ERICSSON

Configuring Routing Policies

SYSTEM ADMINISTRATOR GUIDE

25/1543-CRA 119 1170/1-V1 Uen F

Contents

1 Overview

This document provides an overview of routing policies and describes the tasks and commands used to configure, monitor, troubleshoot, and administer routing policy features through the SmartEdge router.

This document applies to both the Ericsson SmartEdge® and SM family routers. However, the software that applies to the SM family of systems is a subset of the SmartEdge OS; some of the functionality described in this document may not apply to SM family routers.

For information specific to the SM family chassis, including line cards, refer to the SM family chassis documentation.

For specific information about the differences between the SmartEdge and SM family routers, refer to the Technical Product Description *SM Family of Systems* (part number 5/221 02-CRA 119 1170/1) in the **`Product Overview`** folder of this Customer Product Information library.

Routing policies allow you to enforce routing policy decisions onto incoming, outgoing, and redistributed routes. The tools to configure routing policies include Border Gateway Protocol (BGP) autonomous system (AS) path lists, BGP community lists, BGP extended community lists, IP prefix lists, IP Version 6 (IPv6) prefix lists, and route maps with match and set conditions.

Note: When IPv6 addresses are not referenced or explicitly specified, the term, IP address, can refer generally to IP Version 4 (IPv4) addresses, IPv6 addresses, or IP addressing. In instances where IPv6 addresses are referenced or explicitly specified, the term, IP address, refers only to IPv4 addresses.

2 Configuration and Operations Tasks

Note: In this section, the command syntax in the task tables displays only the root command.

To configure routing policies, perform the tasks described in the following sections.

2.1 Configuring AS Path Lists

To configure BGP AS path lists, perform the tasks described in the sections that follow.

2.1.1 Create an AS Path List

To create an AS path list, perform the tasks described in Table 1.

Table 1 Create an AS Path List

Task	Root Command	Notes
Create an AS path list and enter AS path list configuration mode.	*as-path-list (context configuration)*	Enter this command in context configuration mode.
Associate a description with the BGP AS path list.	*description (lists)*	Enter this command in AS path list configuration mode.
Configure the AS path list permit or deny condition.	For the complete list of tasks used to configure the AS path list permit or deny condition, see Configure an AS Path List Permit or Deny Condition.	—

2.1.2 Configure an AS Path List Permit or Deny Condition

When you create several permit or deny conditions for a single list, the system can automatically sequence the entries for you, or you can manually assign a number for each entry. A BGP AS path attribute is compared with BGP AS path list entries in order of ascending sequence number to determine if routes associated with the AS path attribute are permitted or denied.

When you allow the system to automatically sequence entries for you, the system increments each statement by a count of 10. The first statement you

enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you might want to add later. You can also resequence numbers to existing entries in an AS path list.

To configure an AS path list permit or deny condition, perform the tasks described in Table 2. Enter all commands in AS path list configuration mode, unless otherwise noted.

Table 2 Configure an AS Path List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and allow the system to automatically assign sequence numbers for the AS path list statement.	*{permit | deny}*	Use the following command syntax: {*{permit | deny}*{***reg-exp*** | **any**}
Permit or deny routes matching the specified criteria, and manually assign a sequence number for the AS path list statement.	*{permit | deny}*	Use the following command syntax: **seq** ***seq-num*** *{permit | deny}* {***reg-exp*** | **any**}
Assign new sequence numbers to existing entries in a specified AS path list, so that entries are in increments of 10.	*resequence as-path-list*	Enter this command in context configuration mode. This command is useful when you have manually assigned sequence numbers and have no room to insert new entries in between existing entries.

2.2 Configuring BGP Community Lists

To configure BGP community lists, perform the tasks described in the following sections.

2.2.1 Create a BGP Community List

To create a BGP community list, perform the tasks described in Table 3.

Table 3 Create a BGP Community List

Task	Root Command	Notes
Create a BGP community list and enter community list configuration mode.	*community-list*	Enter this command in context configuration mode. A reference to a community list that does not exist, or does not contain any configured entries, implicitly matches and permits all community lists.
Associate a description with the BGP community list.	*description (lists)*	Enter this command in community list configuration mode.
Configure the BGP community list permit or deny condition.	For the complete list of tasks used to configure the BGP community list permit or deny condition, see Configure a BGP Community List Permit or Deny Condition.	—

2.2.2 Configure a BGP Community List Permit or Deny Condition

When you create several permit or deny conditions for a single BGP community list, the system can automatically sequence the entries for you, or you can manually assign a number for each entry. A BGP community attribute is compared with BGP community list entries in order of ascending sequence number to determine if they are permitted or denied.

When you allow the system to automatically sequence entries, the system increments each statement by a count of 10. The first statement you enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you might want to add later. You can also resequence existing entries in a BGP community list.

To configure a BGP community list permit or deny condition, perform the tasks described in Table 4. Enter all commands in community list configuration mode, unless otherwise noted.

