Route Policies

In This Chapter

This chapter provides information about configuring route policies.

Topics in this chapter include:

Configuring Route Policies

Alcatel-Lucent's router supports two databases for routing information. The routing database is composed of the routing information learned by the routing protocols. The forwarding database is composed of the routes actually used to forward traffic through a router. In addition, link state databases are maintained by interior gateway protocols (IGPs) such as IS-IS and OSPF.

Routing protocols calculate the best route to each destination and place these routes in a forwarding table. The routes in the forwarding table are used to forward routing protocol traffic, sending advertisements to neighbors and peers.

A routing policy can be configured that will not place routes associated with a specific origin in the routing table. Those routes will not be used to forward data packets to the intended destinations and the routes are not advertised by the routing protocol to neighbors and peers.

Routing policies control the size and content of the routing tables, the routes that are advertised, and the best route to take to reach a destination. Careful planning is essential to implement route policies that can affect the flow of routing information or packets in and traversing through the router. Before configuring and applying a route policy, develop an overall plan and strategy to accomplish your intended routing actions.

There are no default route policies. Each policy must be created explicitly and applied to a routing protocol or to the forwarding table. Policy parameters are modifiable.

Policy Statements

Route policies contain policy statements containing ordered entries containing match conditions and actions you specify. The entries should be sequenced from the most explicit to least explicit. Packet forwarding and routing can be implemented according to your defined policies. Policy-based routing allows you to dictate where traffic can be routed, through specific paths, or whether to forward or drop the traffic. Route policies can match a given route policy entry and continue searching for other matches within either the same route policy or the next route policy.

The process can stop when the first complete match is found and executes the action defined in the entry, either to accept or reject packets that match the criteria or proceed to the next entry or the next policy. You can specify matching criteria based on source, destination, or particular properties of a route. Route policies can be constructed to support multiple stages to the evaluation and setting various route attributes. You can also provide more matching conditions by specifying criteria such as:

- Autonomous system (AS) path policy options — A combination of AS numbers and regular expression operators.
- Community list — A group sharing a common property.
- Prefix list — A named list of prefixes.
- To and From criteria — A route's source and destination.

Routing Policy Subroutines

It is possible to reference a routing policy from within a routing policy to construct powerful subroutine based policies.

A single level of policy subroutines is supported. Policy subroutines may evaluate true or false through matching and policy entry actions. A policy entry action of ‘accept’ will evaluate as true while a policy entry action of ‘reject’ will evaluate as false.

When using next-policy action state in the subroutine, the match value is defined by the default action behavior. The action is protocol-dependent. See section Default Action Behavior for more information about the default actions that are applied during packet processing.

Note—When subroutines are configured to reject routes, the accept action state can be used as a possible configuration in the subroutine match criteria to return a true-match, and the reject action state can be applied in the main policy entry that has called the subroutine.

If a match is not found during the evaluation of one or more routing policies, the final evaluation will return the accept or the reject provided by the default behavior based on the policy type (import/export) and the destination and/or source protocol.

Policy Evaluation Command

Operators can evaluate a routing policy against a BGP neighbor, routing context, or individual prefix before applying the policy to the neighbor or routing context. This command will display prefixes that are rejected by a policy and what modifications are made by a policy.

Exclusive Editing for Policy Configuration

Operators can set an exclusive lock on policy edit sessions. When the exclusive flag is set by an operator that is editing policy, other users (console or SNMP) are restricted from being able to begin, edit, commit, or abort policy. An administrative override is made available to reset the exclusive flag in the event of a session failure.

Default Action Behavior

The default action specifies how packets are to be processed when a policy related to the route is not explicitly configured. The following default actions are applied in the event that:

- A route policy does not specify a matching condition, all the routes being compared with the route policy are considered to be matches.
- A packet does not match any policy entries, then the next policy is evaluated. If a match does not occur then the last entry in the last policy is evaluated.
- If no default action is specified, the default behavior of the protocol controls whether the routes match or not.

If a default action is defined for one or more of the configured route policies, then the default action is handled as follows:

- The default action can be set to all available action states including accept, reject, next-entry, and next-policy.
- If the action states accept or reject, then the policy evaluation terminates and the appropriate result is returned.
- If a default action is defined and no matches occurred with the entries in the policy, then the default action is used.
- If a default action is defined and one or more matches occurred with the entries of the policy, then the default action is not used.

Denied IP Prefixes

The following IP address prefixes are not allowed by the routing protocols and the Route Table Manager and are not be populated within the forwarding table:

- 0.0.0.0/8 or longer
- 127.0.0.0/8 or longer
- 224.0.0.0/4 or longer
- 240.0.0.0/4 or longer

Any other prefixes that need to be filtered can be filtered explicitly using route policies.

Controlling Route Flapping

Route damping is a controlled acceptance of unstable routes from BGP peers so that any ripple effect caused by route flapping across BGP AS border routers is minimized. The motive is to delay the use of unstable routes (flapping routes) to forward data and advertisements until the route stabilizes.

