

Chapter 4

Configure Routing Policy

This chapter describes the following tasks for configuring routing policies and provides the following examples:

Define Routing Policies on page 41

Apply Routing Policies on page 56

Examples: Routing Policy Configuration on page 67

Example: ISP Network Case Study on page 74

Configure the Discard Interface on page 87

Test Routing Policies on page 88

Define Routing Policies

To define a routing policy, include the policy-statement statement:

```
policy-statement policy-name {
  term term-name {
    from {
      family family-name;
      match-conditions;
      policy subroutine-policy-name;
      prefix-list name;
      route-filter destination-prefix match-type <actions>;
      source-address-filter destination-prefix match-type <actions>;
    }
    to {
      match-conditions;
      policy subroutine-policy-name;
    }
    then actions;
  }
}
```

You can include this statement at the following hierarchy levels:

[edit policy-options]

[edit logical-routers *logical-router-name* policy-options]

The following sections explain the components of the policy statement and provide configuration examples:

Routing Policy Name on page 42

Terms on page 42

Match Conditions on page 43

Actions on page 47

Examples: Define Routing Policies on page 55

Routing Policy Name

Each routing policy is introduced by the keyword `policy-statement` and a name that identifies it:

```
[edit]
policy-options {
  policy-statement policy-name {
    ...
  }
}
```

The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

Each routing policy name must be unique within a configuration.

Terms

A routing policy statement consists of one or more named terms:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      ...
    }
  }
}
```

The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (double quotes).

Each term name must be unique within a routing policy.

A policy statement can include one unnamed term. To configure an unnamed term, omit the term statement when defining match conditions and actions. The unnamed term is always the last term in the policy and cannot be moved.

Although you can use one unnamed term in each policy statement, we recommend that you name all terms.

Match Conditions

Each term can consist of two statements, from and to, that define match conditions:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        policy subroutine-policy-name;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter destination-prefix match-type <actions>;
      }
      to {
        match-conditions;
        policy subroutine-policy-name;
      }
    }
  }
}
```

In the from statement, you define the criteria that an incoming route must match. You can specify one or more match conditions. If you specify more than one, they all must match the route for a match to occur.

The from statement is optional. If you omit it and the to statement, all routes are considered to match.



NOTE: In export policies, omitting the from statement in a routing policy term might lead to unexpected results. For more information, see “Side Effects of Omitting the “from” Statement from an Export Policy” on page 63.

In the to statement, you define the criteria that an outgoing route must match. You can specify one or more match conditions. If you specify more than one, they all must match the route for a match to occur. You can specify most of the same match conditions in the to statement that you can in the from statement. In most cases, specifying a match condition in the to statement produces the same result as specifying the same match condition in the from statement.

The to statement is optional. If you omit both it and the from statement, all routes are considered to match.



NOTE: All conditions in the from and to statements must match for the action to be taken. The match conditions are effectively a logical AND operation. Matching in prefix lists and route lists is handled differently. For more information about these match conditions, including how they are evaluated, see “Configure Prefix Lists” on page 108 and “Configure Route Lists” on page 112.

Table 9 describes the match conditions available for matching an incoming or outgoing route. The table indicates whether you can use the match condition in both from and to statements and whether the match condition functions the same or differently when used with both statements. If a match condition functions differently in a from statement than in a to statement, or if the condition cannot be used in one type of statement, there is a separate description for each type of statement. Otherwise, the same description applies to both types of statements.

Table 9 also indicates whether the match condition is standard or extended. In general, the extended match conditions include criteria that are defined separately from the routing policy (autonomous system [AS] path regular expressions, communities, and prefix lists) and are more complex than standard match conditions. The extended match conditions provide many powerful capabilities. For more information about them, see “Configure Extended Match Conditions” on page 89. The standard match conditions include criteria that are defined within a routing policy and are less complex than the extended match conditions.

For examples of using the from and to statements, see “Examples: Routing Policy Configuration” on page 67.

Table 9: Routing Policy Match Conditions

Match Condition	Match Condition Category	from Statement Description	to Statement Description
aggregate-contributor	Standard	Match routes that are contributing to a configured aggregate. This match condition can be used to suppress a contributor in an aggregate route.	
area <i>area-id</i>	Standard	(Open Shortest Path First [OSPF] only) Area identifier. In a from statement used with an export policy, match a route learned from the specified OSPF area when exporting OSPF routes into other protocols.	
as-path <i>name</i>	Extended	(Border Gateway Protocol [BGP] only) Name of an AS path regular expression. For more information, see “Configure AS Path Regular Expressions” on page 89.	
as-path-group <i>group-name</i>	Extended	(BGP only) Name of an AS path group regular expression. For more information, see “Configure AS Path Regular Expressions” on page 89.	
color <i>preference</i> color2 <i>preference</i>	Standard	Color value. You can specify preference values (color and color2) that are finer-grained than those specified in the preference and preference2 match conditions. The color value can be a number in the range 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. For more information about preference values, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i> .	
community [<i>names</i>]	Extended	Name of one or more communities. If you list more than one name, only one name needs to match for a match to occur (the matching is effectively a logical OR operation). For more information, see “Configure Communities” on page 96.	
external [<i>type</i> <i>metric-type</i>]	Standard	(OSPF only) Match external routes, including routes exported from one level to another. <i>type</i> is an optional keyword. <i>metric</i> can either be 1 or 2. When you do not specify <i>type</i> , this condition matches all external routes. When you specify <i>type</i> , this condition matches only OSPF routes with the specified OSPF metric type.	
family <i>family-name</i>	Standard	Name of an address family. <i>family-name</i> can be either inet or inet6. Match the address family Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) of the route. Default setting is inet.	

Match Condition	Match Condition Category	from Statement Description	to Statement Description
instance <i>instance-name</i>	Standard	Name of one or more routing instances. Match a route learned from one of the specified instances.	Name of one or more routing instances. Match a route to be advertised over one of the specified instances.
interface <i>interface-name</i>	Standard	Name or IP address of one or more router interfaces. Do not use this qualifier with protocols that are not interface-specific, such as internal BGP (IBGP). Match a route learned from one of the specified interfaces. Direct routes match routes configured on the specified interface.	Name or IP address of one or more router interfaces. Do not use this qualifier with protocols that are not interface-specific, such as IBGP. Match a route to be advertised from one of the specified interfaces.
internal	Standard	Match a routing policy against the internal flag for simplified next-hop self policies.	
level <i>level</i>	Standard	(Intermediate System-to-Intermediate System [IS-IS] only) IS-IS level. Match a route learned from a specified level.	(IS-IS only) IS-IS level. Match a route to be advertised to a specified level.
local-preference <i>value</i>	Standard	(BGP only) BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range 0 through 4,294,967,295 ($2^{32} - 1$).	
metric <i>metric</i> metric2 <i>metric</i> metric3 <i>metric</i> metric4 <i>metric</i>	Standard	Metric value. You can specify up to four metric values, starting with metric (for the first metric value) and continuing with metric2, metric3, and metric4. (BGP only) metric corresponds to the multiple exit discriminator (MED), and metric2 corresponds to the interior gateway protocol (IGP) metric if the BGP next hop runs back through another route.	
neighbor <i>address</i>	Standard	Address of one or more neighbors (peers). For BGP, the address can be a directly connected or indirectly connected peer. For all other protocols, the address is the neighbor from which the advertisement is received.	Address of one or more neighbors (peers). For BGP import policies, specifying to neighbor produces the same result as specifying from neighbor. For BGP export policies, specifying the neighbor match condition has no effect and is ignored. For all other protocols, the to statement matches the neighbor to which the advertisement is sent. (Not valid for Routing Information Protocol [RIP].)
next-hop <i>address</i>	Standard	Next-hop address or addresses specified in the routing information for a particular route. For BGP routes, matches are performed against each protocol next hop.	
origin <i>value</i>	Standard	(BGP only) BGP origin attribute, which is the origin of the AS path information. The value can be one of the following: egp—Path information originated in another AS. igp—Path information originated within the local AS. incomplete—Path information was learned by some other means.	
policy [<i>policy-names</i>]	Extended	Name of a policy to evaluate as a subroutine. For information about this extended match condition, see “Configure Subroutines” on page 120.	