Table 4 Configure a BGP Community List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and allow the system to automatically assign sequence numbers for the BGP community list statement.	*{permit | deny}*	Use the following command syntax: *{permit | deny}* {***community-num*** | **local-as** | **no-advertise** | **no-export** | **reg-exp** ***reg-exp*** | **any**}
Permit or deny routes matching the specified criteria, and manually assign a sequence number for the BGP community list statement.	*{permit | deny}*	Use the following command syntax: **seq** ***seq-num*** *{permit | deny}* {***community-num*** | **local-as** | **no-advertise** | **no-export** | **reg-exp** ***reg-exp*** | **any**}
Assign new sequence numbers to existing entries in a BGP community list, so that entries are in increments of 10.	*resequence community-list*	Enter this command in context configuration mode. This command is useful when you have manually assigned sequence numbers and have no room to insert new entries in between existing entries.

2.3 Configuring BGP Extended Community Lists

A BGP extended community is a group of destinations that share some common attributes. Extended community attributes are carried in BGP messages as attributes of the route. They identify the route as belonging to a specific collection of routes, all of which are treated the same with respect to routing policy. Each BGP extended community must be globally unique (contains either a public IP address or autonomous system number [ASN]).

BGP/Multiprotocol Label Switching Virtual Private Networks (BGP/MPLS VPNs) use BGP extended community attributes instead of conventional BGP community attributes.

To configure BGP extended community lists, perform the tasks described in the following sections.

2.3.1 Create a BGP Extended Community List

To create a BGP extended community list, perform the tasks described in Table 5.

Table 5 Create a BGP Extended Community List

Task	Root Command	Notes
Create a BGP extended community list and enter community list configuration mode.	*ext-community-list*	Enter this command in context configuration mode. A reference to an extended community list that does not exist, or does not contain any configured entries, implicitly matches and permits all extended community lists.
Associate a description with the BGP extended community list.	*description (lists)*	Enter this command in extended community list configuration mode.
Configure the BGP extended community list permit or deny condition.	For the complete list of tasks used to configure the BGP extended community list permit or deny condition, see Configure a BGP Extended Community List Permit or Deny Condition.	—

2.3.2 Configure a BGP Extended Community List Permit or Deny Condition

When you create several permit or deny conditions for a single BGP extended community list, the system can automatically sequence the entries for you, or you can manually assign a number for each entry. A BGP extended community attribute is compared with BGP extended community list entries in order of ascending sequence number to determine if they are permitted or denied.

When you allow the system to automatically sequence entries for you, the system increments each statement by a count of 10. The first statement you enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you might want to add later. You can also resequence existing entries in a BGP extended community list.

To configure a BGP extended community list permit or deny condition, perform the tasks described in Table 6. Enter all commands in extended community list configuration mode, unless otherwise noted.

Table 6 Configure a BGP Extended Community List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and allow the system to automatically assign sequence numbers for the BGP extended community list statement.	*{permit | deny}*	Use the following command syntax: *{permit | deny}* {***ext-community-num*** | **reg-exp** ***reg-exp*** | **any**}
Permit or deny routes matching the specified criteria, and manually assign a sequence number for the BGP extended community list statement.	*{permit | deny}*	Use the following command syntax: **seq** ***seq-num*** *{permit | deny}* {***ext-community-num*** | **reg-exp** ***reg-exp*** | **any**}
Assign new sequence numbers to existing entries in a BGP extended community list, so that entries are in increments of 10.	*resequence community-list*	Enter this command in context configuration mode. This command is useful when you have manually assigned sequence numbers and have no room to insert new entries in between existing entries.

2.4 Configuring IP Prefix Lists

To configure IP prefix lists, perform the tasks described in the following sections.

2.4.1 Create an IP Prefix List

To create an IP prefix list, perform the tasks described in Table 7.

Table 7 Create an IP Prefix List

Task	Root Command	Notes
Create an IP prefix list used to filter routes and enter IP prefix list configuration mode.	*ip prefix-list*	Enter this command in context configuration mode. A reference to an IP prefix list that does not exist, or does not contain any configured entries, implicitly matches and permits all IP prefixes.

Table 7 Create an IP Prefix List

Task	Root Command	Notes
Associate a description with the IP prefix list.	*description (lists)*	Enter this command in IP prefix list configuration mode.
Configure the IP prefix list permit or deny condition.	For the complete list of tasks used to configure the IP prefix list permit or deny condition, seeConfigure an IP Prefix List Permit or Deny Condition.	—

2.4.2 Configure an IP Prefix List Permit or Deny Condition

When you create several permit or deny conditions for a single IP prefix list, the system can automatically sequence the entries for you, or you can manually assign a number for each entry.

When you allow the system to automatically sequence the entries for you, the system increments each statement by a count of 10. The first statement you enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you might want to add later. You can also resequence existing entries in an IP prefix list.

