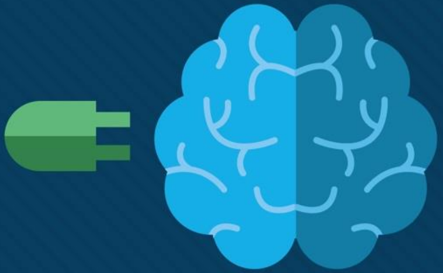


Chapter 12: Advanced BGP

Instructor Materials

CCNP Enterprise: Advanced Routing

Chapter 12 Content

This chapter covers the following content:

- **Route Summarization -** This section provides an overview of the how route summarization works with Border Gateway Protocol (BGP) and some design considerations related to summarization.
- **BGP Route Filtering and Manipulation -** This section demonstrates the filtering and manipulation of routes based on network prefix, AS_Path, or other BGP path attributes.
- **BGP Communities -** This section explains BGP communities and how the well-known communities influence prefix advertisements along with how they are used for conditional prefix filtering or manipulation.
- **Maximum Prefix -** This section explains how a router can limit the number of prefixes received to ensure that the BGP table does not exceed its capacity.
- **Configuration Scalability -** This section explains the use of peer groups and peer templates to assist with BGP configurations on routers with a lot of BGP sessions.

Route Summarization

- Summarizing prefixes conserves router resources and accelerates best-path calculation by reducing the size of the table.
- Summarization also provides the benefit of stability by hiding route flaps from downstream routers, thereby reducing routing churn.
- Route summarization can reduce the size of the BGP table for Internet routers.

Route Summarization

Aggregate Addresses

Dynamic BGP summarization consists of the configuration of an aggregate network prefix. When viable component routes that match the aggregate network prefix enter the BGP table, the aggregate prefix is created. The originating router sets the next hop to Null0 as a discard route for the aggregated prefix for loop prevention.

Dynamic route summarization is accomplished with the BGP address family configuration command **aggregate-address** *network subnet-mask* [**summary-only**] [**as-set**].

Figure 12-1 shows a simple topology in which R1 established an External BGP (eBGP) session with R2, and R2 establishes an eBGP session with R3.

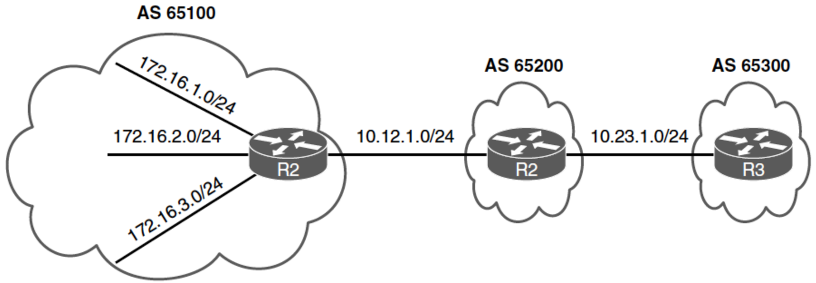


Figure 12-1 *BGP Summarization Topology*

Route Summarization

Route Aggregation

Example 12-1 demonstrates the BGP tables before route aggregation has been performed for R1.

Example 12-2 shows the route aggregation configuration for R1 & R2.

Example 12-3 shows the BGP tables on R1, after aggregation is configured.

Example 12-1 *BGP Tables for R1, R2, and R3 Without Aggregation*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *   10.12.1.0/24     10.12.1.2                0             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2                0             0 65200 ?
 *>  172.16.1.0/24    0.0.0.0                  0         32768 ?
 *>  172.16.2.0/24    0.0.0.0                  0         32768 ?
 *>  172.16.3.0/24    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                0             0 65200 ?
 *>  192.168.3.3/32   10.12.1.2                              0 65200 65300 ?
```

Example 12-2 *Configuring BGP Route Aggregation*

```
R1# show running-config | section router bgp
router bgp 65100
 bgp log-neighbor-changes
 aggregate-address 172.16.0.0 255.255.240.0
 redistribute connected
 neighbor 10.12.1.2 remote-as 65200
```

```
R2# show running-config | section router bgp
router bgp 65200
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 10.12.1.1 remote-as 65100
 neighbor 10.23.1.3 remote-as 65300
 !
 address-family ipv4
  aggregate-address 192.168.0.0 255.255.0.0
  redistribute connected
  neighbor 10.12.1.1 activate
  neighbor 10.23.1.3 activate
 exit-address-family
```

Example 12-3 *BGP Tables for R1, R2, and R3 with Aggregation*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *   10.12.1.0/24     10.12.1.2                0             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2                0             0 65200 ?
 *>  172.16.0.0/20    0.0.0.0                            32768 i
 *>  172.16.1.0/24    0.0.0.0                  0         32768 ?
 *>  172.16.2.0/24    0.0.0.0                  0         32768 ?
 *>  172.16.3.0/24    0.0.0.0                  0         32768 ?
 *>  192.168.0.0/16   10.12.1.2                0             0 65200 i
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
 *>  192.168.2.2/32   10.12.1.2                0             0 65200 ?
 *>  192.168.3.3/32   10.12.1.2                              0 65200 65300 ?
```

Route Summarization

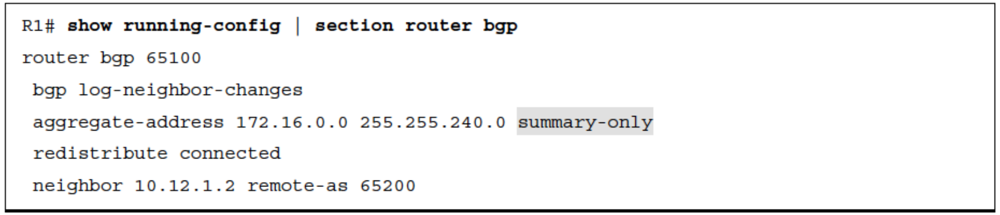
Route Aggregation Configuration with Suppression

The **aggregate-address** command advertises the aggregated network prefix in addition to the original component network prefixes. The optional **summary-only** keyword suppresses the component network prefixes in the summarized network prefix range.

Example 12-4 demonstrates the configuration with the summary-only keyword.

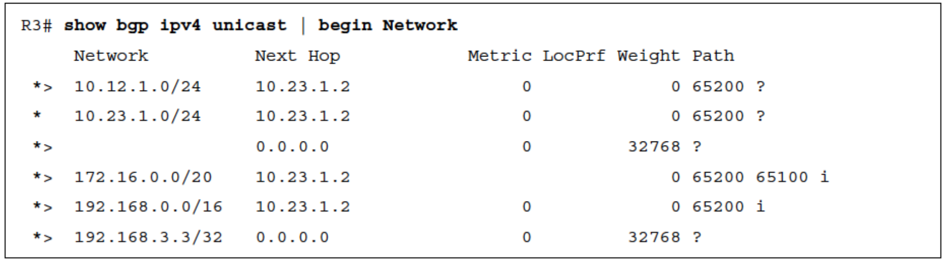
Example 12-5 shows the BGP table for R3 after the summary-only keyword is added to the aggregation command.

Example 12-4 *BGP Route Aggregation Configuration with Suppression*



```
R1# show running-config | section router bgp
router bgp 65100
 bgp log-neighbor-changes
 aggregate-address 172.16.0.0 255.255.240.0 summary-only
 redistribute connected
 neighbor 10.12.1.2 remote-as 65200
```

Example 12-5 *BGP Tables for R3 with Aggregation and Suppression*



```
R3# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.12.1.0/24     10.23.1.2                0             0 65200 ?
 *   10.23.1.0/24     10.23.1.2                0             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  172.16.0.0/20    10.23.1.2                              0 65200 65100 i
 *>  192.168.0.0/16   10.23.1.2                0             0 65200 i
 *>  192.168.3.3/32   0.0.0.0                  0         32768 ?
```

Route Summarization

Verify Route Aggregation with Suppression

Example 12-6 shows the BGP table and the Routing Information Base (RIB) for R2.

Example 12-7 shows that R1's stub networks are suppressed, and the summary discard route for the 172.16.0.0/20 is installed in the RIB as well.

