix upon four possible events:

- **BGP next-hop reachability change**
- **Failure of an interface connected to an eBGP peer**
- **Redistribution change**
- **Reception of new or removed paths for a route**

BGP automatically installs the first received path as the best path. When additional paths are received for the same network prefix length, the newer paths are compared against the current best path. If there is a tie, processing continues until a best-path winner is identified.

BGP Best-Path Algorithm

The BGP best-path algorithm uses the following attributes, in the order shown, for best-path selection:

1. **Weight** – BGP weight is a Cisco-defined attribute. The path with the higher weight is preferred. Influences only outbound routes from a router or an AS. Not advertised to peers.
2. **Local preference** – (LOCAL_PREF) is a well-known path attribute included with path advertisements within the AS. Not advertised between eBGP peers.
3. **Local originated** (network statement, redistribution, or aggregation) – determination as to whether the route originated within the AS. Preference is given to routes advertised or aggregated locally.
4. **Accumulated Interior Gateway Protocol (AIGP)** – provides the ability for BGP to make routing decisions based on IGP path metrics.
5. **Shortest AS_Path –** AS path length typically correlates to the AS hop count. A shorter AS path is preferred over a longer AS path.
6. **Origin type** - The next best-path decision factor is the well-known mandatory BGP attribute named *origin*. Routes with IGP origin are preferred over those with EGP or incomplete origin (least preferred).

BGP Best-Path Algorithm (Cont.)

7. **Lowest MED -** The next BGP best-path decision factor is the non-transitive BGP attribute named *multiple-exit discriminator* (MED). A lower MED is preferred over a higher MED.

8. **eBGP over iBGP** – The best path selection route source preference order is: eBGP peers (most desirable), confederation member AS peers, and iBGP peers (less desirable)..

9. **Lowest IGP next hop-** The next decision step is to use the lowest IGP cost to the BGP next-hop address.

10. **Oldest eBGP Path** - BGP maintains stability in a network by preferring the path from the oldest (established) BGP session. The downfall of this technique is that it does not lead to a deterministic method of identifying the BGP best path from a design perspective.

11. **Router ID** - Prefer the route that comes from the BGP peer with the lower router ID (RID)

12. **Minimum Cluster List Length** - Prefer the route with the minimum cluster list length. In simplest terms, this step locates the path that has traveled the lowest number of iBGP advertisement hops.

13. **Lowest neighbor address -** The last step is to use the path that comes from the lowest BGP neighbor address. This step is limited to iBGP peerings because eBGP peerings used the oldest received path as the tie breaker.

Weight Attribute (Cisco-defined)

Weight - BGP weight is a Cisco-defined attribute and the first step for selecting the BGP best path. Weight is a 16-bit value (0 to 65,535) assigned locally on the router; it is not advertised to other routers.

The path with the higher weight is preferred. Weight can be set for specific routes with an inbound route map or for all routes learned from a specific neighbor.

Weight is not advertised to peers and only influences outbound traffic from a router or an AS. Because it is the first step in the best-path algorithm, it should be used when other attributes should not influence the best path for a specific network.

Example 12-28 *An Example of a BGP Best-Path Choice Based on Weight*

```
R2# show bgp ipv4 unicast 172.16.1.0/24
BGP routing table entry for 172.16.1.0/24, version 3
Paths: (2 available, best #1, table default)
  Refresh Epoch 2
  65300
    10.23.1.3 from 10.23.1.3 (192.18.3.3)
      Origin IGP, metric 0, localpref 100, weight 123, valid, external, best
  Refresh Epoch 2
  65100
    10.12.1.1 from 10.12.1.1 (192.168.1.1)
      Origin IGP, metric 0, localpref 100, valid, external
```

Local Preference Attribute

Local Preference - (LOCAL_PREF) is a discretionary path attribute included with path advertisements throughout an AS. The local preference attribute is a 32-bit value (0 to 4,294,967,295) that indicates the preference for exiting the AS to the destination network.

The local preference is not advertised between eBGP peers and is typically used to influence the next-hop address for outbound traffic leaving an autonomous system.