Alcatel-Lucent's implementation of route damping is based on the following parameters:

- Figure of Merit — A route is assigned a Figure of Merit (FoM), which is proportional to the frequency of flaps. FoM should be able to characterize a route's behavior over a period of time.
- Route flap — A route flap is not limited to the withdrawn route. It also applies to any change in the AS path or the next hop of a reachable route. A change in AS path or next hop indicates that the intermediate AS or the route-advertising peer is not suppressing flapping routes at the source or during the propagation. Even if the route is accepted as a stable route, the data packets destined to the route could experience unstable routing due to the unstable AS path or next hop.
- Suppress threshold — The threshold is a configured value that, when exceeded, the route is suppressed and not advertised to other peers. The state is considered to be down from the perspective of the routing protocol.
- Reuse threshold — When FoM value falls below a configured reuse threshold and the route is still reachable, the route is advertised to other peers.
  The FoM value decays exponentially after a route is suppressed. This requires the BGP implementation to decay thousands of routes from a misbehaving peer.

The two events that could trigger the route flapping algorithm are:

- Route flapping — If a route flap is detected within a configured maximum route flap history time, the route's FoM is initialized and the route is marked as a potentially unstable route. Every time a route flaps, the FoM is increased and the route is suppressed if the FoM crosses the suppress threshold.
- Route reuse timer trigger — A suppressed route's FoM decays exponentially. When it crosses the reuse threshold, the route is eligible for advertisement if it is still reachable.

If the route continues to flap, the FoM, with respect to time scale, looks like a sawtooth waveform with the exponential rise and decay of FoM. To control flapping, the following parameters can be configured:

- `half-life` — The half life value is the time, expressed in minutes, required for a route to remain stable in order for one half of the FoM value to be reduced. For example, if the half life value is 6 (minutes) and the route remains stable for 6 minutes, then the new FoM

value is 3. After another 6 minutes passes and the route remains stable, the new FoM value is 1.5.

- `max-suppress` — The maximum suppression time, expressed in minutes, is the maximum amount of time that a route can remain suppressed.
- `suppress` — If the FoM value exceeds the configured integer value, the route is suppressed for use or inclusion in advertisements.
- `reuse` — If the suppress value falls below the configured `reuse` value, then the route can be reused.

Regular Expressions

The ability to perform a filter match on confederations in the AS path and/or communities is supported. This filter allows customers to configure match criteria for specific confederation sets and sequences within the AS path so that they can be filtered out before cluttering the service provider's routing information base (RIB).When matching communities, the filter allows customers to configure match criteria within the community value.

SR OS uses regular expression strings to specify match criteria for:

- An AS path string; for example, "100 200 300"
- A community string; for example, "100:200" where 100 is the AS number, and 200 is the community-value.
- Any AS path beginning with a confederation SET or SEQ containing 65001 and 65002 only: for example "< 65001 65002 >.*"
- Any AS path containing a confederation SET or SEQ, regardless of the contents: for example, ".* <.*> .*"

A regular expression is expressed in the form of terms and operators.

A term for an AS path regular expression is:

1. Regular expressions should always be enclosed in quotes.
2. An elementary term; for example, an AS number "200"
3. A range term composed of two elementary terms separated by the '-' character like "200-300".
4. The '.' dot wild-card character which matches any elementary term.
5. A regular expression enclosed in parenthesis "( )".
6. A regular expression enclosed in square brackets used to specify a set of choices of elementary or range terms; for example. [100-300 400] matches any AS number between 100 and 300 or the AS number 400.

A term for a community string regular expression is a string that is evaluated character by character and is composed of:

1. An elementary term which for a community string is any single digit like “4”.
2. A range term composed of two elementary terms separated by the ‘-’ character like “2-3”.
3. A colon ':' to delimit the AS number from the community value
4. The '.' dot wild-card character which matches any elementary term or ':'.
5. A regular expression enclosed in parenthesis “( )”.
6. A regular expression enclosed in square brackets used to specify a set of choices of elementary or range terms; for example, [51-37] matches digit 5 or any single digit between 1 and 3 or the digit 7.
7. Extended communities such as “target:” and “origin:” may consist of two regular expressions separated by the ampersand (‘&’) character. The first expression is applied to the as-value of the community string and the second to the local administrative value.

A raw hex format can be keyed to represent all extended communities. For example, "ext:0102:dc0000020032" is the same as “target:65530:20”. Hex values “ext:” and “&” can also be used to filter extended communities. The first expression is applied to the type/subtype of the extended community and the second expression to the value. In this case, hex values can also be used in the operands; for example, the value {3,f} matches a minimum 3 and a maximum 15 repetitions of the term.

The regular expression OPERATORS are listed in Table 19.

Table 19: Regular Expression Operators

Operator	Description
`|`	Matches the term on alternate sides of the pipe.
`*`	Matches multiple occurrences of the term.
`?`	Matches 0 or 1 occurrence of the term.
`+`	Matches 1 or more occurrence of the term.
`( )`	Used to parenthesize so a regular expression is considered as one term.
`[ ]`	Used to demarcate a set of elementary or range terms.
`-`	Used between the start and end of a range.
`{m,n}`	Matches least `m` and at most `n` repetitions of the term.
`{m}`	Matches exactly `m` repetitions of the term.

Table 19: Regular Expression Operators (Continued)

Operator	Description
{m,}	Matches m or more repetitions of the term.
^	Matches the beginning of the string - only allowed for communities.
$	Matches the end of the string - only allowed for communities.
\	An escape character to indicate that the following character is a match criteria and not a grouping delimiter.
<>	Matches any AS path numbers containing a confederation SET or SEQ.
&	Matches ":" between terms of a community string (applicable to extended communities origin, target, bandwidth, ext only).