Example 12-6 *R2's BGP and RIB After Aggregation with Suppression*

```
R2# show bgp ipv4 unicast
BGP table version is 10, local router ID is 192.168.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *   10.12.1.0/24     10.12.1.1                0             0 65100 ?
 *>                   0.0.0.0                  0         32768 ?
 *   10.23.1.0/24     10.23.1.3                0             0 65300 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  172.16.0.0/20    10.12.1.1                0             0 65100 i
 *>  192.168.0.0/16   0.0.0.0                            32768 i
 s>  192.168.1.1/32   10.12.1.1                0             0 65100 ?
 s>  192.168.2.2/32   0.0.0.0                  0         32768 ?
 s>  192.168.3.3/32   10.23.1.3                0             0 65300 ?
```

```
R2# show ip route bgp | begin Gateway
Gateway of last resort is not set

      172.16.0.0/20 is subnetted, 1 subnets
B        172.16.0.0 [20/0] via 10.12.1.1, 00:06:18
B     192.168.0.0/16 [200/0], 00:05:37, Null0
      192.168.1.0/32 is subnetted, 1 subnets
B        192.168.1.1 [20/0] via 10.12.1.1, 00:02:15
      192.168.3.0/32 is subnetted, 1 subnets
B        192.168.3.3 [20/0] via 10.23.1.3, 00:02:15
```

Example 12-7 *R1's BGP and RIB after Aggregation with Suppression*

```
R1# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *   10.12.1.0/24     10.12.1.2                0             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  10.23.1.0/24     10.12.1.2                0             0 65200 ?
 *>  172.16.0.0/20    0.0.0.0                            32768 i
 s>  172.16.1.0/24    0.0.0.0                  0         32768 ?
 s>  172.16.2.0/24    0.0.0.0                  0         32768 ?
 s>  172.16.3.0/24    0.0.0.0                  0         32768 ?
 *>  192.168.0.0/16   10.12.1.2                0             0 65200 i
 *>  192.168.1.1/32   0.0.0.0                  0         32768 ?
```

```
R1# show ip route bgp | begin Gateway
Gateway of last resort is not set
    10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
B        10.23.1.0/24 [20/0] via 10.12.1.2, 00:12:50
      172.16.0.0/16 is variably subnetted, 7 subnets, 3 masks
B        172.16.0.0/20 [200/0], 00:06:51, Null0
B     192.168.0.0/16 [20/0] via 10.12.1.2, 00:06:10
```

The Atomic Aggregate Attribute

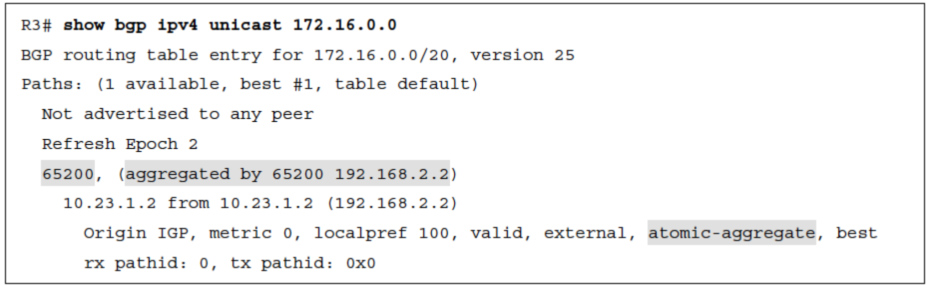
When a BGP router summarizes a route, it does not advertise the AS_Path information from before the aggregation. The atomic aggregate attribute indicates that a loss of path information has occurred.

Example 12-10 shows R3's BGP entry for the 172.16.0.0/20 network prefix.

The route's NLRI information indicates that the routes were aggregated by AS 65200 by the router with the router ID (RID) 192.168.2.2.

In addition, the atomic aggregate attribute has been set to indicate a loss of path attributes such as AS_Path in this scenario.

Example 12-10 *Examining the BGP Atomic Aggregate Attribute*



```
R3# show bgp ipv4 unicast 172.16.0.0
BGP routing table entry for 172.16.0.0/20, version 25
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 2
  65200, (aggregated by 65200 192.168.2.2)
    10.23.1.2 from 10.23.1.2 (192.168.2.2)
      Origin IGP, metric 0, localpref 100, valid, external, atomic-aggregate, best
      rx pathid: 0, tx pathid: 0x0
```

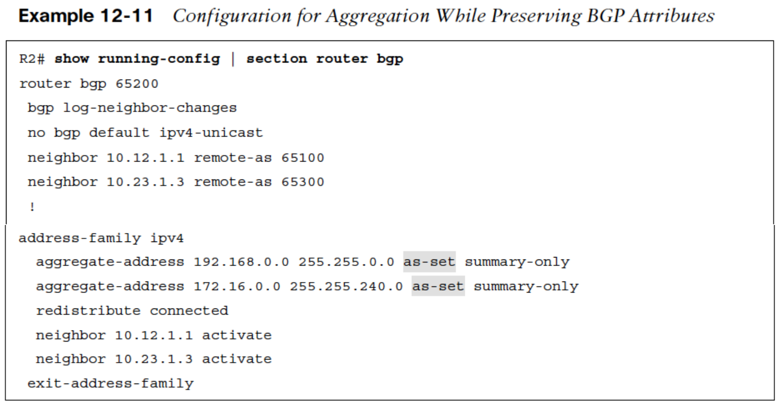
Route Summarization

Route Aggregation with AS_SET

To keep the BGP path information history, the optional as-set keyword may be used with the aggregate-address command. As the router generates the aggregate route, BGP attributes from the component aggregate routes are copied over to it.

The AS_Path settings from the original prefixes are stored in the AS_SET portion of AS_Path. AS_SET, which is displayed within brackets, counts as only one AS hop, even if multiple ASs are listed.

Example 12-11 shows R2's updated BGP configuration for summarizing both prefixes with the as-set keyword.


Example 12-11 *Configuration for Aggregation While Preserving BGP Attributes*

```
R2# show running-config | section router bgp
router bgp 65200
 bgp log-neighbor-changes
 no bgp default ipv4-unicast
 neighbor 10.12.1.1 remote-as 65100
 neighbor 10.23.1.3 remote-as 65300
 !
address-family ipv4
  aggregate-address 192.168.0.0 255.255.0.0 as-set summary-only
  aggregate-address 172.16.0.0 255.255.240.0 as-set summary-only
  redistribute connected
  neighbor 10.12.1.1 activate
  neighbor 10.23.1.3 activate
 exit-address-family
```

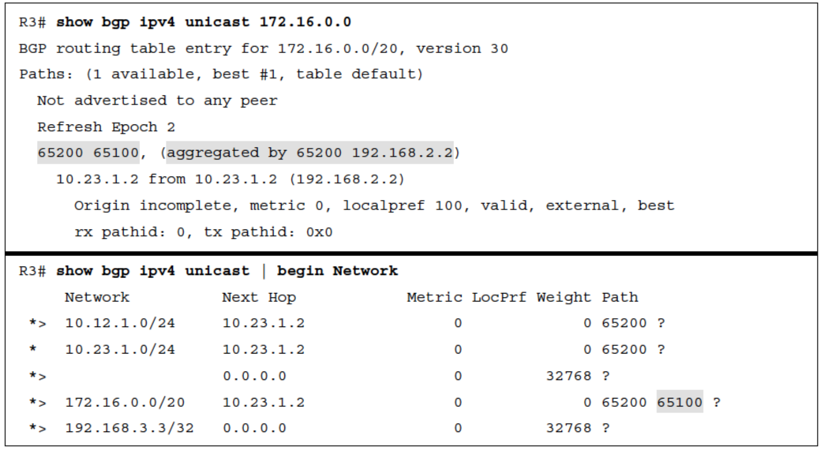
Route Summarization

Verifying Path Attributes and Aggregated Properties

Example 12-12 shows the 172.16.0.0/20 network prefix again, now that BGP attributes are to be propagated into the new prefix. Notice that the AS_Path information now contains AS 65100.

Example 12-13 displays R2's BGP table and the path attributes for the aggregated 192.168.0.0/16 network entry.