A higher value is preferred over a lower value. The default local preference value of 100 is used during best-path calculation, and is included in advertisements to other iBGP peers.

Local preference can influence path selection on other iBGP peers without impacting eBGP peers because local preference is not advertised outside the autonomous system.

Example 12-29 shows the BGP table for the 172.16.1.0/24 network prefix on R2. On the third line of the output, the router indicates that two paths exist, and the first path is the best path. The BGP weight does not exist, so then the local preference is used. The path learned through AS 65300 is the best path because it has a local preference of 333.

Example 12-29 *An Example of a BGP Best-Path Choice Based on Local Preference*

```
R2# show bgp ipv4 unicast 172.16.1.0/24
BGP routing table entry for 172.16.1.0/24, version 4
Paths: (2 available, best #1, table default)
  Advertised to update-groups:
     2
  Refresh Epoch 4
  65300
    10.23.1.3 from 10.23.1.3 (192.18.3.3)
      Origin IGP, metric 0, localpref 333, valid, external, best
  Refresh Epoch 4
  65100
    10.12.1.1 from 10.12.1.1 (192.168.1.1)
      Origin IGP, metric 0, localpref 111, valid, external
```

Locally Originated Attribute

The third decision point in the best-path algorithm is to determine whether the route **originated locally**.

Preference is given in the following order:

1. **Routes that were advertised locally**
2. **Networks that have been aggregated locally**
3. **Routes received by BGP peers**

Accumulated Interior Gateway Protocol (AIGP) Attribute

Accumulated Interior Gateway Protocol (AIGP) - is an optional nontransitive path attribute that is included with advertisements throughout an AS.

BGP does not use path metric due to scalability issues combined with the notion that each AS may use a different routing policy to calculate metrics. The ability for BGP to make routing decisions based on a path metric is a viable option because all the ASs are under the control of a single domain, with consistent routing policies for BGP and IGPs. IGPs are redistributed into BGP.

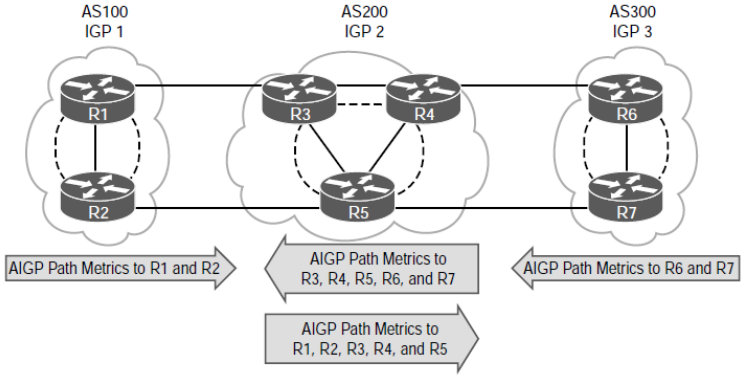


Figure 12-11 *AIGP Path Attribute Exchange Between Autonomous Systems*

- A path with an AIGP metric is preferred to a path without an AIGP metric.
- If the next-hop address requires a recursive lookup, the AIGP path needs to calculate a derived metric to include the distance to the next-hop address. The formula is: Derived AIGP metric = (Original AIGP metric + Next-hop AIGRP metric)
- If multiple AIGP paths exist and one next-hop address contains an AIGP metric and the other does not, the non-AIGP path is not used.
- The next-hop AIGP metric is recursively added if multiple lookups are performed.

Shortest AS Path Attribute

The path length typically correlates to the AS hop count. A shorter AS path is preferred over a longer AS path.

Prepending ASNs to the AS path makes it longer, thereby making that path less desirable compared to other paths. Typically, the AS path is prepended with the network owner's ASN.

In general, a path that has had the AS path prepended is not selected as the BGP best path because the AS path is longer than the non-prepended path advertisement.

Inbound traffic is influenced by prepending AS path length in advertisements to other ASs, and outbound traffic is influenced by prepending advertisements received from other ASs.