Examples of "target:", "origin:" and "ext:" community strings are listed in Table 20.

Table 20: Community Strings Examples

Example Expression	Example Matches
"ext:..&f(.*)[af]$"	Matches the community "ext:0002:fffa0000001a".
'target:1&22	Matches community target:100:221.
target:^1&^22	Matches community target:100:221.
target:(.*)0$&(.*)1$	Matches community target:100:221.
origin:^1&(.*)1$	Matches community origin:100:221.

Examples of AS path and community string regular expressions are listed in Table 21.

Table 21: AS Path and Community Regular Expression Examples

AS Path to Match Criteria	Regular Expression	Example Matches
Null AS path	null[a]	Null AS path
AS path is 11	11	11
AS path is 11 22 33	11 22 33	11 22 33

Table 21: AS Path and Community Regular Expression Examples (Continued)

AS Path to Match Criteria	Regular Expression	Example Matches
Zero or more occurrences of AS number 11	`11*`	Null AS path 11 11 11 11 11 11 11 … 11
Path of any length that begins with AS numbers 11, 22, 33	`11 22 33 .*`	11 22 33 11 22 33 400 500 600
Path of any length that ends with AS numbers 44, 55, 66	`.* 44 55 66`	44 55 66 100 44 55 66 100 200 44 55 66 100 200 300 44 55 66 100 200 300 … 44 55 66
One occurrence of the AS numbers 100 and 200, followed by one or more occurrences of the number 33	`100 200 33+`	100 200 33 100 200 33 33 100 200 33 33 33 100 200 33 33 33 … 33
One or more occurrences of AS number 11, followed by one or more occurrences of AS number 22, followed by one or more occurrences of AS number 33	`11+ 22+ 33+`	11 22 33 11 11 22 33 11 11 22 22 33 11 11 22 22 33 33 11 … 11 22 … 22 33 …33

Path whose second AS number must be 11 or 22	`(. 11) | (. 22) .*`
	or
	`. (11 | 22) .*`
		100 11
		200 22 300 400
		…

Path of length one or two whose second AS number might be 11 or 22	`. (11 | 22)?`	100 200 11 300 22
Path whose first AS number is 100 and second AS number is either 11 or 22	`100 (11 | 22) .*`	100 11 100 22 200 300
Either AS path 11, 22, or 33	`[11 22 33]`	11 22 33

Table 21: AS Path and Community Regular Expression Examples (Continued)

AS Path to Match Criteria	Regular Expression	Example Matches
Range of AS numbers to match a single AS number	`10-14`	10 or 11 or 12 or 13 or 14
	`[10-12]*`	Null AS path 10 or 11 or 12 10 10 or 10 11 or 10 12 11 10 or 11 11 or 11 12 12 10 or 12 11 or 12 12 …
Zero or one occurrence of AS number 11	`11? or 11{0,1}`	Null AS path 11
One through four occurrences of AS number 11	`11{1,4}`	11 11 11 11 11 11 11 11 11 11
One through four occurrences of AS number 11 followed by one occurrence of AS number 22	`11{1,4} 22`	11 22 11 11 22 11 11 11 22 11 11 11 11 22
Path of any length, except nonexistent, whose second AS number can be anything, including nonexistent	`. .* or . .{0,}`	100 100 200 11 22 33 44 55
AS number is 100. Community value is 200.	`^100:200$`	100:200
AS number is 11 or 22. Community value is any number.	`^((11)|(22)):(.*)$`	11:100 22:100 11:200 …
AS number is 11. Community value is any number that starts with 1.	`^11:(1.*)$`	11:1 11:100 11:1100 …
AS number is any number. Community value is any number that ends with 1, 2, or 3.	`^(.*):(.*[1-3])$`	11:1 100:2002 333:55553 …

Table 21: AS Path and Community Regular Expression Examples (Continued)

AS Path to Match Criteria	Regular Expression	Example Matches
AS number is 11 or 22. Community value is any number that starts with 3 and ends with 4, 5 or 9.	`^((11)|(22)):(3.*[459])$`	11:34 22:3335 11:3777779 …
AS number is 11 or 22. Community value ends in 33 or 44.	`[^((11|22)):(.*((33)|(44)))$`	11:33 22:99944 22:555533 …
Range of Community values	`100&^([1-9][0-9]|[1-9][0-9][0-9]|1[0-9][0-9][0-9]|2000)$"`	100:10 100:11 100: … 100:2000

a.The `null` keyword matches an empty AS path.

BGP and OSPF Route Policy Support

OSPF and BGP requires route policy support. Figure 32 and Figure 34 display where route policies are evaluated in the protocol. Figure 32 depicts BGP which applies a route policy as an internal part of the BGP route selection process. Figure 34 depicts OSPF which applies routing policies at the edge of the protocol, to control only the routes that are announced to or accepted from the Route Table Manager (RTM).