Match Condition	Match Condition Category	from Statement Description	to Statement Description
<pre> preference preference preference2 preference </pre>	Standard	<p>Preference value. You can specify a primary preference value (preference) and a secondary preference value (preference2). The preference value can be a number from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route.</p> <p>To specify even finer-grained preference values, see the <code>color</code> and <code>color2</code> match conditions in this table.</p> <p>For more information about preference values, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i>.</p>	
<pre> prefix-list name ip-addresses </pre>	Extended	<p>Named list of IP addresses. You can specify an exact match with incoming routes.</p> <p>For information about this extended match condition, see “Configure Prefix Lists” on page 108.</p>	You cannot specify this match condition.
<pre> protocol protocol </pre>	Standard	Name of the protocol from which the route was learned or to which the route is being advertised. It can be one of the following: <code>aggregate</code> , <code>bgp</code> , <code>direct</code> , <code>dvmp</code> , <code>isis</code> , <code>local</code> , <code>ospf</code> , <code>pim-dense</code> , <code>pim-sparse</code> , <code>rip</code> , <code>ripng</code> , or <code>static</code> .	
<pre> rib routing-table </pre>	Standard	<p>Name of a routing table. The value of <code>routing-table</code> can be one of the following:</p> <ul style="list-style-type: none"> <code>inet.0</code>—Unicast IPv4 routes <code>instance-name.inet.0</code>—Unicast IPv4 routes for a particular routing instance <code>inet.1</code>—Multicast IPv4 routes <code>inet.2</code>—Unicast IPv4 routes for multicast reverse-path forwarding (RPF) lookup <code>inet.3</code>—Multiprotocol Label Switching (MPLS) routes <code>mpls.0</code>—MPLS routes for label-switched path (LSP) next hops <code>inet6.0</code>—Unicast IPv6 routes 	
<pre> route-filter destination-prefix match-type < actions> </pre>	Extended	<p>List of destination prefixes. When specifying a destination prefix, you can specify an exact match with a specific route or a less precise match using match types. You can configure either a common action that applies to the entire list or an action associated with each prefix. For more information, see “Configure Route Lists” on page 112.</p>	You cannot specify this match condition.
<pre> route-type value </pre>	Standard	<p>Type of route. The value can be one of the following:</p> <ul style="list-style-type: none"> <code>external</code>—External route. <code>internal</code>—Internal route. 	

Match Condition	Match Condition Category	from Statement Description	to Statement Description
source-address-filter destination-prefix match-type < actions>	Extended	List of multicast source addresses. When specifying a source address, you can specify an exact match with a specific route or a less precise match using match types. You can configure either a common action that applies to the entire list or an action associated with each prefix. For more information, see “Configure Route Lists” on page 112.	You cannot specify this match condition.
tag string tag2 string	Standard	Tag value. You can specify two tag strings: tag (for the first string) and tag2. These values are local to the router and can be set on configured routes or by using an import routing policy. For OSPF and IS-IS, the tag and tag2 match conditions match the 32-bit tag field in external link-state advertisement (LSA) packets.	

Actions

Each term can include a then statement, which defines the actions to take if a route matches all the conditions in the from and to statements:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        policy subroutine-policy-name;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter destination-prefix match-type <actions>;
      }
      to {
        match-conditions;
        policy subroutine-policy-name;
      }
      then {
        actions;
      }
    }
  }
}
```

If a term does not have from and to statements, all routes are considered to match and the actions apply to them all.

The then statement is optional.

You can specify one or more actions. There are three types of actions:

Flow control actions, which affect whether to accept or reject the route and whether to evaluate the next term or routing policy.

Actions that manipulate route characteristics.

Trace action, which logs route matches.



NOTE: When you specify an action that manipulates the route characteristics, the changes occur in a copy of the source route. The source route itself does not change. The effect of the action is visible only after the route is imported into or exported from the routing table. To view the source route before the routing policy has been applied, use the `show route receive-protocol` command. To view a route after an export policy has been applied, use the `show route advertised-protocol` command.

During policy evaluation, the characteristics in the copy of the source route always change immediately after the action is evaluated. However, the route is not copied to the routing table or a routing protocol until the completion of the policy evaluation is complete.

If you do not include a then statement, one of the following occurs:

The next term in the routing policy, if one is present, is evaluated.

If there are no more terms in the routing policy, the next routing policy, if one is present, is evaluated.

If there are no more terms or routing policies, the accept or reject action specified by the default policy is taken. For more information, see “Default Routing Policies and Actions” on page 21.

The following section describes the following action attributes:

Flow Control Actions on page 48

Actions That Manipulate Route Characteristics on page 49

Trace Action on page 53

Final Action on page 53

Default Action on page 54

Route List Actions on page 55

Flow Control Actions

Table 10 lists the flow control actions. You can specify one of these actions along with the trace action (see “Trace Action” on page 53) or one or more of the actions that manipulate route characteristics (see “Actions That Manipulate Route Characteristics” on page 49).

Table 10: Flow Control Actions

Flow Control Action	Description
accept	Accept the route and propagate it. After a route is accepted, no other terms in the routing policy and no other routing policies are evaluated.
default-action accept	Accept and override any action intrinsic to the protocol. This is a nonterminating policy action.
reject	Reject the route and do not propagate it. After a route is rejected, no other terms in the routing policy and no other routing policies are evaluated.
default-action reject	Reject and override any action intrinsic to the protocol. This is a nonterminating policy action.
next term	Skip to and evaluate the next term in the same routing policy. Any accept or reject action specified in the then statement is skipped. Any actions in the then statement that manipulate route characteristics are applied to the route. next term is the default control action if a match occurs and you do not specify a flow control action.
next policy	Skip to and evaluate the next routing policy. Any accept or reject action specified in the then statement is skipped. Any actions in the then statement that manipulate route characteristics are applied to the route. next policy is the default control action if a match occurs, you do not specify a flow control action, and there are no further terms in the current routing policy.

Actions That Manipulate Route Characteristics

You can specify one or more of the actions listed in Table 11 to manipulate route characteristics.

Table 11: Actions That Manipulate Route Characteristics

Action	Description
as-path-prepend <i>as-path</i>	(BGP only) Affix one or more AS numbers at the beginning of the AS path. If specifying more than one AS number, include the numbers in quotation marks. The AS numbers are added after the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed with a nonconfederation sequence. For more information, see “Configure AS Path Prepend Action” on page 128.
as-path-expand last-as count <i>n</i>	(BGP only) Extract the last AS number in the existing AS path and affix that AS number to the beginning of the AS path <i>n</i> times, where <i>n</i> is a number from 1 through 32. The AS number is added before the local AS number has been added to the path. This action adds AS numbers to AS sequences only, not to AS sets. If the existing AS path begins with a confederation sequence or set, the affixed AS numbers are placed within a confederation sequence. Otherwise, the affixed AS numbers are placed within a nonconfederation sequence. This option is typically used in non-IBGP export policies.
class <i>class-name</i>	(Class of service [CoS] only) Apply the specified class-of-service parameters to routes installed into the routing table. For more information, see the <i>JUNOS Internet Software Network Interfaces and Class of Service Configuration Guide</i> .
color <i>preference</i> color2 <i>preference</i>	Set the preference value to the specified value. The color and color2 preference values are even more fine-grained than those specified in the preference and preference2 actions. The color value can be a number in the range 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route. If you set the preference with the color action, the value is internal to the JUNOS software and is not transitive. For more information about preference values, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i> .

Action	Description
color (add subtract) <i>number</i> color2 (add subtract) <i>number</i>	Change the color preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
community (+ add) [<i>names</i>]	(BGP only) Add the specified communities to the set of communities in the route. For more information, see “Configure Communities” on page 96.
community (- delete) [<i>names</i>]	(BGP only) Delete the specified communities from the set of communities in the route. For more information, see “Configure Communities” on page 96.
community (= set) [<i>names</i>]	(BGP only) Replace any communities that were in the route in with the specified communities. For more information, see “Configure Communities” on page 96.
damping <i>name</i>	(BGP only) Apply the specified route-damping parameters to the route. These parameters override the default damping parameters. This action is useful only in an import policy, because the damping parameters affect the state of routes in the routing table. To apply damping parameters, you must enable BGP flap damping as described in the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i> , and you must create a named list of parameters as described in “Configure Damping Action” on page 129.
destination-class <i>destination-class-name</i>	Maintain packet counts for a route passing through your network, based on the destination address in the packet. You can do the following: Configure group destination prefixes by configuring a routing policy; see “Define Routing Policies” on page 41 and “Examples: Routing Policy Configuration” on page 67. Apply that routing policy to the forwarding table with the corresponding destination class; see “Apply Routing Policies to the Forwarding Table” on page 65. For more information about the forwarding-table configuration statement, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i> . Enable packet counting on one or more interfaces by including the destination-class-usage statement at the [edit interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family inet accounting] hierarchy level (see the <i>JUNOS Internet Software Network Interfaces and Class of Service Configuration Guide</i>). See “Examples: Routing Policy Configuration” on page 67. View the output by using one of the following commands: show interfaces <i>interface-name</i> destination-class <i>destination-class-name</i> , show interfaces <i>interface-name</i> extensive, or show interfaces <i>interface-name</i> statistics (see the <i>JUNOS Internet Software Network and Services Interfaces Command Reference</i>). To configure a packet count based on the source address, use the source-class statement described in this table.
external type <i>metric</i>	Set the external metric type for routes exported by OSPF. You must specify the keyword type.
forwarding-class <i>forwarding-class-name</i>	Create the forwarding-class which includes packets based on both the destination address and the source address in the packet. You can do the following: Configure group prefixes by configuring a routing policy; see “Define Routing Policies” on page 41 and “Examples: Routing Policy Configuration” on page 67. Apply that routing policy to the forwarding table with the corresponding forwarding class; see “Apply Routing Policies to the Forwarding Table” on page 65. For more information about the forwarding-table configuration statement, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i> . Enable packet counting on one or more interfaces by using the procedure described in either the destination-class or source-class actions defined in this table.

Action	Description
install-nexthop <strict> lsp <i>lsp-name</i>	Choose which next hops, among a set of equal LSP next hops, are installed in the forwarding table. Use the export policy for the forwarding table to specify the LSP next hop to be used for the desired routes. Specify the strict option to enable strict mode, which checks to see if any of the LSP next hops specified in the policy are up. If none of the specified LSP next hops are up, the policy installs the discard next hop.
load-balance per-packet	(For export to the forwarding table only) Install all next-hop addresses into the forwarding table and have the forwarding table perform per-packet load balancing. For more information, see “Configure Load-Balance Per-Packet Action” on page 134.
local-preference <i>value</i>	(BGP only) Set the BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$).
local-preference (add subtract) <i>number</i>	Change the local preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value. For BGP, if the attribute value is not known, it is initialized to 100 before the routing policy is applied.
metric <i>metric</i> metric2 <i>metric</i> metric3 <i>metric</i> metric4 <i>metric</i>	Set the metric. You can specify up to four metric values, starting with metric (for the first metric value) and continuing with metric2, metric3, and metric4. (BGP only) metric corresponds to the MED, and metric2 corresponds to the IGP metric if the BGP next hop loops through another router.
metric (add subtract) <i>number</i> metric2 (add subtract) <i>number</i> metric3 (add subtract) <i>number</i> metric4 (add subtract) <i>number</i>	Change the metric value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.
metric (igp minimum-igp) <i>site-offset</i>	(BGP only) Change the metric (MED) value by the specified negative or positive offset. This action is useful only in an external BGP (EBGP) export policy.
next-hop (<i>address</i> peer-address)	Set the next hop. When the advertising protocol is BGP, you can set the next hop only when any third-party next hop can be advertised; that is, when using IBGP or EBGP confederations. If you specify <i>address</i> as self, the next-hop address is replaced by one of the local router’s addresses. The advertising protocol determines which address to use. When the advertising protocol is BGP, this address is set to the local IP address used for the BGP adjacency. A router cannot install routes with itself as the next hop. If you specify peer-address, the next-hop address is replaced by the peer’s IP address. This option is valid only in import policies. Primarily used by BGP to enforce using the peer’s IP address for advertised routes, this option is meaningful only when the next hop is the advertising router or another directly connected router.
origin <i>value</i>	(BGP only) Set the BGP origin attribute to one of the following values: igp—Path information originated within the local AS. egp—Path information originated in another AS. incomplete—Path information was learned by some other means.

Action	Description
<pre>preference preference preference2 preference</pre>	<p>Set the preference value. You can specify a primary preference value (preference) and a secondary preference value (preference2). The preference value can be a number in the range from 0 through 4,294,967,295 ($2^{32} - 1$). A lower number indicates a more preferred route.</p> <p>To specify even finer-grained preference values, see the color and color2 actions in this table.</p> <p>If you set the preference with the preference action, the new preference remains associated with the route. The new preference is internal to the JUNOS software and is not transitive.</p> <p>For more information about preference values, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i>.</p>
<pre>preference (add subtract) number preference2 (add subtract) number</pre>	<p>Change the preference value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.</p>
<pre>source-class source-class-name</pre>	<p>Maintain packet counts for a route passing through your network, based on the source address. You can do the following:</p> <ul style="list-style-type: none"> Configure group source prefixes by configuring a routing policy; see “Define Routing Policies” on page 41 and “Examples: Routing Policy Configuration” on page 67. Apply that routing policy to the forwarding table with the corresponding source class; see “Apply Routing Policies to the Forwarding Table” on page 65. For more information about the forwarding-table configuration statement, see the <i>JUNOS Internet Software Routing Protocols Configuration Guide</i>. Enable packet counting on one or more interfaces by including the source-class-usage statement at the [edit interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family inet accounting] hierarchy level (see the <i>JUNOS Internet Software Network Interfaces and Class of Service Configuration Guide</i>). Also, follow the source-class-usage statement with the input or output statement to define the inbound and outbound interfaces on which traffic monitored for source-class usage (SCU) is arriving and departing (or define one interface for both). The complete syntax is [edit interfaces <i>interface-name</i> unit <i>unit-number</i> family inet accounting source-class-usage (input output input output)]. See the example in “Examples: Routing Policy Configuration” on page 67. View the output by using one of the following commands: show interfaces <i>interface-name</i> source-class <i>source-class-name</i>, show interfaces <i>interface-name</i> extensive, or show interfaces <i>interface-name</i> statistics (see the <i>JUNOS Internet Software Network and Services Interface Command Reference</i>). <p>To configure a packet count based on the destination address, use the destination-class statement described in this table.</p>
<pre>tag tag tag2 tag</pre>	<p>Set the tag value. You can specify two tag strings: tag (for the first string) and tag2. These values are local to the router.</p> <p>(For OSPF and IS-IS only) The tag and tag2 actions set the 32-bit tag field in OSPF external link-state advertisement (LSA) packets, and the 32-bit flag in the IS-IS IP prefix type length values (TLV).</p>
<pre>tag (add subtract) number tag2 (add subtract) number</pre>	<p>Change the tag value by the specified amount. If an addition operation results in a value that is greater than 4,294,967,295 ($2^{32} - 1$), the value is set to $2^{32} - 1$. If a subtraction operation results in a value less than 0, the value is set to 0. If an attribute value is not already set at the time of the addition or subtraction operation, the attribute value defaults to a value of 0 regardless of the amount specified. If you perform an addition to an attribute with a value of 0, the number you add becomes the resulting attribute value.</p>

Trace Action

If you specify the trace action, the match is logged to a trace file. To set up a trace file, you must specify the following elements in the global traceoptions statement:

- Trace filename
- policy option in the flag statement

For more information about the global traceoptions statement, see the *JUNOS Internet Software Routing Protocols Configuration Guide*.

The following example uses the trace filename of policy-log:

```
[edit]
routing-options {
  traceoptions {
    file "policy-log";
    flag policy;
  }
}
```

This action does not affect the flow control during routing policy evaluation.

If a term that specifies a trace action also specifies a flow control action, the name of the term will be logged in the trace file. If a term specifies a trace action only, the word < default> only will be logged.

Final Action

In addition to specifying an action using the then statement in a named term, you can also specify an action using the then statement in an unnamed term, as follows:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        policy subroutine-policy-name;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter destination-prefix match-type <actions>;
      }
      to {
        match-conditions;
        policy subroutine-policy-name;
      }
      then {
        actions;
      }
    }
  }
  then action;
}
```

Default Action

This default-action statement overrides any action intrinsic to the protocol. This action is also nonterminating, so that various policy terms can be evaluated before the policy is terminated. You can specify a default action, either accept or reject, as follows:

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        family family-name;
        match-conditions;
        policy subroutine-policy-name;
        prefix-list name;
        route-filter destination-prefix match-type <actions>;
        source-address-filter destination-prefix match-type <actions>;
      }
      to {
        match-conditions;
        policy subroutine-policy-name;
      }
      then {
        actions;
        default-action (accept | reject);
      }
    }
  }
}
```

The resulting action is set either by the protocol or by the last policy term that is met.

Example: Configure the Default Action for a Policy

Configure a routing policy that matches routes based on three policy terms. If the route matches the first term, a certain community tag is attached. If the route matches two separate terms, then both community tags are attached. If the route does not match any terms, it is rejected (protocol's default action). Note that the terms hub and spoke are mutually exclusive.

```
[edit]
policy-options {
  policy-statement test {
    term set-default {
      then default-action reject;
    }
    term hub {
      from interface ge-2/1/0.5;
      then {
        community add test-01-hub;
        default-action accept;
      }
    }
  }
}
```

```

term spoke {
  from interface [ ge-2/1/0.1 ge-2/1/0.2 ];
  then {
    community add test-01-spoke;
    default-action accept;
  }
}
term management {
  from protocol direct;
  then {
    community add management;
    default-action accept;
  }
}
}

```

Route List Actions

If you specify route lists in the from statement, for each route in the list, you can specify an action to take on that individual route directly, without including a then statement. For more information, see “Configure Route Lists” on page 112.

Examples: Define Routing Policies

This section provides examples of defining routing policies. For more examples, see “Examples: Routing Policy Configuration” on page 67.

Define a Routing Policy from BGP to IS-IS

Accept BGP routes advertised by the peer 192.168.1.1. If a route matches, it is accepted, and no further evaluation is performed on that route. If a route does not match, the accept or reject action specified by the default policy is taken. (For more information about the default routing policies, see “Default Routing Policies and Actions” on page 21.) If you apply this routing policy to imported BGP routes, only the routes learned from the peer 192.168.1.1 and BGP transit routes are accepted from BGP peers.

```

[edit]
policy-options {
  policy-statement bgp-to-isis {
    term term1 {
      from {
        neighbor 192.168.1.1;
      }
      then {
        accept;
      }
    }
  }
}

```

Use Routing Policy to Set Preference

Define a routing policy that matches routes from specific next hops that are being advertised to specific neighbors and that sets the preference. If a route does not match the first term, it is evaluated by the second term. If it still does not match, the next routing policy, if configured, is evaluated; then the accept or reject action specified by the default policy is taken. (For more information about the default routing policies, see “Default Routing Policies and Actions” on page 21.)