Example 12-30 shows the BGP table for the 172.16.1.0/24 network prefix on R2. The second route learned through AS 65100 is the best path. There is not a weight set on either path, and the local preference is identical. The second path has an AS path length of 1, while the first path has an AS path length of 2 (65300 and 65300).

Example 12-30 *An Example of a BGP Best-Path Choice Based on AS Path Length*

```
R2# show bgp ipv4 unicast 172.16.1.0/24
BGP routing table entry for 172.16.1.0/24, version 6
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
     2
  Refresh Epoch 8
  65300 65300
    10.23.1.3 from 10.23.1.3 (192.18.3.3)
      Origin IGP, metric 0, localpref 100, valid, external
  Refresh Epoch 8
  65100
    10.12.1.1 from 10.12.1.1 (192.168.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
```

Origin Type Path Attribute

The next BGP best-path decision factor is the well-known mandatory BGP attribute named **origin**.

By default, networks that are advertised through the network statement are set with the IGP or i origin, and redistributed networks are assigned the Incomplete or ? origin attribute.

The origin preference order is:

1. **IGP origin (most)**
2. **EGP origin**
3. **Incomplete origin (least)**

Example 12-31 *An Example of a BGP Best-Path Choice Based on Origin Type*

```
R2# show bgp ipv4 unicast 172.16.1.0/24
BGP routing table entry for 172.16.1.0/24, version 6
Paths: (2 available, best #2, table default)
  Advertised to update-groups:
     2
  Refresh Epoch 10
  65300
    10.23.1.3 from 10.23.1.3 (192.18.3.3)
      Origin incomplete, metric 0, localpref 100, valid, external
  Refresh Epoch 10
  65100
    10.12.1.1 from 10.12.1.1 (192.168.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
```

Example 12-31 shows the BGP table for the 172.16.1.0/24 network prefix on R2. The second path learned through AS 65100 is the best path because it has an origin of IGP, while first path has an origin of incomplete, which is the least preferred.

Multi-Exit Discriminator (MED) Path Attribute

Multiple-Exit discriminator (MED) - is a non-transitive BGP attribute. MED uses a 32-bit value called a metric. BGP sets the MED automatically to the IGP path metric during network advertisement or redistribution.

If the MED is received from an eBGP session, it can be advertised to other iBGP peers, but it should not be sent to other eBGP peers outside the AS that received it.

A lower MED is preferred over a higher MED. For MED to be an effective decision factor, the paths being decided upon must come from the same ASN.

RFC 4451 guidelines state that a prefix without a MED value should be given priority and, in essence, should be compared with a value of 0.

If the MED is missing from a prefix learned from an eBGP peer, devices use a MED of 0 for the best-path calculation. IOS routers advertise a MED of 0 to iBGP peers.

Example 12-32 *An Example of a BGP Best-Path Choice Based on MED*

```
R2# show bgp ipv4 unicast 172.16.1.0
BGP routing table entry for 172.16.1.0/24, version 9
Paths: (2 available, best #1, table default)
  Advertised to update-groups:
     2
  Refresh Epoch 4
  65300
    10.12.1.1 from 10.12.1.1 (192.168.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
  Refresh Epoch 14
  65300
    10.23.1.3 from 10.23.1.3 (192.18.3.3)
      Origin IGP, metric 33, localpref 100, valid, external
```

Example 12-32 shows the BGP table for the 172.16.1.0/24 network prefix on R2. Notice that R2 is peering only with AS 65300 for MED to be eligible for the best-path selection process. The first path has a MED of 0, and the second path has a MED of 33. The first path is preferred as the MED is lower.

eBGP over iBGP

The next BGP best-path decision factor is whether the route comes from an **iBGP, eBGP, or confederation member AS** (sub-AS) peering. The best-path selection order is:

1. **eBGP peers (most desirable)**
2. **Confederation member AS peers**
3. **iBGP peers (least desirable)**

Note: BGP confederations are beyond the scope of the CCNP and CCIE Enterprise Core

Lowest IGP Metric

The next decision step is to use the **lowest IGP cost** to the BGP next-hop address.