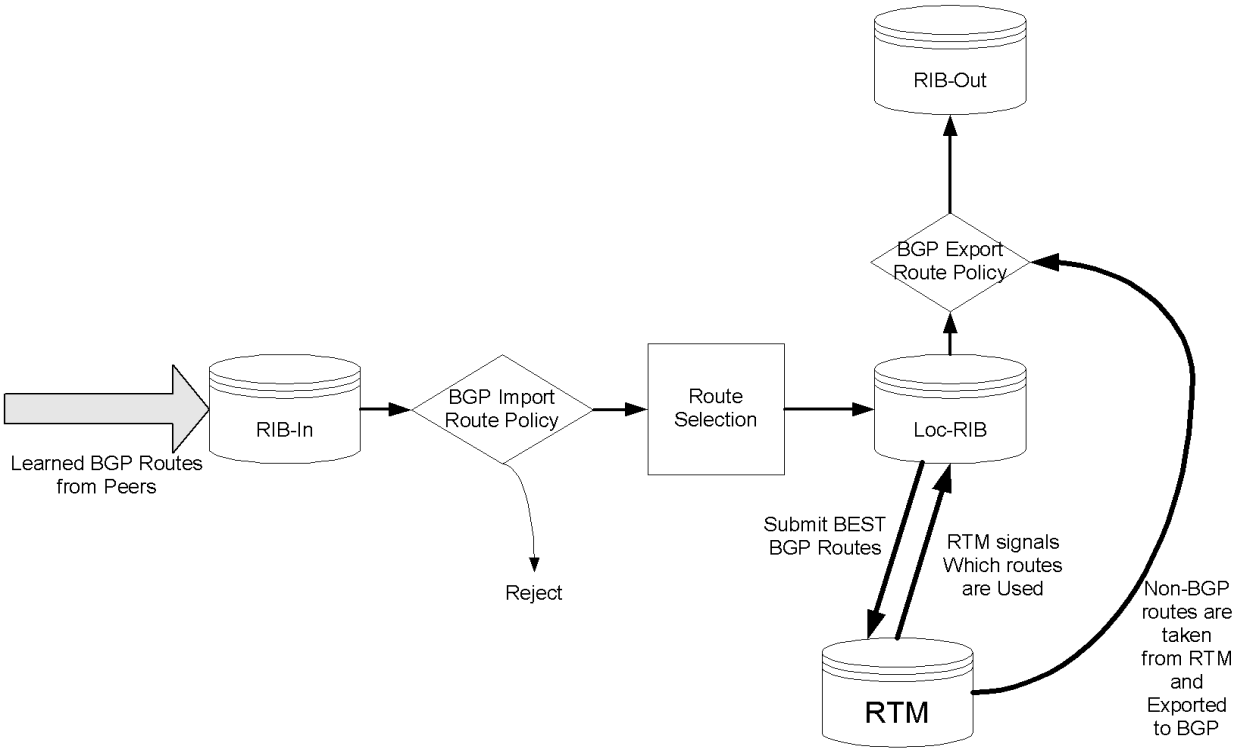

Figure 32: BGP Route Policy Diagram

BGP Route Policies

Alcatel-Lucent's implementation of BGP uses route policies extensively. The implied or default route policies can be overridden by customized route policies. The default BGP properties, with no route policies configured, behave as follows:

- Accept all BGP routes into the RTM for consideration.
- Announce all used BGP learned routes to other BGP peers
- Announce none of the IGP, static or local routes to BGP peers.