To configure an IP prefix list permit or deny condition, perform the tasks described in Table 8. Enter all commands in IP prefix list configuration mode, unless otherwise noted.

Table 8 Configure an IP Prefix List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and allow the system to automatically assign sequence numbers for the IP prefix list statement.	*{permit | deny}*	Use the following command syntax: *{permit | deny}* {**ip-addr/prefix-length** [[{**eq** *eq-value* | **ge** *ge-value* | [**le** *le-value*]}] | **any**}

Table 8 Configure an IP Prefix List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and manually assign a sequence number for the IP prefix list statement.	*{permit | deny}*	Use the following command syntax: **seq** ***seq-num*** *{permit | deny}* {***ip-addr/prefix-length*** [[{**eq** ***eq-value*** | **ge** ***ge-value*** | [**le** ***le-value***]}] | **any**}
Assign new sequence numbers to existing entries in an IP prefix list, so that entries are in increments of 10.	*resequence ip prefix-list*	Enter this command in context configuration mode. This command is useful when you have manually assigned sequence numbers and have no room to insert new entries in between existing entries.

2.5 Configuring IPv6 Prefix Lists

To configure IPv6 prefix lists, perform the tasks described in the following sections.

2.5.1 Create an IPv6 Prefix List

To create an IPv6 prefix list, perform the tasks described in Table 9.

Table 9 Create an IPv6 Prefix List

Task	Root Command	Notes
Create an IPv6 prefix list used to filter routes and enter IPv6 prefix list configuration mode.	*ipv6 prefix-list*	Enter this command in context configuration mode. A reference to an IPv6 prefix list that does not exist, or does not contain any configured entries, implicitly matches and permits all IPv6 prefixes.

Table 9 Create an IPv6 Prefix List

Task	Root Command	Notes
Associate a description with the IPv6 prefix list.	*description (lists)*	Enter this command in IPv6 prefix list configuration mode.
Configure the IPv6 prefix list permit or deny condition.	For the complete list of tasks used to configure the IPv6 prefix list permit or deny condition, see Configure an IPv6 Prefix List Permit or Deny Condition.	—

2.5.2 Configure an IPv6 Prefix List Permit or Deny Condition

When you create several permit or deny conditions for a single IPv6 prefix list, the system can automatically sequence the entries for you, or you can manually assign a number for each entry.

When you allow the system to automatically sequence the entries for you, the system increments each statement by a count of 10. The first statement you enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you might want to add later. You can also resequence existing entries in an IPv6 prefix list.

To configure an IPv6 prefix list permit or deny condition, perform the tasks described in Table 10. Enter all commands in IPv6 prefix list configuration mode, unless otherwise noted.

Table 10 Configure an IPv6 Prefix List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and allow the system to automatically assign sequence numbers for the IPv6 prefix list statement.	*{permit | deny}*	Use the following command syntax: *{permit | deny}* {**ip-addr/prefix-length** [[{**eq** *eq-value* | **ge** *ge-value* | [**le** *le-value*]}] | **any**}

Table 10 Configure an IPv6 Prefix List Permit or Deny Condition

Task	Root Command	Notes
Permit or deny routes matching the specified criteria, and manually assign a sequence number for the IPv6 prefix list statement.	*{permit | deny}*	Use the following command syntax: **seq** ***seq-num*** *{permit | deny}* {***ip-addr/prefix-length*** [[{**eq** ***eq-value*** | **ge** ***ge-value*** | [**le** ***le-value***]}] | **any**}
Assign new sequence numbers to existing entries in an IPv6 prefix list, so that entries are in increments of 10.	*resequence ipv6 prefix-list*	Enter this command in context configuration mode. This command is useful when you have manually assigned sequence numbers and have no room to insert new entries in between existing entries.

2.6 Configuring Route Maps

When you configure route maps, you configure the route map name, and optionally, associate a description with the route map. You can also assign a sequence number to the route map, and permit or deny routes that use a specific sequence number.

After you create a route map, configure the match conditions that are looked at by the system when sending and receiving routes, and configure the set conditions that determine the action the system takes once a match for a route is found. Configure **continue** clauses to affect control flow between route map entries within the route map.

To configure route maps, perform the tasks described in the following sections.

2.6.1 Create a Route Map

To create a route map, perform the tasks described in Table 11.

Table 11 Create a Route Map

Task	Root Command	Notes
Create a route map and implement a routing policy, and enter route map configuration mode.	*route-map (routing policies)*	Enter this command in context configuration mode. You can specify a sequence number for the route map entry, relative to other route map entries in the same route map. Route map entries are tested in order of ascending sequence number. That is, the route map entry with the lowest sequence number is examined first when routes are tested. A reference to a route map that does not exist, or does not contain any configured entries, implicitly matches and permits all routes.
Configure the match condition.	For the complete list of tasks used to configure the match condition, see Configure a Match Condition.	—
Configure the set condition.	For the complete list of tasks used to configure the set condition, see Configure a Set Condition.	—
Use the **`continue`** clause to control flow within a route map.	For the complete list of tasks used to configure the **`continue`** clause, see Control Execution Flow (Continue Clause).	