Figure 12-12 illustrates a topology where R2, R3, R4, and R5 are in AS 400.

AS 400 peers in a full mesh and establishes BGP sessions using Loopback 0 interfaces. R1 advertises the 172.16.0.0/24 network prefix to R2 and R4.

R3 prefers the path from R2 compared to the iBGP path from R4 because the metric to reach the next-hop address is lower.

R5 prefers the path from R4 compared to the iBGP path from R2 because the metric to reach the next-hop address is lower.

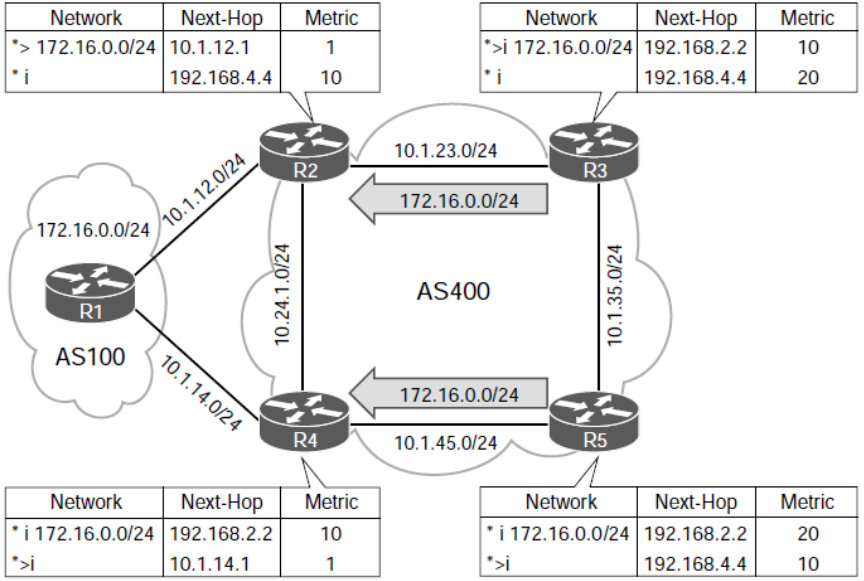


Figure 12-12 *Lowest IGP Metric Topology*

Oldest eBGP Path, Router ID, and Minimum Cluster List Length

Oldest eBGP Path - BGP can maintain large routing tables, and unstable sessions result in the BGP best-path calculation executing frequently. BGP maintains stability in a network by preferring the path from the oldest (established) BGP session. The downfall of this technique is that it does not lead to a deterministic method of identifying the BGP best path from a design perspective.

Router ID - The next step for the BGP best-path algorithm is to select the best path using the lowest router ID of the advertising eBGP router. If the route was received by a route reflector, then the originator ID is substituted for the router ID.

Minimum Cluster List Length - The next step in the BGP best-path algorithm is to select the best path using the lowest cluster list length. The cluster list is a non-transitive BGP attribute that is appended (not overwritten) by a route reflector with its cluster ID. Route reflectors use the cluster ID attribute as a loop-prevention mechanism. The cluster ID is not advertised between ASs and is locally significant. In simplest terms, this step locates the path that has traveled the lowest number of iBGP advertisement hops.

Lowest Neighbor Address

The last step of the BGP best-path algorithm is to select the path that comes from the lowest BGP neighbor address.

This step is limited to iBGP peerings because eBGP peerings use the oldest received path as the tie breaker.

Figure 12-13 demonstrates the concept of choosing the router with the lowest neighbor address. R1 is advertising the 172.16.0.0/24 network prefix to R2. R1 and R2 have established two BGP sessions using the 10.12.1.0/24 and 10.12.2.0/24 networks. R2 selects the path advertised from 10.12.1.1 as it is the lower IP address.