```
[edit]
policy-options {
  policy-statement set-preference {
    term term1 {
      from {
        next-hop [10.0.0.1 10.0.0.2];
      }
      to {
        neighbor 192.168.1.1;
      }
      then {
        preference 10;
      }
    }
    term term2 {
      from {
        next-hop 10.0.0.3;
      }
      to {
        neighbor 192.168.1.1;
      }
      then {
        preference 15;
      }
    }
  }
}
```

Apply Routing Policies

For a routing policy to take effect, you must apply it to either a routing protocol or the forwarding table. This section contains the following information:

Apply Routing Policies to a Routing Protocol on page 56

Apply Routing Policies to the Forwarding Table on page 65

Examples: Apply Routing Policies on page 65

Apply Routing Policies to a Routing Protocol

Before applying routing policies to routing protocols, you must know if each protocol supports import and export policies and the level at which you can apply these policies. Table 8 on page 28 summarizes the import and export policy support for each routing protocol and the level at which you can apply these policies.

For more information about applying routing policies to individual routing protocols, see the *JUNOS Internet Software Routing Protocols Configuration Guide*.

This section describes the following tasks for applying routing policies to a routing protocol and some side effects of these tasks:

Apply a Routing Policy on page 57

Apply a Routing Policy Chain on page 57

Apply Policy Expressions on page 58

Side Effects of Omitting the “from” Statement from an Export Policy on page 63

Apply a Routing Policy

To apply a routing policy to a routing protocol, include the import and export statements:

```
import [ policy-names ];
export [ policy-names ];
```

You can include the statements at the following hierarchy levels:

```
[edit protocols protocol-name]
```

```
[edit logical-routers logical-router-name protocols protocol-name]
```

In the import statement, list the name of the routing policy to be evaluated when routes are imported into the routing table from the routing protocol.

In the export statement, list the name of the routing policy to be evaluated when routes are being exported from the routing table into a dynamic routing protocol. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or different import and export statements.

For information about how the policy framework software evaluates a routing policy, see “How a Routing Policy Is Evaluated” on page 30.

Apply a Routing Policy Chain

To apply multiple routing policies (chains) to a routing protocol, include the import and export statements:

```
import [ policy-names ];
export [ policy-names ];
```

You can include the statements at the following hierarchy levels:

```
[edit protocols protocol-name]
```

```
[edit logical-routers logical-router-name protocols protocol-name]
```

In the import statement, list the names of multiple routing policies to be evaluated when routes are imported into the routing table from the routing protocol.

In the export statement, list the names of multiple routing policies to be evaluated when routes are being exported from the routing table into a dynamic routing protocol. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or different import and export statements.

The policy framework software evaluates the routing policies sequentially, from left to right. If an action specified in one of the policies manipulates a route characteristic, the policy framework software carries the new route characteristic forward during the evaluation of the remaining policies. For example, if the action specified in the first policy of a chain sets a route’s metric to 500, this route matches the criterion of metric 500 defined in the next policy.

For more information about routing policy chain evaluation, see “How a Routing Policy Chain Is Evaluated” on page 31.

Apply Policy Expressions

Policy expressions give the policy framework software a different way to evaluate routing policies. A *policy expression* uses Boolean logical operators with policies. The logical operators establish rules by which the policies are evaluated.

During evaluation of a routing policy in a policy expression, the policy action of accept, reject, or next policy is converted to the value of TRUE or FALSE. This value is then evaluated against the specified logical operator to produce output of either TRUE or FALSE. The output is then converted back to a flow control action of accept, reject, or next policy. The result of the policy expression is applied as it would be applied to a single policy; the route is accepted or rejected and the evaluation ends, or the next policy is evaluated.

Table 12 summarizes the policy actions and their corresponding TRUE and FALSE values and flow control action values. Table 13 describes the logical operators. For complete information about policy expression evaluation, see “How a Policy Expression Is Evaluated” on page 60.

You must enclose a policy expression in parentheses. You can place a policy expression anywhere in the import or export statements and in the from policy statement.

Table 12: Policy Action Conversion Values

Policy Action	Conversion Value	Flow Control Action Conversion Value
Accept	TRUE	Accept
Reject	FALSE	Reject
Next policy	TRUE	Next policy

Table 13: Policy Expression Logical Operators

Logical Operator	Policy Expression Logic	How Logical Operator Affects Policy Expression Evaluation
&& (Logical AND)	<p>Logical AND requires that all values must be TRUE to produce output of TRUE.</p> <p>Routing policy value of TRUE and TRUE produces output of TRUE. Value of TRUE and FALSE produces output of FALSE. Value of FALSE and FALSE produces output of FALSE.</p>	<p>If the first routing policy returns the value of TRUE, the next policy is evaluated. If the first policy returns the value of FALSE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated.</p>
(Logical OR)	<p>Logical OR requires that at least one value must be TRUE to produce output of TRUE.</p> <p>Routing policy value of TRUE and FALSE produces output of TRUE. Value of TRUE and TRUE produces output of TRUE. Value of FALSE and FALSE produces output of FALSE.</p>	<p>If the first routing policy returns the value of TRUE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If the first policy returns the value of FALSE, the next policy is evaluated.</p>
! (Logical NOT)	<p>Logical NOT reverses value of TRUE to FALSE and of FALSE to TRUE. It also reverses the actions of accept and next policy to reject, and reject to accept.</p>	<p>If used with the logical AND operator and the first routing policy value of FALSE is reversed to TRUE, the next policy is evaluated. If the value of TRUE is reversed to FALSE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated.</p> <p>If used with the logical OR operator and the first routing policy value of FALSE is reversed to TRUE, the evaluation of the expression ends and subsequent policies in the expression are not evaluated. If the value of TRUE is reversed to FALSE, the next policy is evaluated.</p> <p>If used with a policy and the flow control action is accept or next policy, these actions are reversed to reject. If the flow control action is reject, this action is reversed to accept.</p>

Policy Expression Examples

The following examples show how to use the logical operators to create policy expressions:

Logical AND—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, policy2 is evaluated. If a value of FALSE is returned, policy2 is not evaluated.

```
export (policy1 && policy2)
```

Logical OR—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, policy2 is not evaluated. If a value of FALSE is returned, policy2 is evaluated.

```
export (policy1 || policy2)
```

Logical OR and logical AND—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, policy2 is skipped and policy3 is evaluated. If after evaluating policy1, a value of FALSE is returned, policy2 is evaluated.

If policy2 returns a value of TRUE, policy3 is evaluated. If policy2 returns a value of FALSE, policy3 is not evaluated.

```
export [(policy1 || policy2) && policy3]
```

Logical NOT—In the following example, policy1 is evaluated first. If after evaluating policy1, a value of TRUE is returned, the value is reversed to FALSE and policy2 is not evaluated. If a value of FALSE is returned, the value is reversed to TRUE and policy2 is evaluated.

```
export (!policy1 && policy2)
```

The sequential list [policy1 policy2 policy3] is not the same as the policy expression (policy1 && policy2 && policy3).

The sequential list is evaluated on the basis of a route matching a routing policy. For example, if policy1 matches and the action is accept or reject, policy2 and policy3 are not evaluated. If policy1 does not match, policy2 is evaluated and so on until a match occurs and the action is accept or reject.

The policy expressions are evaluated on the basis of the action in a routing policy that is converted to the value of TRUE or FALSE and the logic of the specified logical operator. (For complete information about policy expression evaluation, see “How a Policy Expression Is Evaluated” on page 60.) For example, if policy1 returns a value of FALSE, policy2 and policy3 are not evaluated. If policy1 returns a value of TRUE, policy2 is evaluated. If policy2 returns a value of FALSE, policy3 is not evaluated. If policy2 returns a value of TRUE, policy3 is evaluated.

You can also combine policy expressions and sequential lists. In the following example, if policy1 returns a value of FALSE, policy2 is evaluated. If policy2 returns a value of TRUE and contains a next policy action, policy3 is evaluated. If policy2 returns a value of TRUE but does not contain an action, including a next policy action, policy3 is still evaluated (because if you do not specify an action, next term or next policy are the default actions). If policy2 returns a value of TRUE and contains an accept action, policy3 is not evaluated.