2.6.2 Configure a Match Condition

To configure a match condition, perform the tasks described in Table 12. Enter all commands in route map configuration mode, unless otherwise noted.

Table 12 Configure a Match Condition

Task	Root Command	Notes
Permit or deny routes with an associated BGP AS path attribute that matches the specified BGP AS path list.	*match as-path-list*	—
Permit or deny routes with an associated BGP community attribute that matches the specified community list.	*match community-list*	—
Permit or deny routes with an associated BGP extended community attribute that matches the specified extended community list.	*match ext-community-list*	—
Permit or deny routes that have a destination IP address permitted by a specified IP prefix list.	*match ip address prefix-list*	—
Permit or deny routes with a next-hop IP address that is permitted by a specified IP prefix list.	*match ip next-hop prefix-list*	—
Permit or deny routes that have a destination IPv6 address permitted by a specified IPv6 prefix list.	*match ipv6 address prefix-list*	—
Permit or deny routes with a next-hop IPv6 address that is permitted by a specified IPv6 prefix list.	*match ipv6 next-hop prefix-list*	—
Permit or deny routes with a specific metric value.	*match metric*	—
Permit or deny routes that match a specific route type.	*match route-type*	—
Permit or deny routes that match a specific route tag value.	*match tag*	—

2.6.3 Configure a Set Condition

To configure a set condition, perform the tasks described in Table 13. Enter all commands in route map configuration mode, unless otherwise noted.

Table 13 Configure a Set Condition

Task	Root Command	Notes
Prepend an AS path to BGP routes that pass the route map conditions.	*set as-path*	The only global BGP metric available to influence the best path selection is the AS path length. Usually the local AS number is prepended multiple times, increasing the AS path length.
Set the BGP community attribute for routes that pass the route map conditions.	*set community*	A community is a group of destinations that share some common attributes. Each destination can belong to multiple communities. Up to eight communities can be specified. If the **`additive`** keyword is used, communities are added to the existing BGP community list. However, unlike AS path attributes, community attributes do not include duplicate entries.
Delete BGP communities matching the community list from the BGP community attribute for routes that pass the route map conditions.	*set community-list*	—
Set the BGP extended community attribute for routes that pass the route map conditions.	*set ext-community*	An extended community is a group of destinations that share some common attributes. Each destination can belong to multiple extended communities. Up to eight extended communities can be specified. If the **`additive`** keyword is used, extended communities are added to the existing BGP extended community list; however, unlike AS path attributes, extended community attributes do not include duplicate entries. You cannot configure the route origin at both the address family level and the BGP neighbor level. If you aggregate routes containing extended communities, the extended community information for each individual route is lost. You can apply the BGP extended communities attribute to the aggregated route using the **`set ext-community`** command.

Table 13 Configure a Set Condition

Task	Root Command	Notes
Set the BGP route dampening policy for routes that pass the route map conditions.	*set dampening*	—
Set the next-hop IP address used to forward packets for routes that pass the route map conditions.	*set ip next-hop*	—
Set the next-hop IPv6 address used to forward packets for routes that pass the route map conditions.	*set ipv6 next-hop*	—
Set the MPLS label for routes that pass the route map conditions.	*set label*	—
Set the advertisement scope for routes redistributed into Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) routing domains for routes that pass the route map conditions.	*set level*	—
Set the degree of preference for the BGP AS path for routes that pass the route map conditions.	*set local-preference*	—
Set, increment, or decrement the metric value for routes passing the route map condition.	*set metric*	—
Set the metric type for routes passing the route map condition.	*set metric-type*	—
Set the origin of the BGP path for routes that pass the route map conditions.	*set origin*	—
Set the route tag value for routes that pass the route map condition.	*set tag*	—
Set the degree of preference for BGP routes that pass the route map conditions.	*set weight*	—

2.6.4 Control Execution Flow (Continue Clause)

To control execution flow of route map entries in a route map, configure a **`continue`** clause using the tasks described in Table 14. Enter all commands in route map configuration mode, unless otherwise noted.

Table 14 Configure a Continue Clause

Task	Root Command	Notes
Continue execution with the next route map entry if all **`match`** conditions are successful. Execution continues in the next route map entry after all **`set`** clauses in the current route map entry are applied.	*continue*	If a sequence number is specified as an argument to the **`continue`** clause, execution is passed to the route map entry identified by the sequence number.

2.7 Configuring BGP Attribute-Based Accounting

Traffic index counters are maintained on interfaces with traffic index accounting enabled. Traffic indexes are associated with BGP routes based on route-maps matching on BGP attributes. When IP packets are received on an interface with traffic index accounting enabled, and the route lookup for the packet's destination IP address corresponds to a BGP route with a traffic index assigned, the corresponding byte and packet counters are incremented.