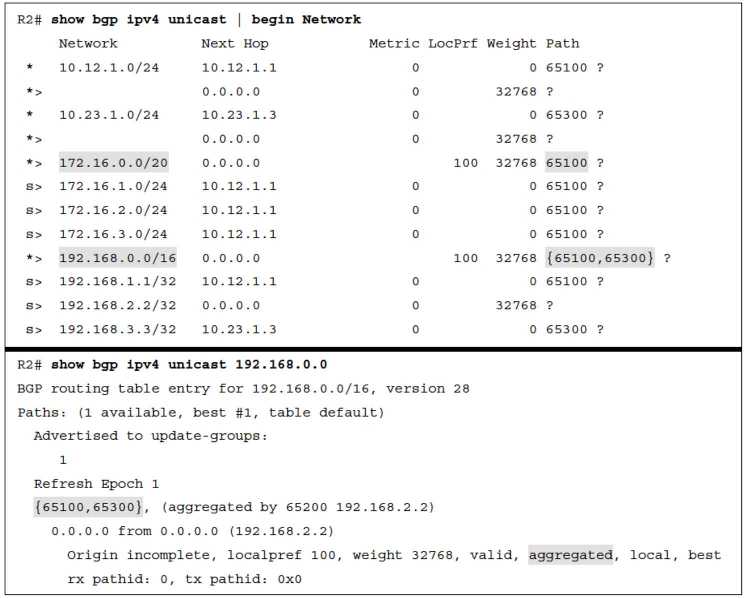
Example 12-12 *Verifying That Path Attributes Are Injected into the BGP Aggregate*



```
R3# show bgp ipv4 unicast 172.16.0.0
BGP routing table entry for 172.16.0.0/20, version 30
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 2
  65200 65100, (aggregated by 65200 192.168.2.2)
    10.23.1.2 from 10.23.1.2 (192.168.2.2)
      Origin incomplete, metric 0, localpref 100, valid, external, best
      rx pathid: 0, tx pathid: 0x0
```

```
R3# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.12.1.0/24     10.23.1.2                0             0 65200 ?
 *   10.23.1.0/24     10.23.1.2                0             0 65200 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  172.16.0.0/20    10.23.1.2                0             0 65200 65100 ?
 *>  192.168.3.3/32   0.0.0.0                  0         32768 ?
```

Example 12-13 *Viewing the Aggregated Properties of 192.168.0.0/16*



```
R2# show bgp ipv4 unicast | begin Network
     Network          Next Hop            Metric LocPrf Weight Path
 *   10.12.1.0/24     10.12.1.1                0             0 65100 ?
 *>                   0.0.0.0                  0         32768 ?
 *   10.23.1.0/24     10.23.1.3                0             0 65300 ?
 *>                   0.0.0.0                  0         32768 ?
 *>  172.16.0.0/20    0.0.0.0                       100  32768 65100 ?
 s>  172.16.1.0/24    10.12.1.1                0             0 65100 ?
 s>  172.16.2.0/24    10.12.1.1                0             0 65100 ?
 s>  172.16.3.0/24    10.12.1.1                0             0 65100 ?
 *>  192.168.0.0/16   0.0.0.0                       100  32768 {65100,65300} ?
 s>  192.168.1.1/32   10.12.1.1                0             0 65100 ?
 s>  192.168.2.2/32   0.0.0.0                  0         32768 ?
 s>  192.168.3.3/32   10.23.1.3                0             0 65300 ?
```

```
R2# show bgp ipv4 unicast 192.168.0.0
BGP routing table entry for 192.168.0.0/16, version 28
Paths: (1 available, best #1, table default)
  Advertised to update-groups:
     1
  Refresh Epoch 1
  {65100,65300}, (aggregated by 65200 192.168.2.2)
    0.0.0.0 from 0.0.0.0 (192.168.2.2)
      Origin incomplete, localpref 100, weight 32768, valid, aggregated, local, best
      rx pathid: 0, tx pathid: 0x0
```

Route Aggregation with AS_SET

R1 does not install the 192.168.0.0/16 network prefix for the same reason that R3 does not install the 192.168.0.0/16 network prefix. R1 thinks that the advertisement is a loop because it detects AS 65100 in AS_Path. You can confirm this by examining R1's BGP table, shown in Example 12-14.

Example 12-14 *R1's BGP Table, with 192.168.0.0/16 Discarded*

```
R1# show bgp ipv4 unicast | begin Network
    Network          Next Hop            Metric LocPrf Weight Path
 *  10.12.1.0/24     10.12.1.2                0             0 65200 ?
 *>                  0.0.0.0                  0         32768 ?
 *> 10.23.1.0/24     10.12.1.2                0             0 65200 ?
 *> 172.16.1.0/24    0.0.0.0                  0         32768 ?
 *> 172.16.2.0/24    0.0.0.0                  0         32768 ?
 *> 172.16.3.0/24    0.0.0.0                  0         32768 ?
 *> 192.168.1.1/32   0.0.0.0                  0         32768 ?
```

BGP Route Filtering and Manipulation

- Conditional route selection is a method for selectively identifying prefixes that are advertised or received from peers.
- Selected routes can be modified or removed to manipulate traffic flows, reduce memory utilization, or improve security.

BGP Route Filtering and Manipulation

Figure 12-2 shows the complete BGP route processing logic. Notice that the route policies occur on inbound route receipt and outbound route advertisement.

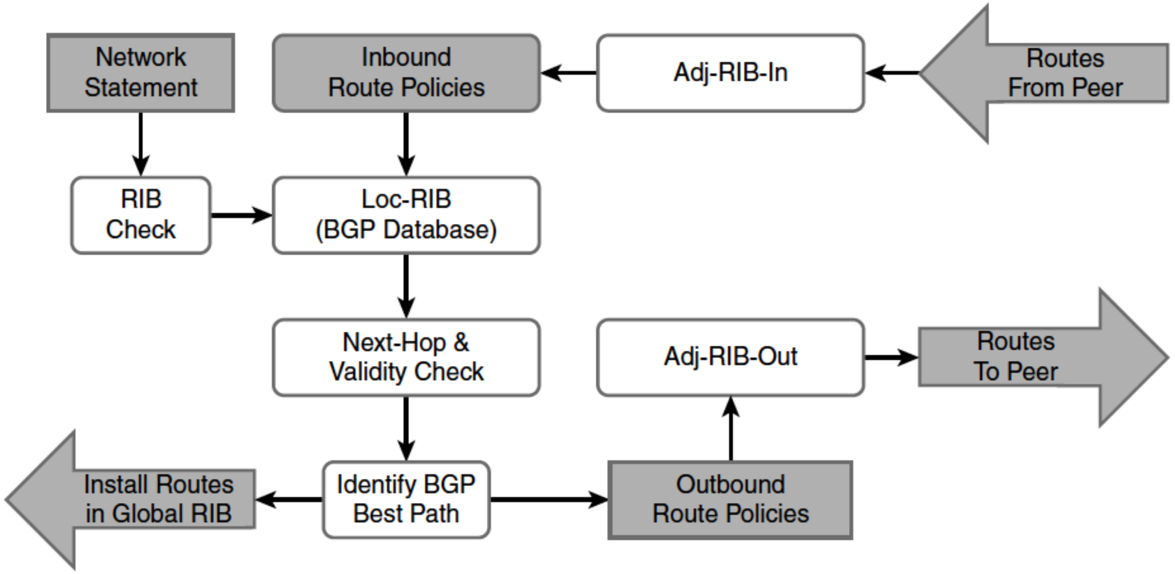


Figure 12-2 *BGP Route Policy Processing*

BGP Route Filtering and Manipulation (Cont.)

IOS XE provides four methods of filtering routes inbound or outbound for a specific BGP peer. Each of these methods can be used individually or simultaneously with other methods:

- **Distribution list -** A distribution list filters network prefixes based on a standard or extended access control list (ACL). An implicit deny is associated with any prefix that is not permitted.
- **Prefix list -** A list of prefix-matching specifications that permit or deny network prefixes in a top-down fashion, much like an ACL. An implicit deny is associated with any prefix that is not permitted.
- **AS_Path ACL/filtering -** A list of regex commands that allows for the permit or deny of a network prefix, based on the current AS_Path values. An implicit deny is associated with any prefix that is not permitted.
- **Route maps -** Route maps provide a method of conditional matching on a variety of prefix attributes and allow you to take a variety of actions. An action could be a simple permit or deny or could include the modification of BGP path attributes. An implicit deny is associated with any prefix that is not permitted.

BGP Route Filtering and Manipulation

Distribution List Filtering

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

Example 12-16 *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
```

Example 12-17 *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 ?
```

Example 12-16 shows R1’s BGP configuration, which demonstrates filtering with distribution lists.

Example 12-17 shows the routing table of R1.

Prefix List Filtering

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 12-18 *Prefix List Filtering Configuration*

```
R1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# ip prefix-list RFC1918 seq 10 permit 10.0.0.0/8 le 32
R1(config)# ip prefix-list RFC1918 seq 20 permit 172.16.0.0/12 le 32
R1(config)# ip prefix-list RFC1918 seq 30 permit 192.168.0.0/16 le 32
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 12-19 *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 ?
```

Example 12-18 shows the configuration of the prefix list and application to R2.

Example 12-19 shows the prefix list has been applied on R1.

BGP Route Filtering and Manipulation

AS_Path Filtering

There may be times when conditionally matching network prefixes may be too complicated, and identifying all routes from a specific organization is preferred. In such a case, path selection can be used with the BGP AS_Path. AS_Path filtering is accomplished using an AS_Path ACL, which uses regular expressions for matching.

To parse through the large number of available ASNs (4,294,967,295), you can use regular expressions (regex).

Table 12-2 provides a brief list and description of the common regex query modifiers.