```
export [(policy1 || policy2) policy3]
```

How a Policy Expression Is Evaluated

During evaluation, the policy framework software converts policy actions to values of TRUE or FALSE, which are factors in determining the flow control action that is performed upon a route. However, the software does not actually perform a flow control action on a route until it evaluates an entire policy expression.

The policy framework software evaluates a policy expression as follows:

1. The software evaluates a route against the first routing policy in a policy expression and converts the specified or default action to a value of TRUE or FALSE. (For information about the policy action conversion values, see Table 12 on page 58.)
2. The software takes the value of TRUE or FALSE and evaluates it against the logical operator used in the policy expression (see Table 13 on page 59). Based upon the logical operator used, the software determines whether or not to evaluate the next routing policy, if one is present.

The policy framework software uses a method of shortcut evaluation. When a result is certain, the software stops evaluating subsequent routing policies in the policy expression. For example, if the policy expression specifies logical AND and the evaluation of the first routing policy returns the value of FALSE, the software determines that the output will be FALSE no matter what the values of the unevaluated routing policies are. Therefore, the software does not evaluate the subsequent routing policies in this policy expression.

3. The software performs Steps 1 and 2 for each subsequent routing policy in the policy expression, if they are present and if the software has determined that it is appropriate to evaluate them.
4. After evaluating the last routing policy, if it is appropriate, the software evaluates the value of TRUE or FALSE obtained from each routing policy evaluation. Based upon the logical operator used, it calculates an output of TRUE or FALSE.
5. The software converts the output of TRUE or FALSE back to an action. (For information about the policy action conversion values, see Table 12 on page 58.) The action is performed.

If each policy in the expression returned a value of TRUE, the software converts the output of TRUE back to the flow control action specified in the last policy. For example, if the policy expression (policy1 && policy2) is specified and policy1 specifies accept and policy2 specifies next term, the next term action is performed.

If an action specified in one of the policies manipulates a route characteristic, the policy framework software carries the new route characteristic forward during the evaluation of the remaining policies. For example, if the action specified in the first policy of a policy expression sets a route's metric to 500, this route matches the criteria of metric 500 defined in the next policy. However, if a route characteristic manipulation action is specified in a policy located in the middle or the end of a policy expression, it is possible, because of the shortcut evaluation, that the policy is never evaluated and the manipulation of the route characteristic never occurs.

Example: Evaluating Policy Expressions

The following sample routing policy uses three policy expressions:

```
[edit]
policy-options {
  policy-statement policy-A {
    from {
      route-filter 10.10.0.0/16 orlonger;
    }
    then reject;
  }
}

policy-options {
  policy-statement policy-B {
    from {
      route-filter 10.20.0.0/16 orlonger;
    }
    then accept;
  }
}

protocols {
  bgp {
    neighbor 192.168.1.1;
    export (policy-A && policy-B);
  }
  neighbor 192.168.2.1;
  export (policy-A || policy-B);
}
  neighbor 192.168.3.1;
  export (!policy-A);
}
}
```

The policy framework software evaluates the transit BGP route 10.10.1.0/24 against the three policy expressions specified in the sample routing policy as follows:

(policy-A && policy-B)—10.10.1.0/24 is evaluated against policy-A. 10.10.1.0/24 matches the route list specified in policy-A, so the specified action of reject is returned. reject is converted to a value of FALSE, then FALSE is evaluated against the specified logical AND. Because the result of FALSE is certain no matter what the results of the evaluation of policy-B are (in policy expression logic, any result AND a value of FALSE produces the output of FALSE), policy-B is not evaluated and the output of FALSE is produced. The FALSE output is converted to “reject,” and 10.10.1.0/24 is rejected.

(policy-A || policy-B)—10.10.1.0/24 is evaluated against policy-A. 10.10.1.0/24 matches the route list specified in policy-A, so the specified action of reject is returned. reject is converted to a value of FALSE, then FALSE is evaluated against the specified logical OR. Because logical OR requires at least one value of TRUE to produce an output of TRUE, 10.10.1.0/24 is evaluated against policy-B. 10.10.1.0/24 does not match policy-B, so the default action of “next policy” is returned. “Next policy” is converted to a value of TRUE, then the

value of FALSE (for policy-A evaluation) and TRUE (for policy-B evaluation) are evaluated against the specified logical OR. In policy expression logic, FALSE OR TRUE produce an output of TRUE. The output of TRUE is converted to “next policy.” (TRUE is converted to “next policy” because “next policy” was the last action retained by the policy framework software.) policy-B is the last routing policy in the policy expression, so the action specified by the default export policy for BGP, “accept,” is taken.

(!policy-A)—10.10.1.0/24 is evaluated against policy-A. 10.10.1.0/24 matches the route list specified in policy-A, so the specified action of reject is returned. reject is converted to a value of FALSE, then FALSE is evaluated against the specified logical NOT. The value of FALSE is reversed to an output of TRUE based on the rules of logical NOT. The output of TRUE is converted to “accept,” and route 10.10.1.0/24 is accepted.

Side Effects of Omitting the “from” Statement from an Export Policy

In export policies, omitting the from statement in a term might lead to unexpected results. By default, if you omit the from statement, all routes are considered to match. For example, static and direct routes are not exported by BGP by default. However, if you create a term with an empty from statement, these routes inadvertently could be exported because they matched the from statement. For example, the following routing policy is designed to reject a few route ranges and then export routes learned by BGP (which is the default export behavior):

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  bgp {
    group 4 {
      export statics-policy;
      type external;
      peer-as 47;
      neighbor 192.168.1.1;
    }
  }
}
policy-options {
  policy-statement statics-policy {
    term term1 {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 172.16.1.1/3 orlonger;
      }
      then reject;          # reject the prefixes in the route list
    }
    term term2 {
      then {
        accept;           # accept all other routes, including static and direct routes
      }
    }
  }
}
```

However, this routing policy results in BGP advertising static and direct routes to its peers because:

term1 rejects the destination prefixes enumerated in the route list.

term2, because it has no from statement, matches all other routes, including static and direct routes, and accepts all these routes (with the accept statement).

To modify the routing policy shown above so that an IGP does not export unwanted routes, you can specify the following additional terms:

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  isis {
    export statics-policy;
  }
}
policy-options {
  policy-statement statics-policy {
    term term1 {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 172.16.1.1/3 orlonger;
      }
      then reject;           # reject the prefixes in the route list
    }
    term term2 {           # reject direct routes
      from protocol direct;
      then reject;
    }
    term term3 {          # reject static routes
      from protocol static;
      then reject;
    }
    term term4 {          # reject local routes
      from protocol local;
      then reject;
    }
    term term5 {          # reject aggregate routes
      from protocol aggregate;
      then reject;
    }
    term term6 {          # accept all other routes
      then accept;
    }
  }
}
```

Apply Routing Policies to the Forwarding Table

To apply an export routing policy to the forwarding table, include the export statement:

```
export [ policy-names ];
```

You can include this statement at the following hierarchy levels:

```
[edit routing-options forwarding-table]
```

```
[edit logical-routers logical-router-name routing-options forwarding-table]
```

In the export statement, list the name of the routing policy to be evaluated when routes are being exported from the routing table into the forwarding table. Only active routes are exported from the routing table.

You can reference the same routing policy one or more times in the same or a different export statement.

For information about how the policy framework software evaluates a routing policy, see “How a Routing Policy Is Evaluated” on page 30.

You can apply export policies to routes being exported from the routing table into the forwarding table for the following features:

- Per-packet load balancing

- Class of service (CoS)

For more information about per-packet load balancing, see “Configure Load-Balance Per-Packet Action” on page 134. For more information about CoS, see the *JUNOS Internet Software Network Interfaces and Class of Service Configuration Guide*.

Examples: Apply Routing Policies

Configure the router to export to IS-IS the routes that match the dmz and local-customers routing policies:

```
[edit]
protocols {
  isis {
    export [dmz local-customers];
  }
}
```

For three BGP peer groups, apply various export and import filters:

```
[edit]
protocols {
  bgp {
    group 1 {
      type external;
      peer-as 47;
      export local-customers;
      import [martian-filter long-prefix-filter as47-filter];
      neighbor 192.168.1.4;
      neighbor 192.168.1.5;
    }
    group 2 {
      type external;
      peer-as 42;
      export local-customers;
      import [martian-filter long-prefix-filter as42-filter];
      neighbor 192.168.1.4;
      neighbor 192.168.1.5;
    }
    group 3 {
      type internal;
      export local-customers;
      neighbor 10.1.1.1;
    }
  }
}
```

Apply the long-prefix-filter prefix only to routes learned from a particular peer within a group:

```
[edit]
protocols {
  bgp {
    group 4 {
      type external;
      peer-as 47;
      export local-customers;
      import [martian-filter as47-filter];
      neighbor 192.168.1.4;
      neighbor 192.168.1.5;
      neighbor 192.168.1.6 {
        import [martian-filter as47-filter long-prefix-filter];
      }
    }
  }
}
```

Examples: Routing Policy Configuration

Redistribute BGP routes with a community tag of 444:5 into IS-IS, changing the metric to 14:

```
[edit]
protocols {
  isis {
    export edu-to-isis;
  }
}
policy-options {
  community edu members 444:5;
  policy-statement edu-to-isis {
    from {
      protocol bgp;
      community edu;
    }
    then {
      metric 14;
      accept;
    }
  }
}
```

Redistribute OSPF routes from area 1 only into BGP, and do not advertise routes learned by BGP:

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  bgp {
    export ospf-into-bgp;
    group {
      type external;
      peer-as 23;
      allow {
        0.0.0.0/0;
      }
    }
  }
}
policy-options {
  policy-statement ospf-into-bgp {
    term ospf-only {
      from {
        protocol ospf;
        area 1;
      }
      then accept;
    }
  }
}
```

Define a routing policy to export direct routes into IS-IS for all interfaces, even if IS-IS is not configured on an interface:

```
[edit]
protocols {
  isis {
    export direct-routes;
  }
}
policy-options {
  policy-statement direct-routes {
    from protocol direct;
    then accept;
  }
}
```