To configure BGP attribute-based accounting, perform the tasks described in Table 15.

Table 15 Configure BGP Attribute-Based Accounting

Task	Root Command	Notes
Set the traffic index value for routes that pass the route map conditions.	*set traffic-index*	Enter this command in route map configuration mode.
Assign a traffic index to routes installed for a BGP address family.	*table-map*	Enter this command in BGP address family configuration mode. To determine the attribute modifications and filtering conditions of the applied route map, use the **`route-map`** command in context configuration mode. For more information about this command, see *Configuring BGP*.
Enable BGP attribute-based accounting on an interface.	*traffic-index accounting*	Enter this command in interface configuration mode.

2.8 Configuring BGP Destination-Based QoS

BGP destination-based quality of service (QoS) provides multiple levels of service based on a customer's IP destination. BGP routes can be assigned a Differentiated Services Code Point (DSCP) value based on the BGP traffic indexing and table map features associated with route maps. This feature is useful to treat traffic differently depending on which policy it matches.

If the destination of a packet matches a BGP route configured in a route map that contains a **`set dscp`** statement, and that route map is enabled via the **`table-map`** command in BGP address family configuration mode, and the ingress interface of the packet is enabled via the **`mark dscp destination`** command in interface configuration mode, the packet is marked according to the statement defined by the **`set dscp`** statement of the route map.

To configure BGP destination-based QoS, perform the tasks described in Table 16.

Table 16 Configure BGP Destination-Based QoS

Task	Root Command	Notes
Set the DSCP value for routes that pass route map conditions.	*set dscp*	Enter this command in route map configuration mode. BGP routes can be assigned a DSCP value based on the BGP table-map route-map. When a packet is received on an interface with mark dscp destination enabled, and the packet is routed using a route with an associated DSCP, the packet's DSCP is updated and the IP header checksum is recalculated.
Assign the DSCP value to routes installed for a BGP address family.	*table-map*	Enter this command in BGP address family configuration mode. For more information about this command, see *Configuring BGP*.
Set the DSCP byte, based on BGP attributes, such as community list and autonomous AS path, for incoming IP traffic on the specified interface.	*mark dscp destination*	Enter this command in interface configuration mode. BGP destination based QoS supports setting the DSCP byte for IP traffic based on BGP attributes including community list and AS path. This can be used by a service provider (SP) to provide multiple levels of service based on a customers IP destination.

2.9 Routing Policy Operations

To manage routing policy functions, perform the appropriate tasks described in Table 17. Enter the **`show`** commands in any mode; enter the **`clear`** and **`debug`** commands in exec mode.

Table 17 Routing Policy Operations Tasks

Task	Root Command
Clear match and cache hit counts for a specified Border Gateway Protocol (BGP) autonomous system (AS) path list.	*clear as-path-list*
Clear match and cache hit counts for a specified BGP community list.	*clear community-list*
Clear match and cache hit counts for a specified BGP extended community list.	*clear ext-community-list*
Clear match and cache hit counts for a specified IP prefix list.	*clear ip prefix-list*
Clear match and cache hit counts for a specified IP Version 6 (IPv6) prefix list.	*clear ipv6 prefix-list*
Clear match and cache hit counts for a specified route map.	*clear route-map*
Enable the generation of debug messages for the maintenance and comparison of BGP AS path attributes within AS path lists.	*debug as-path-list*
Enable the generation of debug messages for the maintenance of BGP community lists and for the comparison of BGP community attributes within community lists.	*debug community-list*
Enable the generation of debug messages for the maintenance of BGP extended community lists and for the comparison of BGP extended community attributes within extended community lists.	*debug ext-community-list*
Enable the generation of debug messages for the maintenance of IP prefix lists and for the comparison of IP prefix entries to IP prefix lists.	*debug ip prefix-list*
Enable the generation of debug messages for the maintenance of IP Version 6 (IPv6) prefix lists and for the comparison of IPv6 prefix entries to IPv6 prefix lists.	*debug ipv6 prefix-list*
Enable the generation of debug messages for all configured routing policies in general.	*debug policy general*
Enable the generation of debug messages for the maintenance of route maps, and for the comparison of routes to route maps.	*debug route-map*
Display information about configured BGP AS path lists.	*show as-path-list*
Display information about configured BGP community lists.	*show community-list*
Display the routing policy configuration information for the current context.	*show configuration policy*

Table 17 Routing Policy Operations Tasks

Task	Root Command
Display information about configured BGP extended community lists.	*show ext-community-list*
Display information about configured IP prefix lists.	*show ip prefix-list*
Display information about configured IPv6 prefix lists.	*show ipv6 prefix-list*
Display information about configured route maps.	*show route-map*

3 Configuration Examples

This section provides examples of configuring simple and complex IP prefix lists, AS path lists, and community lists.