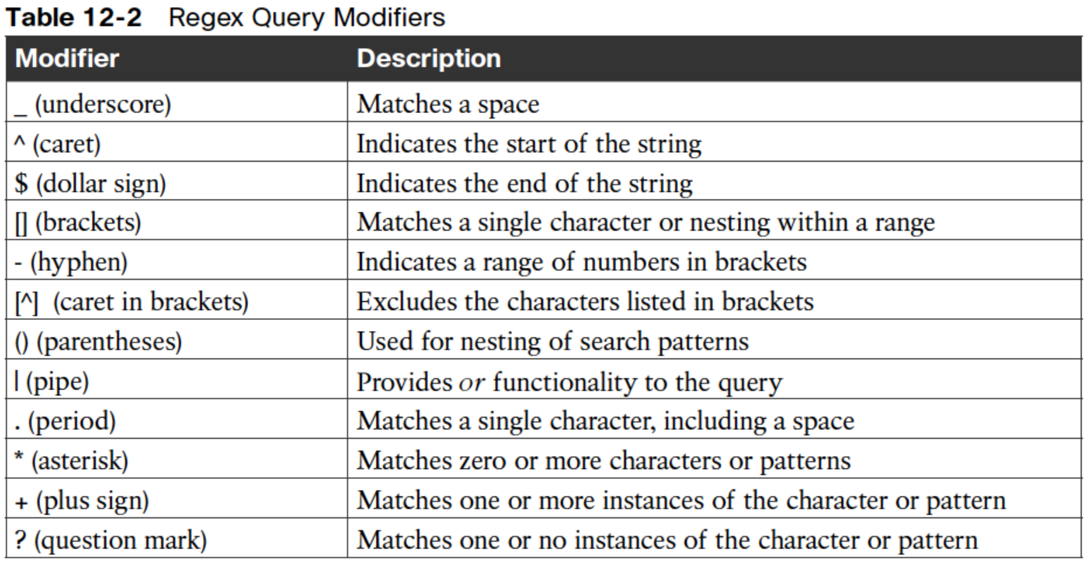


Table 12-2 Regex Query Modifiers

Modifier	Description
_ (underscore)	Matches a space
^ (caret)	Indicates the start of the string
$ (dollar sign)	Indicates the end of the string
[] (brackets)	Matches a single character or nesting within a range
- (hyphen)	Indicates a range of numbers in brackets
[^] (caret in brackets)	Excludes the characters listed in brackets
() (parentheses)	Used for nesting of search patterns
| (pipe)	Provides *or* functionality to the query
. (period)	Matches a single character, including a space
* (asterisk)	Matches zero or more characters or patterns
+ (plus sign)	Matches one or more instances of the character or pattern
? (question mark)	Matches one or no instances of the character or pattern

BGP Table for Regex Queries

The BGP table can be parsed with regex using the command **show bgp** *afi safi* **regexp** *regex-pattern*.

Figure 12-3 provides a reference topology, and Example 12-20 shows a reference BGP table to demonstrate the various regex query modifiers that can be used for a variety of common tasks.

AS_Path for the prefix 172.16.129.0/24 includes AS 300 twice non-consecutively for a specific purpose. This would not be seen in real life because it indicates a routing loop.

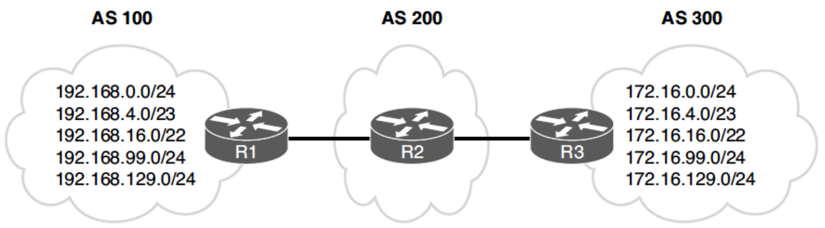
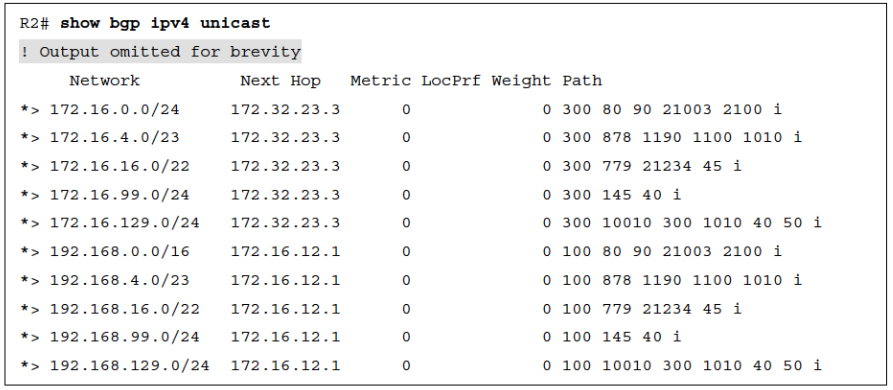


Figure 12-3 *BGP Regex Reference Topology*

Example 12-20 *BGP Table for Regex Queries*



```
R2# show bgp ipv4 unicast
! Output omitted for brevity
     Network          Next Hop     Metric LocPrf Weight Path
*> 172.16.0.0/24    172.32.23.3       0             0 300 80 90 21003 2100 i
*> 172.16.4.0/23    172.32.23.3       0             0 300 878 1190 1100 1010 i
*> 172.16.16.0/22   172.32.23.3       0             0 300 779 21234 45 i
*> 172.16.99.0/24   172.32.23.3       0             0 300 145 40 i
*> 172.16.129.0/24  172.32.23.3       0             0 300 10010 300 1010 40 50 i
*> 192.168.0.0/16   172.16.12.1       0             0 100 80 90 21003 2100 i
*> 192.168.4.0/23   172.16.12.1       0             0 100 878 1190 1100 1010 i
*> 192.168.16.0/22  172.16.12.1       0             0 100 779 21234 45 i
*> 192.168.99.0/24  172.16.12.1       0             0 100 145 40 i
*> 192.168.129.0/24 172.16.12.1       0             0 100 10010 300 1010 40 50 i
```

BGP Route Filtering and Manipulation

BGP Regex Queries

Example 12-22 shows how to use the underscore (_) to imply a space left of the ASN (100) remove the unwanted ASNs. The regex query includes the following unwanted ASN: 10010.

Example 12-25 shows the caret (^) in the regex pattern.

Example 12-27 provides a solution using the dollar sign ($) for the regex the pattern _40$.

Example 12-22 *BGP Regex Query for AS _100*

```
R2# show bgp ipv4 unicast regexp _100
! Output omitted for brevity
     Network          Next Hop        Metric LocPrf Weight Path
*> 172.16.129.0/24   172.32.23.3          0             0 300 10010 300 1010 40 50 i
*> 192.168.0.0/16    172.16.12.1          0             0 100 80 90 21003 2100 i
*> 192.168.4.0/23    172.16.12.1          0             0 100 878 1190 1100 1010 i
*> 192.168.16.0/22   172.16.12.1          0             0 100 779 21234 45 i
*> 192.168.99.0/24   172.16.12.1          0             0 100 145 40 i
*> 192.168.129.0/24  172.16.12.1          0             0 100 10010 300 1010 40 i
```

Example 12-25 *BGP Regex Query with Caret*

```
R2# show bgp ipv4 unicast regexp ^300_
! Output omitted for brevity
     Network          Next Hop          Metric LocPrf Weight Path
*> 172.16.0.0/24     172.32.23.3            0             0 300 80 90 21003 2100 i
*> 172.16.4.0/23     172.32.23.3            0             0 300 878 1190 1100 1010 i
*> 172.16.16.0/22    172.32.23.3            0             0 300 779 21234 45 i
*> 172.16.99.0/24    172.32.23.3            0             0 300 145 40 i
*> 172.16.129.0/24   172.32.23.3            0             0 300 10010 300 1010 40 50 i
```

Example 12-27 *BGP Regex Query with Dollar Sign*

```
R2# show bgp ipv4 unicast regexp _40$
! Output omitted for brevity
     Network          Next Hop        Metric  LocPrf Weight Path
*> 172.16.99.0/24    172.32.23.3         0             0 300 145 40 i
*> 192.168.99.0/24   172.16.12.1         0      100    0 100 145 40 i
```

BGP Route Filtering and Manipulation

AS_Path ACLs

Selecting routes from a BGP neighbor by using AS_Path requires the definition of an AS_Path access control list (AS_Path ACL). The AS_Path ACL processing is performed in a sequential top-down order, and the first qualifying match processes against the appropriate permit or deny action. An implicit deny exists at the end of the AS path ACL.