Define a routing policy to export IS-IS Level 1 internal-only routes into Level 2:

```
[edit]
protocols {
  isis {
    export L1-L2;
  }
}
policy-statement L1-L2 {
  term one {
    from {
      level 1;
      external;
    }
    then reject;
  }
  term two {
    from level 1;
    to level 2;
    then accept;
  }
}
```

Define a routing policy to export IS-IS Level 2 routes into Level 1:

```
[edit]
protocols {
  isis {
    export L2-L1;
  }
}
policy-statement L2-L1 {
  term one {
    from level 2;
    to level 1;
    then accept;
  }
}
```

Configure different forwarding next-hop LSPs for different destination prefixes learned from BGP:

```

routing-options {
  router-id 10.10.20.101;
  autonomous-system 2;
  forwarding-table {
    export forwarding-policy;
  }
}
policy-options {
  policy-statement forwarding-policy {
    term one {
      from {
        protocol bgp;
        route-filter 10.1.0.0/16 orlonger;
      }
      then {
        install-nexthop lsp mc-c-lsp-1;
        accept;
      }
    }
    term two {
      from {
        protocol bgp;
        route-filter 10.2.0.0/16 orlonger;
      }
      then {
        install-nexthop lsp mc-c-lsp-2;
        accept;
      }
    }
    term three {
      from {
        protocol bgp;
        route-filter 10.3.0.0/16 orlonger;
      }
      then {
        install-nexthop lsp mc-c-lsp-3;
        accept;
      }
    }
  }
}
protocols {
  mpls {
    label-switched-path mc-c-lsp-1 {
      from 10.10.20.101;
      to 10.10.20.103;
    }
    label-switched-path mc-c-lsp-2 {
      from 10.10.20.101;
      to 10.10.20.103;
    }
  }
}

```

```

        label-switched-path mc-c-lsp-3 {
            from 10.10.20.101;
            to 10.10.20.103;
        }
    }
}

```

Configure a routing policy to group destination prefixes:

```

[edit]
policy-options {
    policy-statement set-dest-class {
        term 1 {
            from community nets1;
            then {
                destination-class on-net;
                accept;
            }
        }
        term 2 {
            from community nets2;
            then {
                destination-class off-net;
                accept;
            }
        }
    }
}
community nets1 [7:8 9:10];
community nets2 [1:2 4:5];

```

Apply a routing policy to the forwarding table with the corresponding destination class:

```

[edit]
routing-options {
    forwarding-table {
        export set-dest-class;
    }
}

```

Enable packet counting on an interface:

```

[edit interfaces]
interfaces so-1/0/1 {
    unit 0 {
        family inet6 {
            accounting {
                destination-class-usage;
            }
        }
    }
}

```

Configure a routing policy to group source prefixes, and allow prefixes that match the policy statement to have a source class created for them:

```
[edit]
policy-options {
  policy-statement set-gold-class {
    term {
      from
        route-filter 10.210.0.0/16 orlonger;
        route-filter 10.215.0.0/16 orlonger;
      then {
        source-class gold-class;
      }
    }
  }
}
```

Apply a routing policy to the forwarding table with the corresponding source class:

```
[edit]
routing-options {
  forwarding-table {
    export set-gold-class;
  }
}
```

Enable packet counting on an interface. In this example, one interface accommodates both input and output:

```
[edit interfaces]
interfaces ge/0/0/0 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
          input;
          output;
        }
      }
    }
  }
}
```

Configure a routing policy to group source and destination prefixes in a forwarding class:

```
[edit]
policy-options {
  policy-statement set-bronze-class {
    term {
      from
        route-filter 10.210.0.0/16 orlonger;
        route-filter 10.215.0.0/16 orlonger;
      then {
        forwarding-class bronze-class;
      }
    }
  }
}
```

Apply a routing policy to the forwarding table with the corresponding forwarding class:

```
[edit]
routing-options {
  forwarding-table {
    export set-bronze-class;
  }
}
```

Enable counting of incoming source packets on an interface:

```
[edit interfaces]
interfaces fe/1/0/0 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
          input;
        }
      }
    }
  }
}
interfaces fe/1/0/1 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
          output;
        }
      }
    }
  }
}
interfaces fe/1/0/2 {
  unit 0 {
    family inet {
      accounting {
        destination-class-usage;
      }
    }
  }
}
```

Configure a policy that accepts routes with the destination prefixes 201:db8::8000/32 and 201:db8::8001/32:

```
[edit policy-options]
policy-statement export_exact {
  term a {
    from {
      route-filter 201:db8::8000/32 exact;
      route-filter 201:db8::8001/32 exact;
    }
    then {
      accept;
    }
  }
  term b {
    then {
      reject;
    }
  }
}
```

Configure a policy that accepts routes with the destination prefix 201:db8::8000/32:

```
[edit policy-options]
policy-statement export_exact {
  term a {
    from {
      protocol bgp;
      route-filter 201:db8::8000/32 exact;
    }
    then {
      accept;
    }
  }
  term b {
    then {
      reject;
    }
  }
}
```

Configure a policy that accepts routes with the destination prefix 201:db8::8000/32:

```
[edit policy-options]
policy-statement export_exact {
  term a {
    from {
      protocol bgp;
      route-filter 201:db8::8000/32 exact;
    }
    then {
      accept;
    }
  }
}
```

```

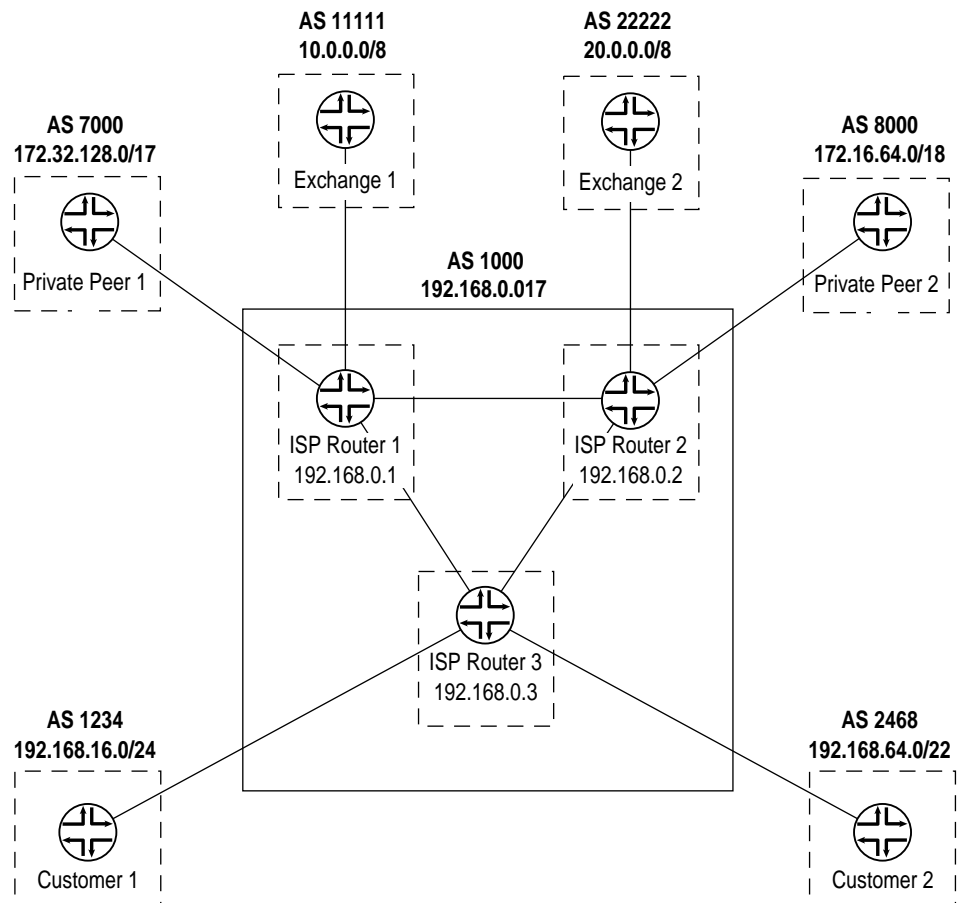
term b {
  then {
    reject;
  }
}

```

Example: ISP Network Case Study

This section provides an example of how policies might be used in a typical Internet service provider (ISP) network. In this network example (see Figure 11), the ISP's AS number is 1000. The ISP has two transit peers (AS 11111 and AS 22222) to which it connects at an exchange point. The ISP is also connected to two private peers (AS 7000 and AS 8000) with which it exchanges specific customer routes. The ISP has two customers (AS 1234 and AS 2468) to which it connects to using the BGP protocol.