3.1 Simple IP Prefix List

The following example configures a simple IP prefix list that allows routes from networks **128.141.1.0/24**, **129.142.2.0/24**, and **130.143.3.0/24**. The last prefix list entry (sequence 40) is optional, because denial is the default action for any prefix not explicitly specified:

```
[local]Redback(config-ctx)#ip prefix-list simple-prefix-list
[local]Redback(config-prefix-list)#seq 10 permit 128.141.1.0/24
[local]Redback(config-prefix-list)#seq 20 permit 129.142.2.0/24
[local]Redback(config-prefix-list)#seq 30 permit 130.143.3.0/24
[local]Redback(config-prefix-list)#seq 40 deny 0.0.0.0/0
```

The following example applies the IP prefix list, **simple-prefix-list**, to BGP neighbor, **192.100.100.1**, as a BGP inbound route filter:

```
[local]Redback(config-ctx)#router bgp 100
[local]Redback(config-bgp)#neighbor 192.100.100.1 external
[local]Redback(config-neighbor)#address-family ipv4 unicast
[local]Redback(config-addrfamily)#prefix-list simple-prefix-list in
```

3.2 Complex IP Prefix List

This section contains an example of a more complex IP prefix list that allows routes from the following subnetworks:

- Any subnet in the class A network 10 with a prefix length greater than 16 and less than 20
- Any subnet in the class A network 11 with a prefix length exactly equal to 24

- Any subnet or host address in the class A network 12

The IP prefix list configuration is as follows:

```
[local]Redback(config-ctx)#ip prefix-list complex-prefix-list
[local]Redback(config-prefix-list)#seq 10 permit 10.0.0.0/8 ge 16 le 20
[local]Redback(config-prefix-list)#seq 20 permit 11.0.0.0/8 eq 24
[local]Redback(config-prefix-list)#seq 30 permit 12.0.0.0/8 le 32
[local]Redback(config-prefix-list)#seq 40 deny 0.0.0.0/0
```

The following example applies the **complex-prefix-list** IP prefix list to BGP neighbor, **192.100.101.5**, as a BGP outbound route filter:

```
[local]Redback(config-ctx)#router bgp 100
[local]Redback(config-bgp)#neighbor 192.100.101.5 external
[local]Redback(config-neighbor)#address-family ipv4 unicast
[local]Redback(config-addrfamily)#prefix-list complex-prefix-list out
```

3.3 Simple AS Path List

The following example configures a simple AS path list that denies BGP path attributes starting with **AS 100** or ending with **AS 200**, but allows everything else:

```
[local]Redback(config-ctx)#as-path-list simple-as-path
[local]Redback(config-as-path-list)#seq 10 deny ^100
[local]Redback(config-as-path-list)#seq 20 deny 200$
[local]Redback(config-as-path-list)#seq 30 permit any
```

The following example applies the AS path list, **simple-as-path**, to BGP neighbor, **192.100.105.10**, as a BGP inbound route filter:

```
[local]Redback(config-ctx)#router bgp 100

[local]Redback(config-bgp)#neighbor 192.100.105.10 external

[local]Redback(config-neighbor)#address-family ipv4 unicast

[local]Redback(config-addrfamily)#as-path-list simple-as-path in
```

3.4 Complex AS Path List

The AS path list example in this section denies:

- Any AS path containing a private AS number (64500–65535)
- Any AS path with AS 100, AS 200, AS 300, or AS 400 anywhere in the sequence
- Any AS path ending in AS 500 or AS 600
- Any AS path starting with 666

The AS path list configuration is as follows:

```
[local]Redback(config-ctx)#as-path-list complex-as-path

[local]Redback(config-as-path-list)#seq 10 deny _(65[0-9][0-9][0-9]
|64[5-9][0-9][0-9])_

[local]Redback(config-as-path-list)#seq 20 deny _(100|200|300|400)_

[local]Redback(config-as-path-list)#seq 30 deny (500|600)$

[local]Redback(config-as-path-list)#seq 40 deny $666

[local]Redback(config-as-path-list)#seq 50 permit any
```

The following example applies the **complex-as-path** AS path list to BGP neighbor, **192.100.106.20**, as a BGP outbound route filter:

```
[local]Redback(config-ctx)#router bgp 100

[local]Redback(config-bgp)#neighbor 192.100.106.20 external

[local]Redback(config-neighbor)#address-family ipv4 unicast

[local]Redback(config-addrfamily)#as-path-list complex-as-path out
```

3.5 Simple Community List

This following example configures a simple community list that denies community lists containing **10:10**, **20:20**, or the well-known community no-export (65535:65281), but allows any others:

```
[local]Redback(config-ctx)#community-list simple-community-list
[local]Redback(config-community-list)#seq 10 deny 10:10
[local]Redback(config-community-list)#seq 20 deny 20:20
[local]Redback(config-community-list)#seq 30 deny no-export
[local]Redback(config-community-list)#seq 40 permit any
```