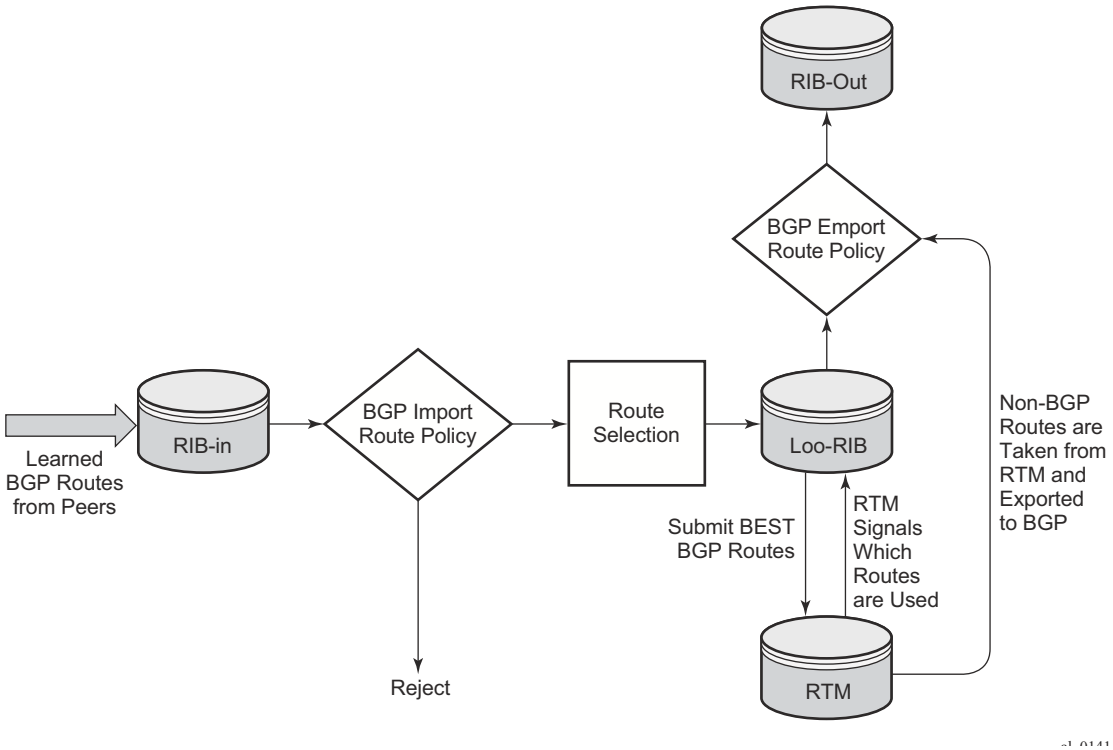

Figure 33: BGP Route Policy Diagram

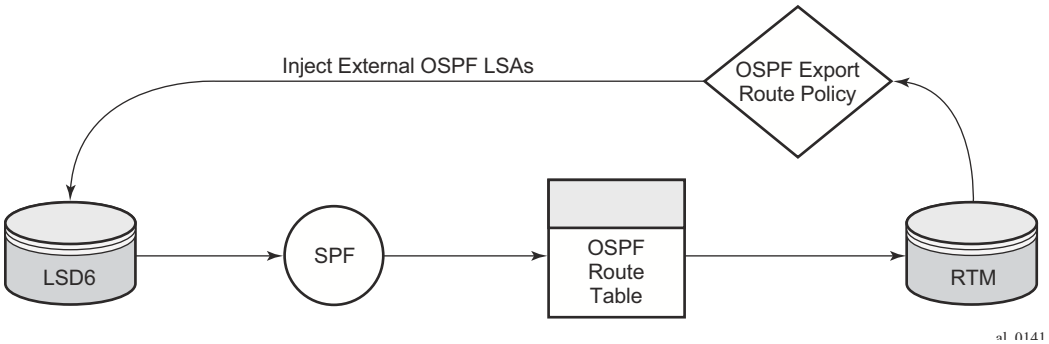

Figure 34: OSPF Route Policy Diagram

Re-advertised Route Policies

Occasionally, BGP routes may be readvertised from BGP into OSPF, IS-IS, and RIP. OSPF export policies (policies control which routes are exported to OSPF) are not handled by the main OSPF task but are handled by a separate task or an RTM task that filters the routes before they are presented to the main OSPF task.

Triggered Policies

With triggered policy enabled, deletion and re-addition of a peer after making changes to export policy causes the new updates sent out to all peers

Triggered policy is not honored if a new peer added to BGP. Update with the old policy is sent to the newly added peer. New policy does not get applied to the new peer until the peer is flapped.

With triggered policy enabled, if a new bgp/static route comes in, new addition or modification of an export policy causes the updates to sent out dynamically to all peers with the new/modified export policy

When multiple peers, say P1, P2 and P3 share the same export policy, any modifications to export policy followed by clear soft on one of the peer P1, will send out routes to P1 only according to newly modified policy.

Though routes with newly modified policy are not sent to other peers (P2, and P3) as no clear soft issues on these peers, RIB-OUT will show that new routes with modified policy are sent to all the peers. RIB-IN on peers P2 and P3 are shown correctly.

Note that with the triggered policy enabled, deletion and re-addition of a peer after making changes to export policy causes the new updates sent out to all peers.

The triggered policy is not honored if a new peer is added to BGP. An update with the old policy is sent to the newly added peer. The new policy is not applied to the new peer until the peer is flapped.

With the triggered policy enabled, if a new BGP/static route comes in, the new addition or modification of an export policy causes the updates to be sent out dynamically to all peers with the new/modified export policy.

When multiple peers, such as P1, P2 and P3, share the same export policy, any modifications to the export policy followed by **clear soft** on one of the peer P1s, will send out routes to P1 only according to the newly modified policy. Though routes with the newly modified policy are not sent to other peers (P2 and P3) as there are no clear soft issues on these peers, RIB-OUT will show that the new routes with the modified policy are sent to all the peers. RIB-IN on peers P2 and P3 are shown correctly.

Set MED to IGP Cost using Route Policies

This feature sets MED to the IGP cost of a route exported into BGP as an action in route policies. The **med-out** command in the bgp, group, and neighbor configuration context supports this option, but this method lacks per-prefix granularity. The enhanced **metric** command supported as a route

policy action supports setting MED to a fixed number, or adding, or subtracting a fixed number from the received MED, and sets IGP cost option. The enhanced **metric** {**set** {**igp** | **number1**} | { **add** | **subtract**} **number2** } command is under **config>router>policy-options>policy-statement>entry>action.**

The **metric set igp** command, when used in a BGP export policy, have the same effect as the current **med-out igp** command, except that it applies only to the routes matched by the policy entry.

The effect of the metric set igp command depends on the route type and policy type as summarized in Table 22.

Table 22: Metric Set IGP Effect

BGP Policly Type	Matched Route Type	Set Metric IGP Effect
Export	Non-BGP route (static, OSPF, ISIS, etc.)	Add MED attribute. Set value to M.
Export	BGP route w/o MED	Add MED attribute. Set value to D.
Export	BGP route with MED (value A)	Overwrite MED attribute with value D.

BGP Policy Subroutines

Currently, BGP policies only support a single level/tier of configuration that makes configuring complicated policies difficult to meet Internet transit/peering policy requirements.

This feature allows an operator to configure a policy entry that may use a sub-policy as match criteria to improve the flexibility of match criteria. Only a single level of policy nesting is supported (the sub-policy must not have any sub-policies of its own).