Figure 11: ISP Network Example



1741

In this example, the ISP policies are configured in an outbound direction; that is, the example focuses on the routes that the ISP announces to its peers and customers, and includes the following:

1. The ISP has been assigned AS 1000 and the routing space of 192.168.0/17. With the exception of the two customer networks shown in Figure 11, all other customer routes are simulated with static routes.
2. The ISP has connectivity to two different exchange peers: AS 1111 and AS 2222. These peers are used for transit service to other portions of the Internet. This means that the ISP is accepting all routes (the full Internet routing table) from those BGP peers. To help maintain an optimized Internet routing table, the ISP is configured to advertise only two aggregate routes to the transit peers.
3. The ISP also has direct connectivity to two private peers: AS 7000 and AS 8000. The ISP administrators want all data to the private peers to use this direct link. As a result, all the customer routes from the ISP are advertised to those private peers. These peers then advertise all their customer routes to the ISP.
4. Finally, the ISP has two customers with which it communicates using BGP: AS 1234 and AS 2468. Each customer has a different set of requirements.

The following section describes the following topics:

Request a Single Default Route on the Customer 1 Router on page 76

Request Specific Routes on the Customer 2 Router on page 77

Configure Peer Policy on ISP Router 3 on page 79

Configure Private and Exchange Peers on ISP Router 1 and 2 on page 81

Configure Locally Defined Static Routes on the Exchange Peer 2 Router on page 84

Configure Outbound and Generated Routes on the Private Peer 2 Router on page 85

Request a Single Default Route on the Customer 1 Router

Customer 1 has only a single route to the ISP and is using the ISP for transit service. This customer has requested a single default route (0.0.0.0/0) from the ISP.

```
[edit]
interfaces {
  so-0/0/1 {
    description "Connection to ISP Router 3";
    unit 0 {
      family inet {
        address 10.222.70.1/30;
      }
    }
  }
  fxp0 {
    description "MGMT INTERFACE - DO NOT DELETE";
    unit 0 {
      family inet {
        address 10.251.0.9/24;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 192.168.16.1/32;
      }
    }
  }
}
routing-options {
  static {
    route 192.168.16.0/27 reject;
    route 192.168.16.32/27 reject;
    route 192.168.16.64/27 reject;
    route 192.168.16.96/27 reject;
    route 192.168.16.128/27 reject;
    route 192.168.16.160/27 reject;
    route 192.168.16.192/27 reject;
  }
  autonomous-system 1234;
}
protocols {
  bgp {
    group AS1000-Peers {
      type external;
      export send-statics;
      peer-as 1000;
      neighbor 10.222.70.2;
    }
  }
}
```

```

policy-options {
  policy-statement send-statics {
    term static-routes {
      from protocol static;
      then accept;
    }
  }
}

```

Request Specific Routes on the Customer 2 Router

Customer 2 has a link to the ISP, as well as a link to AS 8000. This customer has requested specific customer routes from the ISP, as well as from AS 8000. Customer 2 wants to use the ISP for transit service to the Internet, and has requested a default route from the ISP.

```

[edit]
interfaces {
  so-0/0/1 {
    description "Connection to ISP Router 3";
    unit 0 {
      family inet {
        address 10.222.61.2/30;
      }
    }
  }
  so-0/0/2 {
    description "Connection to Private-Peer 2";
    unit 0 {
      family inet {
        address 10.222.6.1/30;
      }
    }
  }
  fxp0 {
    description "MGMT INTERFACE - DO NOT DELETE";
    unit 0 {
      family inet {
        address 10.251.0.8/24;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 192.168.64.1/32;
      }
    }
  }
}

```

```

routing-options {
  static {
    route 192.168.64.0/25 reject;
    route 192.168.64.128/25 reject;
    route 192.168.65.0/25 reject;
    route 192.168.66.0/25 reject;
    route 192.168.67.0/25 reject;
    route 192.168.65.128/25 reject;
    route 192.168.66.128/25 reject;
    route 192.168.67.128/25 reject;
  }
  autonomous-system 2468;
}
protocols {
  bgp {
    group External-Peers {
      type external;
      import inbound-routes;
      export outbound-routes;
      neighbor 10.222.61.1 {
        peer-as 1000;
      }
      neighbor 10.222.6.2 {
        peer-as 8000;
      }
    }
  }
}
policy-options {
  policy-statement outbound-routes {
    term statics {
      from protocol static;
      then accept;
    }
    term internal-bgp-routes {
      from {
        protocol bgp;
        as-path my-own-routes;
      }
      then accept;
    }
    term no-transit {
      then reject;
    }
  }
  policy-statement inbound-routes {
    term AS1000-primary {
      from {
        protocol bgp;
        as-path AS1000-routes;
      }
      then {
        local-preference 200;
        accept;
      }
    }
  }
}

```

```

    term AS8000-backup {
      from {
        protocol bgp;
        as-path AS8000-routes;
      }
      then {
        local-preference 50;
        accept;
      }
    }
  }
  as-path my-own-routes "";
  as-path AS1000-routes "1000 .*";
  as-path AS8000-routes "8000 .*";
}

```

Configure Peer Policy on ISP Router 3

On ISP Router 3, a separate policy is in place for each customer. The default route for Customer 1 is being sent by the customer-1-peer policy. This policy finds the 0.0.0.0/0 default route in inet.0 and accepts it. The policy also rejects all other routes, thereby not sending all BGP routes on the ISP Router. The customer-2-peer policy is for Customer 2 and contains the same policy terms, which also send the default route and no other transit BGP routes. The additional terms in the customer-2-peer policy send the ISP customer routes to Customer 2. Because there are local static routes on ISP Router 3 that represent local customers, these routes are sent as well as all other internal (192.168.0/17) routes announced to the local router by the other ISP routers.

```

[edit]
routing-options {
  static {
    # simulate local customer routes
    route 192.168.72.0/22 reject;
    route 192.168.76.0/22 reject;
    route 192.168.80.0/22 reject;
    route 192.168.84.0/22 reject;
    route 192.168.88.0/22 reject;
    route 192.168.92.0/22 reject;
    route 192.168.72.0/21 reject;
    route 192.168.80.0/21 reject;
    route 192.168.88.0/21 reject;
  }
  generate {
    # install a default route if certain routes
    # from the Exchange Peers are advertised using BGP
    route 0.0.0.0/0 policy if-upstream-routes-exist;
  }
  autonomous-system 1000;
}

```

```

protocols {
  bgp {
    group Internal-Peers {
      type internal;
      local-address 192.168.0.3;
      export internal-peers;
      neighbor 192.168.0.1;
      neighbor 192.168.0.2;
    }
    group Customer-2 {
      type external;
      export customer-2-peer;
      peer-as 2468;
      neighbor 10.222.61.2;
    }
    group Customer-1 {
      type external;
      export customer-1-peer;
      peer-as 1234;
      neighbor 10.222.70.1;
    }
  }
  isis {
    level 1 disable;
    interface so-0/0/0.0;
    interface ge-0/1/0.0;
    interface lo0.0;
  }
}
policy-options {
  policy-statement internal-peers { # advertise local customer routes to all
  peers
    term statics {
      from protocol static;
      then accept;
    }
    term next-hop-self { # set the BGP next hop to Self for EBGp
      then { # routes advertised to IBGP peers
        next-hop self;
      }
    }
  }
  policy-statement if-upstream-routes-exist {
    term only-certain-contributing-routes {
      from { # allow either the 10.100/17 or the 10.101.0.0/27
        route-filter 10.100.0.0/17 exact; #route to activate the generated route
        route-filter 10.101.0.0/17 exact;
      }
      then accept;
    }
  } # do not allow any other route to activate
  the generated route in the routing table
}

```

```

    term reject-all-other-routes {
        then reject;
    }
}
policy-statement customer-2-peer { # advertise customer routes to all peers
    term statics {
        from protocol static;
        then accept;
    }
    term isp-and-customer-routes { # advertise internal AS 1000 customer
        from {
            to the customer
            protocol bgp;
            route-filter 192.168.0.0/17 orlonger;
        }
        then accept;
    }
    term default-route { # advertise just the default route to AS 2468
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term reject-all-other-routes { # do not advertise any other routes,
        then reject;
    }
}
policy-statement customer-1-peer {
    term default-route { # advertise just the default route to AS 1234
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term reject-all-other-routes { # do not advertise any other routes,
        then reject;
    }
}
}
}

```

Configure Private and Exchange Peers on ISP Router 1 and 2

ISP Router 1 and ISP Router 2 each have two policies configured: the private-peers policy and the exchange-peers policy. Because of their similar configurations, this example describes the configuration for only ISP Router 2.

On ISP Router 2, the private-peers policy sends the ISP customer routes to the Private Peer 2 router. The policy accepts all local static routes (local ISP Router 2 customers) and all BGP routes in the 192.168.0/17 range (advertised by other ISP routers). These two terms represent the ISP customer routes. The final term rejects all other routes, which includes the entire Internet routing table sent by the exchange peers. These routes do not need to be sent to Private Peer 2 for two reasons:

The peer already maintains a connection to Exchange Peer 2 in our example, so the routes are redundant.

The Private Peer wants customer routes only. The `private-peers` policy accomplishes this goal. The `exchange-peers` policy sends routes to the Exchange Peer 2 router.

In the example, only two routes need to be sent to Exchange Peer 2:

The aggregate route that represents the AS 1000 routing space of 192.168.0/17. This route is configured as an aggregate route locally and is advertised by the `exchange-peers` policy.

The address space assigned to Customer 2, 192.168.64/22. This smaller aggregate route needs to be sent to Exchange Peer 2 because the customer is also attached to the AS 8000 peer (Private Peer 2).