3.6 Complex Community List

This section contains an example of a complex community list that denies communities with:

- **400** as the first 16 bits (the AS number) and anything for the second 16 bits of the community number
- **500** or **600** as the first 16 bits (the AS number) and **1**, **2**, or **3** as the second 16 bits of the community number
- The community that maps to the 32-bit quantity 4 billion (**4000000000**)

The community list configuration is as follows:

```
[local]Redback(config-ctx)#community-list complex-community-list
[local]Redback(config-community-list)#seq 10 deny reg-exp _400:[0-9]._
[local]Redback(config-community-list)#seq 20 deny reg-exp _(500|600):(1|2|3)_
[local]Redback(config-community-list)#seq 30 deny 4000000000
[local]Redback(config-community-list)#seq 40 permit any
```

3.7 Simple Route Map

The following protocol redistribution example configures a simple route map that sets metrics based on network destination address:

```
[local]Redback(config-ctx)#ip prefix-list select-network-20
[local]Redback(config-prefix-list)#seq 10 permit 20.0.0.0/8
[local]Redback(config-prefix-list)#exit
[local]Redback(config-ctx)#ip prefix-list select-network-30
[local]Redback(config-prefix-list)#seq 10 permit 30.0.0.0/8
[local]Redback(config-prefix-list)#exit
[local]Redback(config-ctx)#route-map proto-redist permit 10
[local]Redback(config-route-map)#match ip address prefix-list select-network-20
[local]Redback(config-route-map)#set metric 100
[local]Redback(config-route-map)#exit
[local]Redback(config-ctx)#route-map proto-redist permit 20
[local]Redback(config-route-map)#match ip address prefix-list select-network-30
[local]Redback(config-route-map)#set metric 200
```

The following example applies the **proto-redis** route map to BGP neighbor, **192.100.105.100**, as a BGP inbound route filter:

```
[local]Redback(config-ctx)#router bgp 100
[local]Redback(config-bgp)#neighbor 192.100.105.100 external
[local]Redback(config-neighbor)#address-family ipv4 unicast
[local]Redback(config-addrfamily)#route-map proto-redist in
```

3.8 Complex Route Map

This section contains an example of a complex route map that modifies communities based on AS path lists. For routes corresponding to paths containing private autonomous systems, it will set the community list attribute to the well-known community no-advertise. For routes corresponding to AS paths traversing AS 100, the communities **100:1**, **100:2**, and **100:3** are added to the BGP community list attribute. This route map and the corresponding communities can be used in conjunction with BGP.

The route map configuration is as follows:

```
[local]Redback(config-ctx)#as-path-list private-as
[local]Redback(config-as-path-list)#seq 10 permit _(65[0-9][0-9][0-9]|64[5-9][0-9]
[0-9])_
[local]Redback(config-as-path-list)#exit
[local]Redback(config-ctx)#as-path-list traverse-100
[local]Redback(config-as-path-list)#seq 10 permit _100_
[local]Redback(config-as-path-list)#exit
[local]Redback(config-ctx)#route-map modify-community permit 10
[local]Redback(config-route-map)#match as-path-list private-AS
[local]Redback(config-route-map)#set community no-advertise
[local]Redback(config-as-route-map)#exit
[local]Redback(config-ctx)#route-map modify-community permit 20
[local]Redback(config-route-map)#match as-path-list traverse-100
[local]Redback(config-route-map)#set community 100:1 100:2 100:3 additive
```

The following example applies the **modify-community** route map to BGP neighbor, **192.100.106.100**, as a BGP outbound route filter:

```
[local]Redback(config-ctx)#router bgp 100
[local]Redback(config-bgp)#neighbor 192.100.106.100 external
[local]Redback(config-neighbor)#address-family ipv4 unicast
[local]Redback(config-addrfamily)#route-map modify-community out
```

3.9 Route Map with the Continue Clause

This example shows how to use the **continue** clause to affect execution flow in a route map.

If a successful match occurs in route map entry 10, the AS path is prepended with 10 and execution continues at route map entry 30. In route map entry 30, the AS path is prepended with 12 (so that the AS path is 12 10), and the metric is set to 10.

If the match in route map entry 10 is not successful, control is passed to route map entry 20. If a successful match occurs in route map entry 20, the AS path is prepended with 11, and execution flows to route map entry 30, where the AS path is then prepended with 12 (so that the AS path is 12 11), and the metric is set to 10.