Route Policies for BGP Next-Hop Resolution and Peer Tracking

This feature adds the flexibility to attach a route policy to the BGP next-hop resolution process; it also allows a route policy to be associated with the optional BGP peer-tracking function. BGP next-hop resolution is a fundamental part of BGP protocol operation; it determines the best matching route (or tunnel) for the BGP next-hop address and uses information about this resolving route in the best path selection algorithm and to program the forwarding table. Attaching a policy to BGP next-hop resolution provides more control over which IP routes in the routing table can become resolving routes. Similar flexibility is also available for BGP peer-tracking, which is an optional feature that allows the session with a BGP neighbor to be taken down if there is no IP route to the neighbor address or if the best matching IP route is rejected by the policy.

Routing Policy Parameterization

Routing policy parameterization allows operators a powerful and flexible configuration approach to routing policies for policies are often reused across BGP peers of a common type (transit; peer; customer; etc).

In current modes of operation as shown in Figure 35, an operator must create individual routing policies, prefix-lists, AS-Path lists, community lists, etc for each peer despite many times the policy ultimately being the same. In this case, should an operator with 100 peers with a common policy behavior but unique policies have to make a change to entry 135 in the policy, they must do it on all policies – a significant amount of work that can result in incorrect/inconsistent policy behavior.

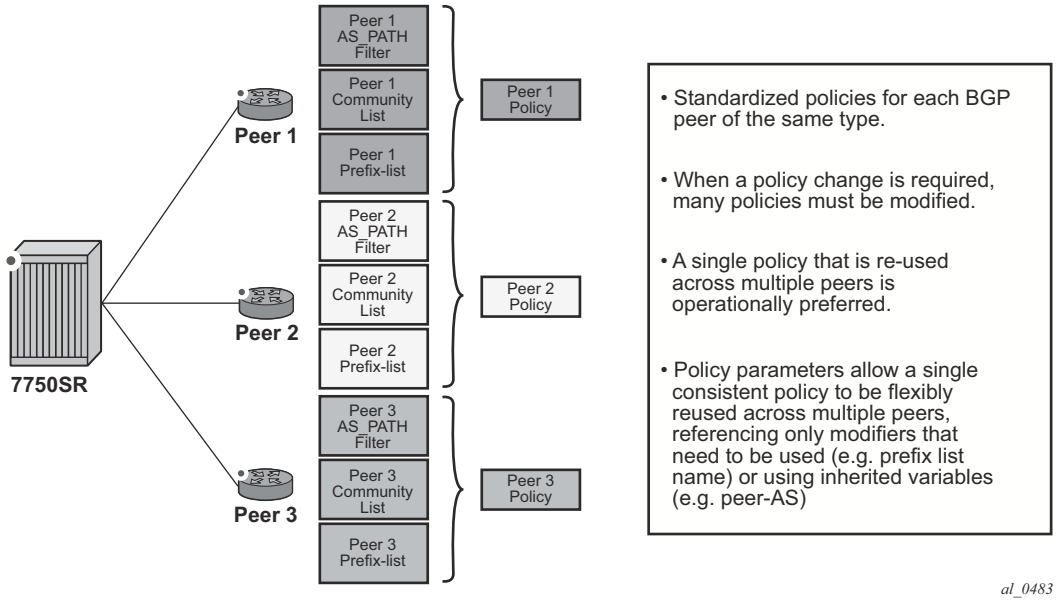

Figure 35: Route Policy Past Mode of Operation

Using a parameter based system allows an operator to have a single policy that is consistent across all peers of a type, while retaining the flexibility to reference different policy functions (prefixes, prefix-lists, community lists, etc) with unique names if required, by defining variables and the variable value. This feature will be able to inherit some of these variables directly from the routing process (for example, "@peerAS@" would be the configured BGP peer-AS).

Additionally, rather than policies being fixed and requiring many statements, the use of parameters and variables may be passed to simplify policy configuration. This reduces the number of policies

required on a peering edge router with large numbers of peers where only small amounts of configuration changes between peers, such as the ASN and prefix-list name.

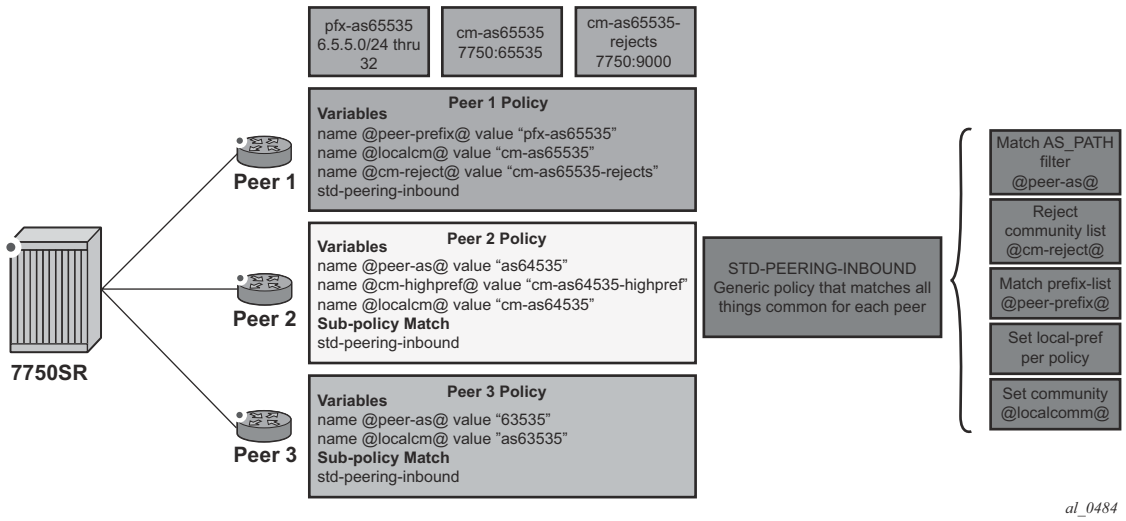

Figure 36: Route Policy Parameterization using sub-policies

Variables expansion can use two formats, with the variable name delimited by at-signs (@) at the beginning and the end.