The command **ip as-path access-list** *acl-number* {**deny** | **permit**} *regex-query* creates the AS_Path access-list. The ACL is then applied with the command **neighbor** *ip-address* **filter-list** *acl-number* {**in** | **out**}.

Example 12-37 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 12-37 *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
```

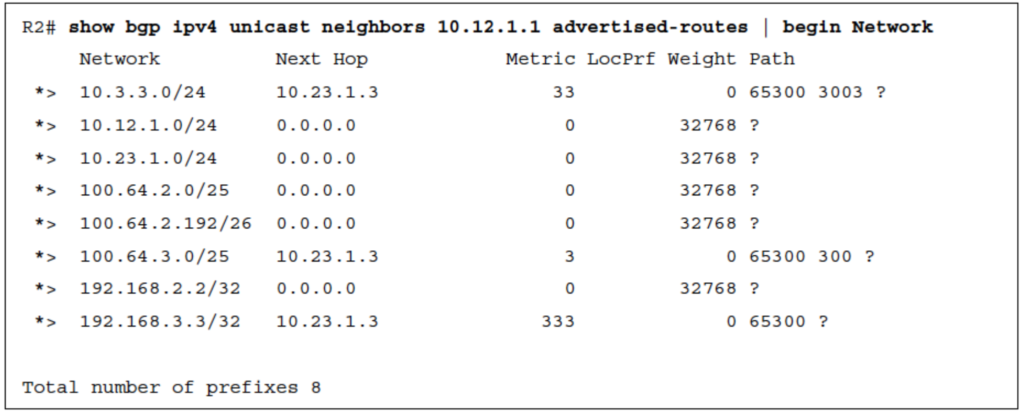
BGP Route Filtering and Manipulation

AS_Path ACLs (Cont.)

Example 12-36 shows the routes before the BGP AS_Path ACL was applied.

Example 12-38 shows the routes being advertised to R1. Notice that the routes do not all have AS_Paths confirming that only locally originating routes are being advertised externally.

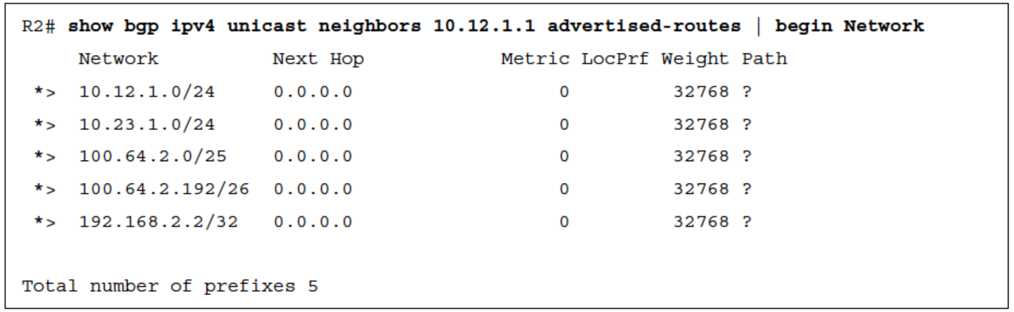
Example 12-36 *Reference BGP Table before Applying AS_Path Access List*



```
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
```

Example 12-38 *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
```

Route Maps

Route maps provide more functionality than pure filtering; they provide a method to manipulate BGP path attributes as well.

Route maps are applied on a BGP neighbor basis for routes that are advertised or received. A different route map can be used for each direction.

The route map is associated to the BGP neighbor with the command **neighbor** *ip-address* **route-map** *route-map-name* {**in** | **out**} under the specific address family.

Route maps allow for multiple steps in processing as well.

BGP Route Filtering and Manipulation

Route Map Configuration

Example 12-40 shows R1’s configuration, where multiple prefix lists are referenced along with an AS_Path ACL.

This route map includes four steps:

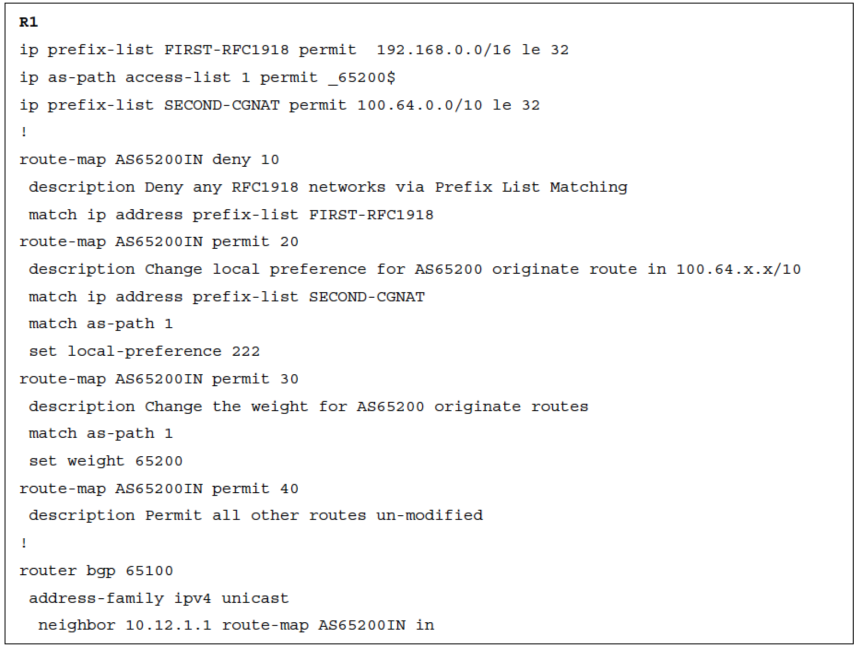
Step 1. Deny any routes that are in the 192.168.0.0/16 network by using a prefix list.

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

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

Step 4. Permit all other routes to process.

Example 12-40 *R1’s Route Map Configuration for Inbound AS 65200 Routes*



```
R1
ip prefix-list FIRST-RFC1918 permit  192.168.0.0/16 le 32
ip as-path access-list 1 permit _65200$
ip prefix-list SECOND-CGNAT permit 100.64.0.0/10 le 32
!
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 Filtering and Manipulation

Verifying Route Map

Example 12-39 shows R1’s BGP routing table before a route map was applied and Example 12-41 shows after the route map was applied. The following actions 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 originate 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 have been assigned the locally significant BGP attribute weight 65200.
- All other routes were received and not modified.

Example 12-39 *BGP Table Before Application of 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 ?
```