If the match in route map entry 10 is unsuccessful and the match in route map entry 20 is unsuccessful, the AS path is set to 12, and the metric is set to 10.

```
[local]Redback(config-ctx)#route-map MAP1 permit 10
[local]Redback(config-route-map)#match ip address prefix-list p1
[local]Redback(config-route-map)#set as-path prepend 10
[local]Redback(config-route-map)#continue 30
[local]Redback(config-croute-map)#exit
[local]Redback(config-ctx)#route-map MAP1 permit 20
[local]Redback(config-route-map)#match ip address prefix-list p2
[local]Redback(config-route-map)#set as-path prepend 11
[local]Redback(config-route-map)#continue
[local]Redback(config-croute-map)#exit
[local]Redback(config-ctx)#route-map MAP1 permit 30
[local]Redback(config-route-map)#set as-path prepend 12
[local]Redback(config-route-map)#set metric 10
[local]Redback(config-croute-map)#exit
```

3.10 BGP Attribute-Based Accounting

The following example configures BGP attribute-based accounting. Policies are configured to classify the routes which are to be used for BGP policy accounting, and traffic index values are set for routes that pass route map conditions. The **bgp-accounting** traffic index is assigned to routes installed for the BGP address family. BGP attribute-based accounting is enabled on the interface, **joe-customer**.

The BGP attribute-based accounting configuration is as follows:

1. Configure policies to classify the routes which are to be used for BGP attribute-based accounting.

```
[local]Redback(config-ctx)#community-list Customer04
[local]Redback(config-community-list)#seq 10 permit 200:20
[local]Redback(config-community-list)#exit
[local]Redback(config-ctx)#community-list SP-Network
[local]Redback(config-community-list)#seq 10 permit 200:30
[local]Redback(config-community-list)#exit
[local]Redback(config-ctx)#community-list SP-Services
[local]Redback(config-community-list)#seq 10 permit 200:10
[local]Redback(config-community-list)#exit
[local]Redback(config-ctx)#route-map bgp-accounting permit 10
[local]Redback(config-route-map)#match community-list SP-Services
[local]Redback(config-route-map)#set traffic-index 1
[local]Redback(config-route-map)#exit
[local]Redback(config-ctx)#route-map bgp-accounting permit 20
[local]Redback(config-route-map)#match community-list Customer04
[local]Redback(config-route-map)#set traffic-index 2
[local]Redback(config-route-map)#exit
[local]Redback(config-ctx)#route-map bgp-accounting permit 30
[local]Redback(config-route-map)#match community-list SP-Network
[local]Redback(config-route-map)#set traffic-index 3
```

2. Configure **table-map** to assign a traffic-index to routes installed for a particular BGP address family.

```
[local]Redback(config-ctx)#router bgp 1
[local]Redback(config-bgp)#address-family ipv4 unicast
[local]Redback(config-addrfamily)#table-map bgp-accounting
```

3. Enable **traffic-index accounting** on applicable interface.

```
[local]Redback(config-ctx)#interface joe-customer

[local]Redback(config-if)#ip address 10.200.1.1/30

[local]Redback(config-if)#traffic-index accounting
```

3.11 BGP Destination-Based QoS

BGP destination-based QoS supports setting the DSCP byte for IP traffic based on BGP attributes including community list and AS path. This can be used by a service provider (SP) to provide multiple levels of service based on a customers IP destination.

BGP routes can be assigned a DSCP value based on the BGP table-map route-map. When a packet is received on an interface with mark dscp destination enabled and the packet is routed using a route with associated DSCP, the packet's DSCP is updated and the IP header checksum is recalculated.

The BGP destination-based QoS configuration is as follows:

1. Configure policies to classify the routes which are to be used for BGP attribute-based accounting.

```
[local]Redback(config-ctx)#community-list Bronze-service
[local]Redback(config-community-list)#seq 10 permit 200:10
[local]Redback(config-community-list)#exit
[local]Redback(config-ctx)#community-list Silver-service
[local]Redback(config-community-list)#seq 10 permit 200:20
[local]Redback(config-community-list)#exit
[local]Redback(config-ctx)#community-list Gold-service
[local]Redback(config-community-list)#seq 10 permit 200:30
[local]Redback(config-community-list)#exit
[local]Redback(config-ctx)#route-map destination-qos permit 10
[local]Redback(config-route-map)#match community-list Gold-service
[local]Redback(config-route-map)#set dscp ef
[local]Redback(config-route-map)#exit
[local]Redback(config-ctx)#route-map destination-qos permit 20
[local]Redback(config-route-map)#match community-list Silver-service
[local]Redback(config-route-map)#set dscp af11
[local]Redback(config-route-map)#exit
[local]Redback(config-ctx)#route-map destination-qos permit 20
[local]Redback(config-route-map)#match community-list Bronze-service
[local]Redback(config-route-map)#set dscp df
```

2. Configure **table-map** to assign a DSCP to routes installed for a particular BGP address family.

```
[local]Redback(config-ctx)#router bgp 1
[local]Redback(config-bgp)#address-family ipv4 unicast
[local]Redback(config-addrfamily)#table-map destination-qos
```

3. Enable **mark dscp destination** on applicable interface.

```
[local]Redback(config-ctx)#interface jane-customer
[local]Redback(config-if)#ip address 10.200.1.1/30
[local]Redback(config-if)#mark dscp destination
```