- Standard variables must start and end with at-signs (@), for example: @variable@
- Midstring variables must be enclosed with at-signs (@) and may be midstring, for example: @variable@, start@variable@end, @variable@end, start@variable@

In Release 13.0.R1, the policy variable expansion functionality is extended with midstring variable expansion for global policy objects. Release 12.0.R1 supports policy variables contained completely in the name, such as “@localcomm@” or “@peeras@”. The following global policy objects support midstring variable expansion: **as-path**, **as-path expression**, **as-path-group**, **as-path-group expression**, **community**, and **prefix-list**.

In Release 13.0.R4, the policy variable expansion functionality is extended with more policy variables that can be expanded in subpolicy action items. The following action items support policy variables: **aigp-metric**, **as-path-prepend**, **local-preference**, **metric**, **next-hop**, **damping**, **origin**, **preference**, **tag**, **and type**. Release 12.0.R1 supports policy variables for **as-path**, **as-path-group**, **community** and **prefix-list**, and release 12.0.R4 adds policy variable support to **as-path expression** and **as-path-group expression**.

Table 23: Route Policy Variable Support in Policy Parameters

Parameter Name	Variable in Policy "from" Statement	Variable in Policy "action" Statement	Variable Format	Standard Format Release	Midstring Format Release
aigp-metric	No	Yes	Standard	13.0.R4	N/A
as-path	Yes	Yes	Midstring	12.0.R1	13.0.R1
as-path expression	Yes	No	Midstring	12.0.R4	13.0.R1
as-path-group	Yes	No	Midstring	12.0.R1	13.0.R1
as-path-group expression	Yes	No	Midstring	12.0.R4	13.0.R1
as-path-prepend	No	Yes	Standard	13.0.R4	N/A
community	Yes	Yes	Midstring	12.0.R1	13.0.R1
damping	No	Yes	Midstring	13.0.R4	13.0.R4
local-preference	No	Yes	Standard	13.0.R4	N/A
metric	No	Yes	Standard	13.0.R4	N/A
next-hop	No	Yes	Standard	13.0.R4	N/A
origin	No	Yes	Standard	13.0.R4	N/A
preference	No	Yes	Standard	13.0.R4	N/A
prefix-list	Yes	No	Midstring	12.0.R1	13.0.R1
tag	No	Yes	Standard	13.0.R4	N/A
type	No	Yes	Standard	13.0.R4	N/A

Note that midstring variables are only supported in the object name. Standard variables are supported in policy parameters inside as-path or as-path-group expression objects. For example: as-path "as" expression ".{65001,@variable@}65022".

For the definition of the variables, there are three different possible types:

- **name** *name-string* **value** *value-string*
- **name** *name-string* **address** *ip-address*
- **name** *name-string* **number** *value-number*

Depending on the parameter referenced, the correct type should be specified as follows:

- value-string: **as-path**, **as-path-group**, **community**, **prefix-list**, **damping**

- ip-address: **next-hop**
- value-number: **aigp-metric**, **as-path-prepend**, **local-preference, metric, origin**, **origin-validation**, **preference**, **tag**, **type**

The logical flow of this is to configure a per-peer policy in which the variable names and values are defined. Using Figure 36 as the example, the following configuration would be applied:

```
prefix-list "pfx-as63535"
    prefix 6.3.5.0/24 through 32
exit
prefix-list "pfx-as64535"
    prefix 6.4.5.0/24 through 32
exit
prefix-list "pfx-as65535"
    prefix 6.5.5.0/24 through 32
exit

community "cm-as63535" members "7750:63535"
community "cm-as65535" members "7750:65535"
community "cm-as64535-rejects" members "64535:14"
community "cm-as65535-rejects" members "65535:14"
community "cm-as64535-highpref" members "7777:64535"

as-path "as63535" expression "^63535$"
as-path "as64535" expression "^64535$"
as-path "as65535" expression "^65535$"

policy-statement "peer1"
    entry 10
        from
            policy-variables
                name "@localcm@" value "cm-as65535"
                name "@peer-as@" value "as65535"
                name "@cm-reject@" value "cm-as65535-rejects"
                name "@cm-highpref@" value "cm-as65535-highpref"
                name "@peer-prefix@" value "pfx-as65535"
            exit
            policy "std-peering-inbound"
        exit
        action accept
        exit
    exit
exit

policy-statement "peer2"
    description "peer2 inbound at IXP ABC, using std-peering-inbound"
    entry 10
        from
            policy-variables
                name "@localcm@" value "cm-as64535"
                name "@peer-as@" value "as64535"
                name "@cm-reject@" value "cm-as64535-rejects"
                name "@cm-highpref@" value "cm-as64535-highpref"
                name "@peer-prefix@" value "pfx-as64535"
            exit
            policy "std-peering-inbound"
        exit
        action accept
        exit
    exit
exit
```

```
policy-statement "peer3"
    description "peer3 inbound at IXP ABC, using std-peering-inbound"
    entry 10
        from
            policy-variables
                name "@localcm@" value "cm-as63535"
                name "@peer-as@" value "as63535"
                name "@cm-reject@" value "cm-as63535-rejects"
                name "@cm-highpref@" value "cm-as63535-highpref"
                name "@peer-prefix@" value "pfx-as63535"
            exit
            policy "std-peering-inbound"
        exit
        action accept
        exit
    exit
exit

policy-statement "std-peering-inbound"
    description "Standard inbound peering policy for all standard IXP peers"
    entry 10
        from
            community "@cm-reject@"
        exit
        action reject
    exit
    entry 20
        from
            prefix-list "@peer-prefix@"
            as-path "@peer-as@"
        exit
        action accept
            community add "@localcm@"
            local-preference 400
        exit
    exit
    entry 30
        from
            community "@cm-highpref@"
        exit
        action accept
            community add "@localcm@"
            local-preference 4000
        exit
    exit
exit
```