Example 12-41 *Verification of 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 ?
```

BGP Route Filtering and Manipulation

Clearing BGP Connections

Depending on the change to the BGP route manipulation technique, the BGP session may need to be refreshed to take effect. BGP supports two methods of clearing a BGP session.

The first method is a hard reset, which tears down the BGP session, removes BGP routes from the peer, and is the most disruptive.

- You can initiate a hard reset on a router with the command **clear ip bgp** *ip-address*. You can clear all the router's BGP sessions by using an asterisk (*) in lieu of the peer's IP address.

The second method is a soft reset, which invalidates the BGP cache and requests a full advertisement from its BGP peer. A soft reset reduces the number of routes that must be exchanged if multiple address families are configured with a single BGP peer.

- You can initiate a soft reset on a router with the command **clear ip bgp** *ip-address* **soft**
- You can perform a soft reset for a specific address family with the command **clear bgp** *afi safi* {*ip-address* | *} **soft** [**in** | **out**].

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 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 through 4,294,967,295) or as two 16-bit numbers (0 through 65535):(0 through 65535), commonly referred to as new format.

By convention, with private BGP communities, the first 16 bits represent the AS of the community origination, and the second 16 bits represent a pattern defined by the originating AS.

The private BGP community pattern can vary from organization to organization and does not need to be registered. The pattern could signify geographic locations for one AS while signifying a method of route advertisement in another AS.

In 2006, RFC 4360 expanded the capabilities of BGP communities by providing an extended format. Extended BGP communities provide structure for various classes of information and are commonly used for VPN services.

BGP Communities

Enabling BGP Community Support

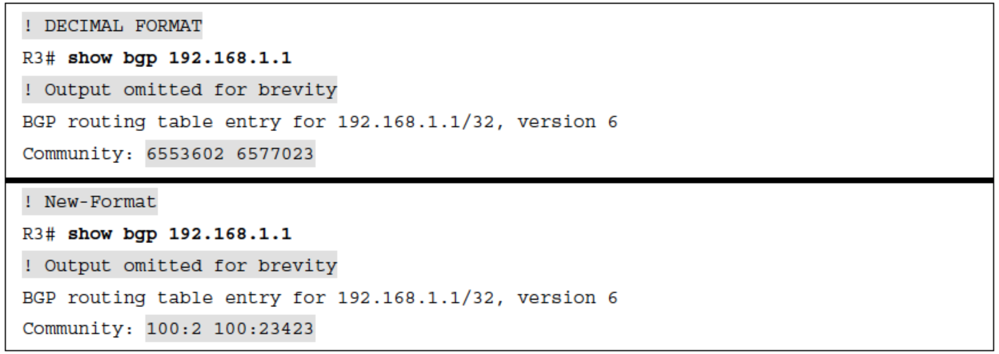
IOS and IOS XE routers do not advertise BGP communities to peers by default. Communities are enabled on a neighbor-by-neighbor basis with the BGP address family configuration command **neighbor** *ip-address* **send-community** [**standard** | **extended** | **both**] under the neighbor's address family configuration.

If a keyword is not specified, standard communities are sent by default.

IOS XE nodes can display communities in new format, and they are easier to read if you use the global configuration command **ip bgp-community new-format**.

Example 12-42 shows the BGP community in decimal format on top and then in new format.

Example 12-42 *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 Communities

Well-Known Communities

RFC 1997 defined a set of global communities (known as well-known communities) that use the community range 4,294,901,760 (0xFFFF0000) to 4,294,967,295 (0xFFFFFFFF). All routers that are capable of sending/receiving BGP communities must implement well-known communities.

The following are the common well-known communities:

- Internet
- No_Advertise
- No_Export
- Local-AS

BGP Communities

No-Advertise BGP Community

No_Advertise community routes should not be advertised to any BGP peer. The No_Advertise BGP community can be advertised from an upstream BGP peer or locally with an inbound BGP policy. The No_Advertise community is set with the command **set community no-advertise** within a route map.

- Figure 12-4 demonstrates that R1 is advertising the 10.1.1.0/24 network to R2.
- Example 12-43 shows R2's Network Layer Reachability Information (NLRI) for the 10.1.1.0/24 network prefix.
- BGP routes that are set with the No_Advertise community are quickly seen with the command **show bgp** *afi safi* **community no-advertise**, as shown in Example 12-44.

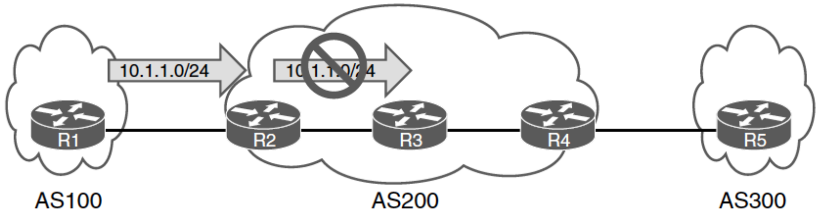


Figure 12-4 *BGP No_Advertise Community Topology*

Example 12-43 *BGP Attributes for No_Advertise Routes*

```
R2# show bgp 10.1.1.0/24
! Output omitted for brevity
BGP routing table entry for 10.1.1.0/24, version 18
Paths: (1 available, best #1, table default, not advertised to any peer)
  Not advertised to any peer
  Refresh Epoch 1
  100, (received & used)
    10.1.12.1 from 10.1.12.1 (192.168.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Community: no-advertise
```

Example 12-44 *Display of Prefixes with No_Advertise Community*

```
R2# show bgp ipv4 unicast community no-advertise
! Output omitted for brevity
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      10.1.12.1                0             0 100 i
```

BGP Communities

No-Export BGP Community

When a route is received with the No_Export community, the route is not advertised to any eBGP peer. The No_Export community is set with the command **set community no-export** within a route map.

- Figure 12-5 shows a topology with three ASs.
- Example 12-45 confirms that the prefix is not advertised to any eBGP peer.
- The command **show bgp** *afi safi* **community no-export** shows all the BGP prefixes that contain the Local-AS community, as demonstrated in Example 12-46.

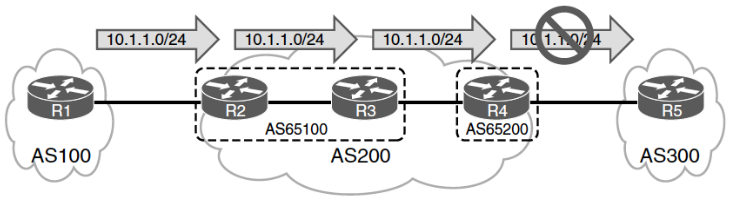


Figure 12-5 *BGP No-Export Community Topology*

Example 12-45 *BGP Attributes for No_Export Routes*

```
R3# show bgp ipv4 unicast 10.1.1.0/24
BGP routing table entry for 10.1.1.0/24, version 6
Paths: (1 available, best #1, table default, not advertised to EBGP peer)
  Advertised to update-groups:
     3
  Refresh Epoch 1
  100, (Received from a RR-client), (received & used)
    10.1.23.2 from 10.1.23.2 (192.168.2.2)
      Origin IGP, metric 0, localpref 100, valid, confed-internal, best
      Community: no-export
```

```
R4# show bgp ipv4 unicast 10.1.1.0/24
! Output omitted for brevity
BGP routing table entry for 10.1.1.0/24, version 4
Paths: (1 available, best #1, table default, not advertised to EBGP peer)
  Not advertised to any peer
  Refresh Epoch 1
  (65100) 100, (received & used)
    10.1.23.2 (metric 20) from 10.1.34.3 (192.168.3.3)
      Origin IGP, metric 0, localpref 100, valid, confed-external, best
      Community: no-export
```

Example 12-46 *Viewing BGP Routes with the No_Export Community*

```
R4# show bgp ipv4 unicast community no-export | b Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      10.1.23.2                0    100      0 (65100) 100 i
```

```
R2# show bgp ipv4 unicast community no-export | b Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      10.1.12.1                0             0 100 i
```

BGP Communities

No-Export SubConfed Community

With the No_Export_SubConfed community known as the Local-AS community, a route is not advertised outside the local AS. The Local- AS community is set with the command **set community local-as** within a route map.

- Figure 12-6 shows a topology with three ASs.
- Example 12-47 confirms that the prefix is not advertised outside local AS and that the prefix is not advertised to any peer.
- The command **show bgp** *afi safi* **community local-as** shows all the BGP prefixes that contain the Local-AS community, as demonstrated in Example 12-48.

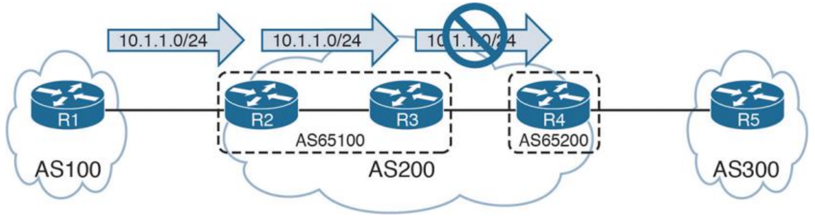


Figure 12-6 *BGP Local-AS Community Topology*

Example 12-47 *BGP Attributes for Local-AS Routes*

```
R3# show bgp ipv4 unicast 10.1.1.0/24
BGP routing table entry for 10.1.1.0/24, version 8
Paths: (1 available, best #1, table default, not advertised outside local AS)
  Not advertised to any peer
  Refresh Epoch 1
  100, (Received from a RR-client), (received & used)
    10.1.23.2 from 10.1.23.2 (192.168.2.2)
      Origin IGP, metric 0, localpref 100, valid, confed-internal, best
      Community: local-AS
```

Example 12-48 *Viewing BGP Routes with the Local-AS Community*

```
R3# show bgp ipv4 unicast community local-AS  | b Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>i 10.1.1.0/24      10.1.23.2                0    100      0 100 i
```