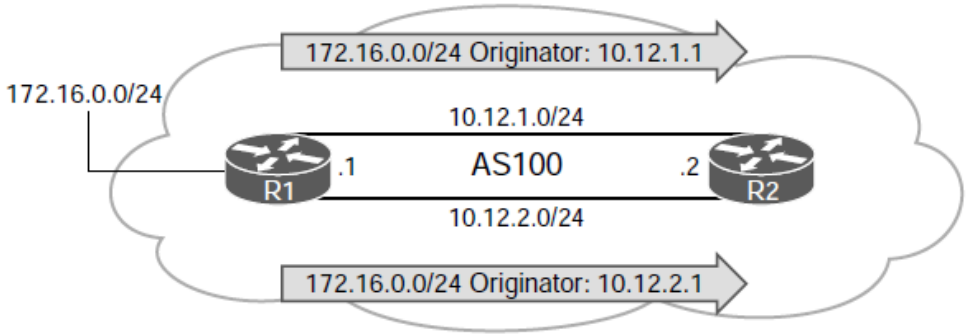


Figure 12-13 *Lowest IP Address*

Prepare for the Exam

Prepare for the Exam

Key Topics for Chapter 12

Description
Resiliency in service providers
Internet transit routing
Extended ACL IGP network selection
Extended ACL BGP network selection
Prefix match specifications
Prefix matching with length parameters
Prefix lists
Regular expressions
Route map components
Route map syntax and processing
Route map conditional matching

Prepare for the Exam

Key Topics for Chapter 12 (Cont.)

Description
Route map matching with multiple conditions
Route map optional actions
BGP distribute list filtering
BGP prefix list filtering
BGP AS path ACL/filtering
BGP route maps for neighbors
BGP communities
Enabling BGP community support
BGP community list
Setting private BGP communities
Routing path selection using longest match
BGP best-path algorithm

Prepare for the Exam

Key Terms for Chapter 12

Key Terms
AS path access control list (ACL)
BGP Community
BGP Multihoming
distribute list
prefix list
regular expression (regex)
route map
transit routing

Prepare for the Exam

Command Reference for Chapter 12

Task	Command Syntax
Configure a prefix list	**{ip | ipv6} prefix-list *prefix-list-name* [seq *sequence-number*] {permit | deny} *high-orderbit-pattern/high-order-bit-count* [ge *ge-value*][le *le-value*]**
Create a route map entry	**route-map *route-map-name* [permit | deny][*sequence-number*]**
Conditionally match in a route map by using the AS path	**match as-path *acl-number***
Conditionally match in a route map by using an ACL	**match ip address {*acl-number* | *acl-name*}**
Conditionally match in a route map by using a prefix list	**match ip address prefix-list *prefix-list-name***
Conditionally match in a route map by using a local preference	**match local-preference *local-preference***

Command Reference for Chapter 12 (Cont.)

Task	Command Syntax
Filter routes to a BGP neighbor by using an ACL	**neighbor *ip-address* distribute-list {*acl-number* | *acl-name*} {in|out}**
Filter routes to a BGP neighbor by using a prefix list	**neighbor *ip-address* prefix-list *prefix-list-name* {in | out}**
Create an ACL based on the BGP AS path	**ip as-path access-list *acl-number* {deny | permit} *regex-query***
Filter routes to a BGP neighbor by using an AS path ACL	**neighbor *ip-address* filter-list *acl-number* {in|out}**
Associate an inbound or outbound route map with a specific BGP neighbor	**neighbor *ip-address* route-map *route-map-name* {in|out}**
Configure IOS-based routers to display the community in new format for easier readability of BGP communities	**ip bgp-community new-format**

Command Reference for Chapter 12 (Cont.)

Task	Command Syntax
Create a BGP community list for conditional route matching	**ip community-list {*1-500* | standard *listname* | expanded *list-name*} {permit | deny} *community-pattern***
Set BGP communities in a route map	**set community *bgp-community* [additive]**
Initiate a route refresh for a specific BGP peer	**clear bgp *afi safi* {*ip-address*|*} soft [in | out]**
Display the current BGP table, based on routes that meet a specified AS path regex pattern	**show bgp *afi safi* regexp *regex-pattern***
Display the current BGP table, based on routes that meet a specified BGP community	**show bgp *afi safi* community *community***


CISCO