Sending these two routes to Exchange Peer 2 allows other networks in the Internet to reach the customer through either the ISP or the Private Peer. If just the Private Peer were to advertise the /22 network while the ISP maintained only its /17 aggregate, then all traffic destined for the customer would transit AS 8000 only. Because the customer also wants routes from the ISP, the 192.168.64/22 route is announced by ISP Router 2. Like the larger aggregate route, the 192.168.64/22 route is configured locally and is advertised by the `exchange-peers` policy. The final term in that policy rejects all routes, including the specific customer networks of the ISP, the customer routes from Private Peer 1, the customer routes from Private Peer 2, and the Internet routing table from Exchange Peer 1. In essence, this final term prevents the ISP from performing transit services for the Internet at large.

```
[edit]
routing-options {
  static {
    route 192.168.32.0/22 reject;
    route 192.168.36.0/22 reject;
    route 192.168.40.0/22 reject;
    route 192.168.44.0/22 reject;
    route 192.168.48.0/22 reject;
    route 192.168.52.0/22 reject;
    route 192.168.32.0/21 reject;
    route 192.168.40.0/21 reject;
    route 192.168.48.0/21 reject;
  }
  aggregate {
    route 192.168.0.0/17;
    route 192.168.64.0/22;
  }
  autonomous-system 1000;
}
```

```

protocols {
  bgp {
    group Internal-Peers {
      type internal;
      local-address 192.168.0.2;
      export internal-peers;
      neighbor 192.168.0.1;
      neighbor 192.168.0.3;
    }
    group AS8000-Peers {
      type external;
      export private-peers;
      peer-as 8000;
      neighbor 10.222.45.2;
    }
    group AS22222-Peers {
      type external;
      export exchange-peers;
      peer-as 22222;
      neighbor 10.222.46.1;
    }
  }
  isis {
    level 1 disable;
    interface so-0/0/0.0;
    interface ge-0/2/0.0;
    interface lo0.0;
  }
}
policy-options {
  policy-statement internal-peers {
    term statics {
      from protocol static;
      then accept;
    }
    term next-hop-self {
      then {
        next-hop self;
      }
    }
  }
  policy-statement private-peers {
    term statics {
      from protocol static;
      then accept;
    }
    term isp-and-customer-routes {
      from {
        protocol bgp;
        route-filter 192.168.0.0/17 orlonger;
      }
      then accept;
    }
    term reject-all {
      then reject;
    }
  }
}

```

```

policy-statement exchange-peers {
  term AS1000-Aggregate {
    from {
      protocol aggregate;
      route-filter 192.168.0.0/17 exact;
    }
    then accept;
  }
  term Customer-2-Aggregate {
    from {
      protocol aggregate;
      route-filter 192.168.64.0/22 exact;
    }
    then accept;
  }
  term reject-all-other-routes {
    then reject;
  }
}

```

Configure Locally Defined Static Routes on the Exchange Peer 2 Router

The Exchange Peer 2 router exchanges all routes with all BGP peers. The outbound-routes policy for Exchange Peer 2 advertises locally defined static routes using BGP.

```

[edit]
protocols {
  bgp {
    group Peers {
      type external;
      export outbound-routes;
      neighbor 10.222.4.1 {
        peer-as 11111;
      }
      neighbor 10.222.44.2 {
        peer-as 8000;
      }
      neighbor 10.222.46.2 {
        peer-as 1000;
      }
    }
  }
}
policy-options {
  policy-statement outbound-routes { # advertise the simulated Internet routes
    term statics { # to all BGP peers
      from protocol static;
      then accept;
    }
  }
}

```

Configure Outbound and Generated Routes on the Private Peer 2 Router

The Private Peer 2 router performs two main functions:

Advertise routes local to AS 8000 to both the Exchange Peers and the ISP routers. The outbound-routes policy advertises the local static routes (that is, customers) on the router, and also advertises all routes learned by BGP that originated in either AS 8000 or AS 2468. These routes include other AS 8000 customer routes in addition to the AS 2468 customer. The AS routes are identified by an AS path regular expression match criteria in the policy.

Advertise the 0.0.0.0/0 default route to the AS 2468 customer router. To accomplish this, the Private Peer creates a generated route for 0.0.0.0/0 locally on the router. This generated route is further assigned a policy called if-upstream-routes-exist, which allows only certain routes to contribute to the generated route, making it an active route in the routing table. Once the route is active, it can be sent to the AS 2468 router using BGP and the configured policies. The if-upstream-routes-exist policy accepts only the 20.100.0.0/17 route from Exchange Peer 2, and rejects all other routes. If the 20.100.0.0/17 route is withdrawn by the Exchange Peer, the Private Peer loses the 0.0.0.0/0 default route and withdraws the default route from the AS 2468 customer router.

```
[edit]
routing-options {
    static {
        route 172.16.64.0/20 reject;
        route 172.16.80.0/20 reject;
        route 172.16.96.0/20 reject;
        route 172.16.112.0/20 reject;
        route 172.16.72.0/21 reject;
        route 172.16.88.0/21 reject;
        route 172.16.104.0/21 reject;
        route 172.16.120.0/21 reject;
    }
    generate {
        route 0.0.0.0/0 policy if-upstream-routes-exist;
    }
    autonomous-system 8000;
}
protocols {
    bgp {
        group External-Peers {
            type external;
            export outbound-routes;
            neighbor 10.222.44.1 {
                peer-as 22222;
            }
            neighbor 10.222.45.1 {
                peer-as 1000;
            }
        }
    }
}
```

```

        group Customers {
            type external;
            export internal-routes;
            neighbor 10.222.6.1 {
                peer-as 2468;
            }
        }
    }
}
policy-options {
    policy-statement outbound-routes { # advertise local customer routes
        term statics {
            from protocol static;
            then accept;
        }
        term allowed-bgp-routes {
            from {
                # advertise routes
                as-path [ my-own-routes AS2468-routes ];
            }
            then accept;
        }
        term no-transit {
            then reject; # do not advertise any other routes,
        }
    }
    policy-statement internal-routes { # advertise local customer routes
        term statics {
            from protocol static;
            then accept;
        }
        term default-route { # advertise just the default route
            from {
                route-filter 0.0.0.0/0 exact;
            }
            then accept;
        }
        term reject-all-other-routes { # do not advertise any other routes,
            then reject;
        }
    }
    policy-statement if-upstream-routes-exist {
        term as-22222-routes {
            from {
                # allow the 10.100.0.0/17 route to activate
                route-filter 10.100.0.0/17 exact;the generated route the routing table
            }
            then accept;
        }
        term reject-all-other-routes {
            then reject; # do not allow any other route to activate
                the generated route in the routing table
        }
    }
    as-path my-own-routes "";
    as-path AS2468-routes "2468";
}

```

Configure the Discard Interface

The discard interface allows you to protect a network from denial-of-service (DoS) attacks by identifying the target IP address that is being attacked and configuring a policy to forward all packets to a discard interface. All packets forwarded to the discard interface are dropped.

To configure the discard interface, include the dsc statement:

```
[edit]
interface interface-name {
  dsc {
    unit 0 {
      family inet {
        filter {
          input filter-name;
          output filter-name;
        }
      }
    }
  }
}
```

You can configure the statement at the following hierarchy levels:

```
[edit interface interface-name]
```

```
[edit logical-routers logical-router-name interface interface-name]
```

The dsc interface name denotes the discard interface. The discard interface supports only unit 0. For more information about configuring interfaces, see the *JUNOS Internet Software Network Interfaces and Class of Service Configuration Guide*.

The following two configurations are required to configure a policy to forward all packets to the discard interface.

Configure an input policy to associate a community with the discard interface:

```
[edit]
policy-options {
  community community-name members [ community-id ];
  policy-statement statement-name {
    term term-name {
      from community community-name;
      then {
        next-hop address;          #Remote end of the point-to-point interface
        accept;
      }
    }
  }
}
```

Configure an output policy to set up the community on the routes injected into the network:

```
[edit]
policy-options {
  policy-statement statement-name {
    term term-name {
      from prefix-list name;
      then community (set | add | delete) community-name;
    }
  }
}
```

Test Routing Policies

Before applying a routing policy in a live environment, you can use the test policy command to ensure that the policy produces the results that you expect. The test policy command uses the following syntax:

```
user@host> test policy policy-name prefix
```

For more information about test commands, see the *JUNOS Internet Software Protocols, Class of Service, Chassis, and Management Command Reference*.

Example: Test a Routing Policy

Test the following policy, which looks for unwanted routes and rejects them:

```
[edit policy-options]
policy-statement reject-unwanted-routes {
  term drop-these-routes {
    from {
      route-filter 0/0 exact;
      route-filter 10/8 orlonger;
      route-filter 172.16/12 orlonger;
      route-filter 192.168/16 orlonger;
      route-filter 224/3 orlonger;
    }
    then reject;
  }
}
```

Test this policy against all routes in the routing table:

```
user@host> test policy reject-unwanted-routes 0/0
```

Test this policy against a specific set of routes:

```
user@host> test policy reject-unwanted-routes 10.49.0.0/16
```