```
R2# show bgp ipv4 unicast community local-AS  | b Network
     Network          Next Hop            Metric LocPrf Weight Path
 *>  10.1.1.0/24      10.1.12.1                0             0 100 i
```

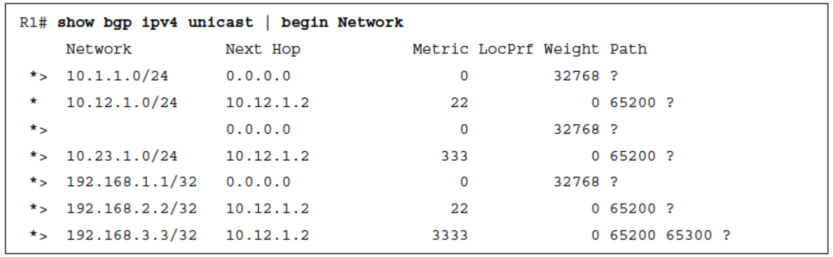
BGP Communities

Conditionally Matching BGP Communities

Conditionally matching BGP communities allows for selection of routes based on the BGP communities within the route’s path attributes so that selective processing can occur in route maps.

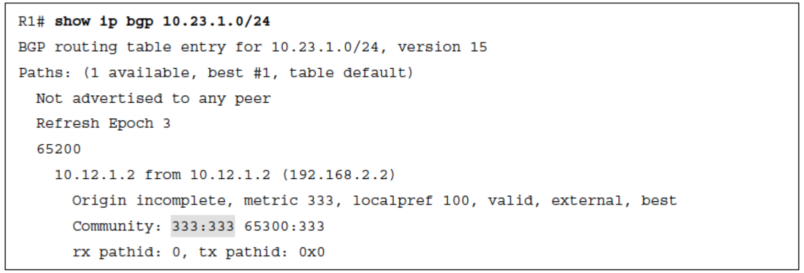
You display the entire BGP table by using the command **show bgp** *afi safi* **detail** and then manually select a route with a specific community. However, if the BGP community is known, you can display all the routes by using the command **show bgp** *afi safi* **community** *community*.

Example 12-49 *BGP Routes from R2 (AS 65200)*



```
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             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 ?
```

Example 12-50 *Viewing BGP Path Attributes for the 10.23.1.0/24 Network*



```
R1# show ip bgp 10.23.1.0/24
BGP routing table entry for 10.23.1.0/24, version 15
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Refresh Epoch 3
  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
      rx pathid: 0, tx pathid: 0x0
```

Example 12-49 shows the BGP table for R1, which has received multiple routes from R2 (AS 65200).

Example 12-50 shows the explicit path entry for the 10.23.1.0/24 network and all the BGP path attributes.

BGP Communities

Conditionally Matching BGP Communities (Cont.)

Conditionally matching requires the creation of a community list that shares a similar structure to an ACL, can be standard or expanded, and can be referenced by number or name.

- The configuration syntax for a community list is **ip community-list** {*1-500* | **standard** *list-name* | **expanded** *list-name*} {**permit** | **deny**} *community-pattern*. The community list is referenced in a route map with the command **match community** *1-500*.
- Example 12-51 demonstrates the creation of a BGP community list that matches on the community 333:333.
- Example 12-52 shows the BGP table after the route map has been applied to the neighbor.

Example 12-51 *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
```

Example 12-52 *R1's BGP Table After Application of 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 ?
```

BGP Communities

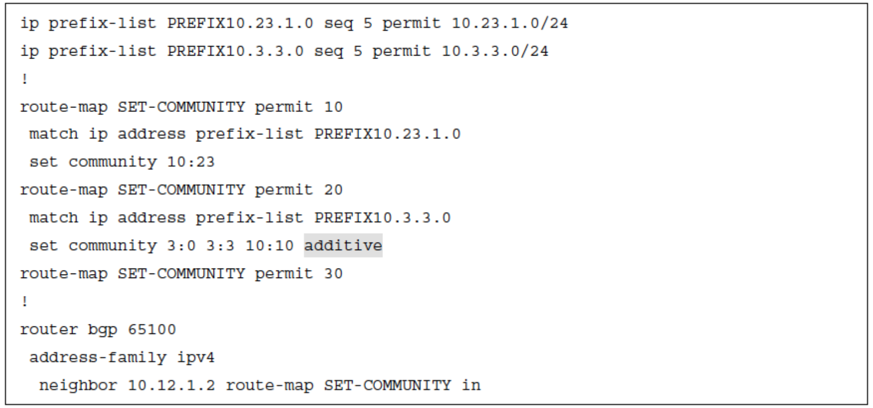
Private BGP Communities

You set a private BGP community in a route map by using the command set community bgp-community [additive]. By default, when you set a community, any existing communities are overwritten, but you can preserve them by using the optional additive keyword.

Example 12-54 shows the configuration where the BGP community is set on the 10.23.1.0/24 network.

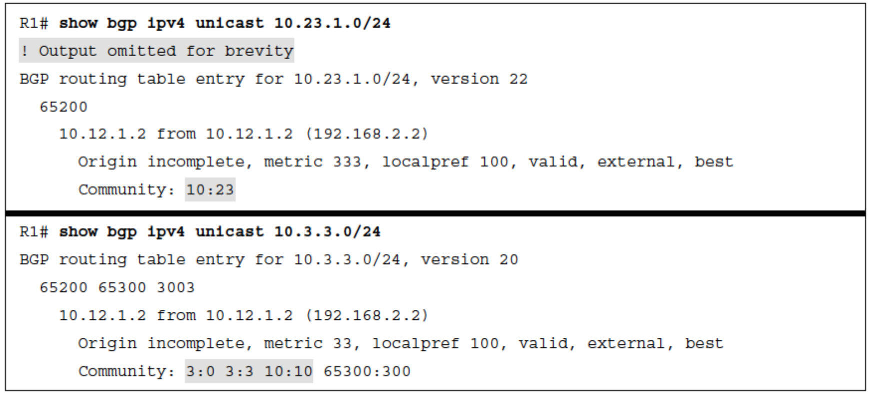
After the route map has been applied and the routes have been refreshed, the path attributes can be examined, as demonstrated in Example 12-55.

Example 12-54 *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 12-55 *Verification of 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
```

Maximum Prefix

- The BGP maximum prefix feature restricts the number of routes that are received from a BGP peer.
- This feature ensures that the BGP table does not overwhelm the router by exceeding its memory or processing capability.
- Prefix limits are typically set for BGP peers on low-end routers as a safety mechanism to ensure that they do not become overloaded.

Maximum Prefix

The BGP maximum prefix feature restricts the number of routes that are received from a BGP peer. This feature ensures that the BGP table does not overwhelm the router by exceeding its memory or processing capability.

Routers can place prefix restrictions on a BGP neighbor by using the BGP address family configuration command **neighbor** *ip-address* **maximum-prefix** *prefix-count* [*warning-percentage*] [**restart** *time*] [**warning-only**].

When a peer advertises more routes than the maximum prefix count, the peer moves the neighbor to the idle (PfxCt) state in the finite-state machine (FSM), closes the BGP session, and sends out the appropriate syslog message. The BGP session is not automatically reestablished by default. If you want to restart the BGP session after a certain amount of time, you can use the optional keyword restart time.

Maximum Prefix (Cont.)

A warning is not generated before the prefix limit is reached. By adding a warning percentage (set to 1 to 100) after the maximum prefix count, you can have a warning message sent when the percentage is exceeded. The maximum prefix behavior of closing the BGP session can be changed by using the optional keyword warning-only so that a warning message is generated instead. When the threshold has been reached, additional prefixes are discarded.

Example 12-56 shows the maximum prefix configuration.

Example 12-57 shows the 10.12.1.2 neighbor has exceeded the maximum prefix threshold and shut down the BGP session.

Example 12-56 *Maximum Prefix Configuration*

```
router bgp 100
 neighbor 10.12.1.2 remote-as 200
!
 address-family ipv4
  neighbor 10.12.1.2 activate
  neighbor 10.12.1.2 maximum-prefix 7
```

Example 12-57 *Maximum Prefix Violation*

```
R1# show bgp ipv4 unicast summary | begin Neighbor
Neighbor        V       AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.12.1.2       4      200       0       0        1    0    0 00:01:14 Idle (PfxCt)
```

```
R1# show log | include BGP
05:10:04.989: %BGP-5-ADJCHANGE: neighbor 10.12.1.2 Up
05:10:04.990: %BGP-4-MAXPFX: Number of prefixes received from 10.12.1.2 (afi 0)
 reaches 6, max 7
05:10:04.990: %BGP-3-MAXPFXEXCEED: Number of prefixes received from 10.12.1.2
 (afi 0): 8 exceeds limit 7
05:10:04.990: %BGP-3-NOTIFICATION: sent to neighbor 10.12.1.2 6/1
 (Maximum Number of Prefixes Reached) 7 bytes 00010100 000007
05:10:04.990: %BGP-5-NBR_RESET: Neighbor 10.12.1.2 reset
 (Peer over prefix limit)
05:10:04.990: %BGP-5-ADJCHANGE: neighbor 10.12.1.2 Down Peer over prefix limit
```

Configuration Scalability

- BGP configurations can become fairly large as features are configured or BGP sessions increase.
- IOS-based operating systems provide methods to apply a similar configuration to multiple neighbors.
- This simplifies the configuration from a deployment standpoint and makes the configuration easier to read.

Configuration Scalability
IOS Peer Groups

IOS peer groups simplify BGP configuration and reduce system resource use (CPU and memory) by grouping BGP peers together into BGP update groups.