This configuration would take slightly different actions depending on the peer.

Peer 1

- Prefixes that have a community matching ‘cm-as65535-rejects’ are specifically rejected.
- Prefix list ‘pfx-as65535’ is evaluated and prefixes accepted based on that prefix-list.
- Local-preference on accepted prefixes is set to 400.
- Community ‘7750:65535’ is added to accepted prefixes.
- As community-list ‘cm-65535-highpref’ doesn’t exist, this entry is not evaluated.

Peer 2

- As community-list ‘cm-64535-rejects’ doesn’t exist, this entry is not evaluated.
- Prefix list ‘pfx-as64535’ and AS-path ‘as64535’ is evaluated and prefixes accepted based on that prefix-list and AS-path combo.
- Local-preference on accepted prefixes is set to 400.
- Community ‘7750:64535’ is added to accepted prefixes.
- Prefixes matching ‘cm-as64535-highpref’ are set to a local-preference of 4000.

Peer 3

- As community-list ‘cm-as63535-rejects’ doesn’t exist, this entry is not evaluated.
- Prefix-list ‘pfx-as63535’ and AS-path ‘as63535’ is evaluated and prefixes accepted based on that prefix-list and AS-path combo.
- Local-preference on accepted prefixes is set to 400.
- Community ‘7750:63533’ is added on accepted prefixes.
- As community-list ‘cm-63535-highpref’ doesn’t exist, this entry is not evaluated.

When to Use Route Policies

The following are examples of circumstances of when to configure and apply unique route policies.

- When you want to control the protocol to allow all routes to be imported into the routing table. This enables the routing table to learn about particular routes to enable packet forwarding and redistributing packets into other routing protocols.
- When you want to control the exporting of a protocol's learned active routes.
- When you want a routing protocol to announce active routes learned from another routing protocol, which is sometimes called *route redistribution*.
- When you want unique behaviors to control route characteristics. For example, change the route preference.
- When you want unique behaviors to control route characteristics. For example, change the route preference, AS path, or community values to manipulate the control the route selection.
- When you want to control BGP route flapping (damping).

Route Policy Configuration Process Overview

Figure 37 displays the process to provision basic route policy parameters.

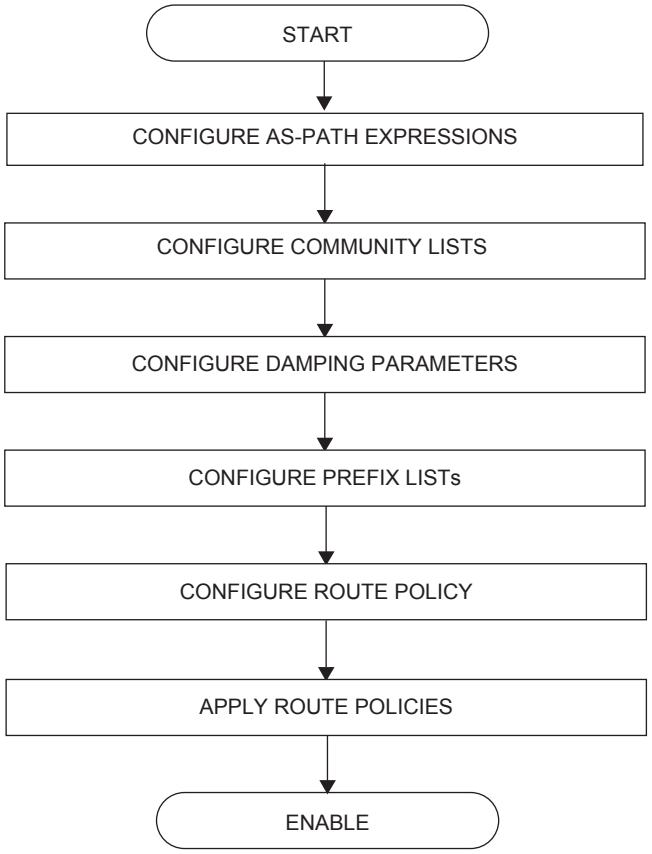

Figure 37: Route Policy Configuration and Implementation Flow

Configuration Notes

This section describes route policy configuration caveats.

General

- When configuring policy statements, the policy statement name must be unique.