BGP update groups enable a router to perform the outbound routing policy processing one time and then replicate the update to all the members (as opposed to performing the outbound routing policy processing for every router).

All peer group settings use the peer-groupname field in lieu of the ip-address field in the neighbor ip-address commands. All routers in the peer group are in the same update group and therefore must be of the same session type: internal (iBGP) or external (eBGP).

Configuration Scalability

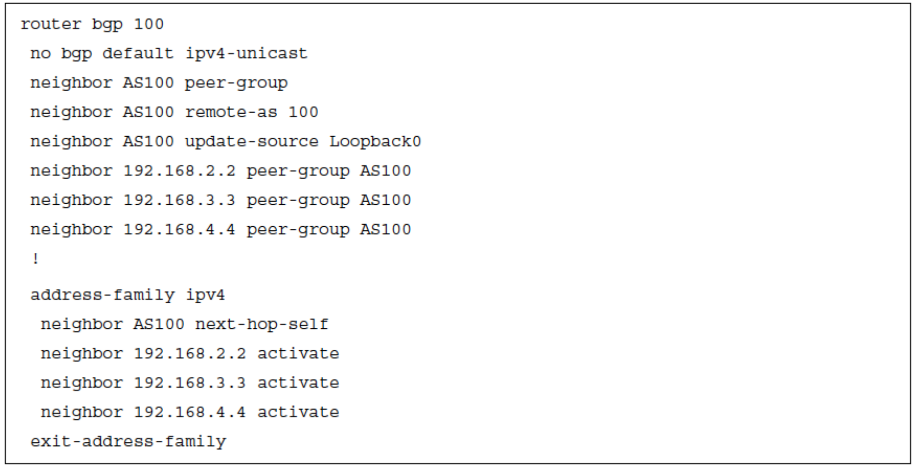
Peer Group Configuration

A peer group is defined by using the command **neighbor group-name** *peer-group* in the global BGP configuration. All BGP parameters are configured using peer-group groupname in lieu of neighbor ip-address.

To link BGP peer IP addresses to the peer group use the command **neighbor** *ip-address* **peer-group** *group-name*. BGP neighbors cannot be activated by peer group name and must be activated for each address family by IP address.

Example 12-58 shows R1's BGP configuration for peering with R2, R3, and R4.

Example 12-58 *Example Peer Group Configuration*



```
router bgp 100
 no bgp default ipv4-unicast
 neighbor AS100 peer-group
 neighbor AS100 remote-as 100
 neighbor AS100 update-source Loopback0
 neighbor 192.168.2.2 peer-group AS100
 neighbor 192.168.3.3 peer-group AS100
 neighbor 192.168.4.4 peer-group AS100
 !
 address-family ipv4
  neighbor AS100 next-hop-self
  neighbor 192.168.2.2 activate
  neighbor 192.168.3.3 activate
  neighbor 192.168.4.4 activate
 exit-address-family
```

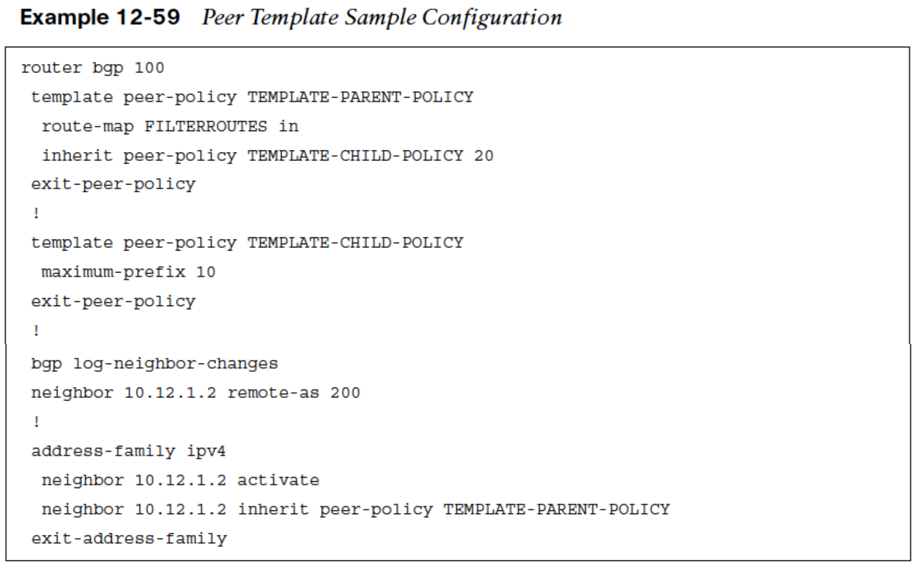
Configuration Scalability

IOS Peer Templates

IOS BGP peer templates allow for a reusable pattern of settings that can be applied as needed in a hierarchical format through inheritance and nesting of templates. If a conflict exists between an inherited configuration and the invoking peer template, the invoking template preempts the inherited value.

There are two types of BGP peer templates:

- **Peer session -** This type of template involves configuration settings specifically for the BGP session. You define peer session template settings with the BGP configuration command **template peer-session** *template-name* and then enter any BGP session related configuration commands.
- **Peer policy -** This type of template involves configuration settings specifically for the address family policy. You define peer policy template settings with the BGP configuration command **template peer-policy** *template-name* and then enter any BGP address family–related configuration commands.



Example 12-59 *Peer Template Sample Configuration*

```
router bgp 100
 template peer-policy TEMPLATE-PARENT-POLICY
  route-map FILTERROUTES in
  inherit peer-policy TEMPLATE-CHILD-POLICY 20
 exit-peer-policy
 !
 template peer-policy TEMPLATE-CHILD-POLICY
  maximum-prefix 10
 exit-peer-policy
 !
 bgp log-neighbor-changes
 neighbor 10.12.1.2 remote-as 200
 !
 address-family ipv4
  neighbor 10.12.1.2 activate
  neighbor 10.12.1.2 inherit peer-policy TEMPLATE-PARENT-POLICY
 exit-address-family
```

Prepare for the Exam

Prepare for the Exam

Key Topics for Chapter 12

Description		
Aggregate addresses	Prefix list filtering	The No_Export BGP community
Route aggregation with suppression	Regular expressions (regex)	The Local-AS No_Export_SubConfed_ BGP Community
The Atomic aggregate attribute	AS_Path ACLs	Community list configuration
Route aggregation with AS_SET	Route maps	Setting private BGP communities
BGP route policy processing	Clearing BGP connections	Maximum prefix
BGP filtering methods	Enabling BGP community support	IOS peer groups
Distribution list filtering	The No_Advertise BGP community	IOS peer templates

Prepare for the Exam

Key Terms for Chapter 12

Key Terms	
access control list (ACL)	No_Advertise community
AS path	peer group
atomic aggregate	peer template
BGP community	prefix list
BGP multihoming	regular expressions
distribute list	route map
Local-AS community	transit routing
No_Export community	

Prepare for the Exam

Command Reference for Chapter 12

Task	Command Syntax
Configure a BGP aggregate IPv4 prefix	**aggregate-address network** *subnet-mask* [**summary-only**] [**as-set**]
Configure a BGP aggregate IPv6 prefix	**aggregate-address** *prefix/prefixlength* [**summary-only**] [**as-set**]
Configure a prefix list	{**ip** | **ipv6**} **prefix-list** *prefix-list-name* [**seq** *sequence-number*] {**permit** | **deny**} *high-order-bit-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 using AS_Path	**match as-path** *acl-number*
Conditionally match in a route map using an ACL	**match ip address** {*acl-number* | *acl-name*}
Conditionally match in a route map using a prefix list	**match ip address prefix-list** *prefix-list-name*

Prepare for the Exam

Command Reference for Chapter 12

Task	Command Syntax
Conditionally match in a route map using local preference	**match local-preference** *local-preference*
Filter routes to a BGP neighbor using an ACL	**neighbor** *ip-address* **distribute-list** {*acl-number* | *acl-name*} {**in** | **out**}
Filter routes to a BGP neighbor 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 using an AS_Path ACL	**neighbor** *ip-address* **filter-list** *acl-number* {**in** | **out**}
Associate an inbound or outbound route map to 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**

Prepare for the Exam

Command Reference for Chapter 12

Task	Command Syntax
Create a BGP community list for conditional route matching	**ip community-list** {*1-500* | **standard** *list-name* | **expanded** *list-name*} {**permit** | **deny**} *community-pattern*
Set BGP communities in a route map	**set community** *bgp-community* [**additive**]
Configure the maximum number of BGP prefixes that a neighbor can receive	**neighbor** *ip-address* **maximum-prefix** *prefix-count* [*warning-percentage*] [**restart** *time*] [**warning-only**]
Define a BGP peer group	**neighbor** *group-name* **peer-group**
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*