

Chapter 5

Configure Routing Policy

To configure routing policy, you include statements at the [edit policy-options] hierarchy level of the configuration. This chapter describes the following tasks and provides the following examples:

Define Routing Policies on page 35

Apply Routing Policies on page 45

Examples: Routing Policy Configuration on page 52

Define Routing Policies

To define routing policies, include one or more policy-statement statements at the [edit policy-options] hierarchy level:

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

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

Policy Name on page 36

Policy Terms on page 36

Policy Match Conditions on page 37

Policy Actions on page 39

Examples: Define Routing Policies on page 43

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 policy name must be unique within a configuration.

Policy Terms

A 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 policy.

When a policy contains only a single term, the term statement is optional. However, we recommend that you always include it in case you later add terms to the policy.

Policy Match Conditions

Each term consists of two statements, `from` and `to`, that define match conditions:

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

In the `from` statement, you define the conditions that an incoming route must match. You can match the following:

- Information about the route's source. You can specify one or more of match conditions. If you specify more than one, they all must match the route for a match to occur.

- Routes coming from a source. For more information, see "Examples: Routing Policy Configuration" on page 52.

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

The `to` statement defines match conditions that pertain to the route's destination or the protocol into which the route is being advertised. You can specify one or more of match conditions. If you specify more than one, they all must match the route for a match to occur.

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



Note

When you are matching information about a route's source or destination, 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 route lists is handled differently.

Table 2 describes the conditions available for matching a route's source or destination. Two of the conditions—`as-path` and `community`—have complex arguments. For these conditions, you define the arguments separately, giving them a name, and then you specify that name in the match condition. For information about defining these arguments, see "Examples: Create AS Path Regular Expressions in Routing Policies" on page 68 and "Define Communities to Use in Routing Policies" on page 69.

Table 2: Policy Conditions That Match a Route's Source or Destination

Match Condition	Description
<i>area area-id</i>	(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.
<i>as-path name</i>	(BGP only) Name of an AS path regular expression. When creating policies that apply to AS path regular expressions, you also must create named regular expressions as described in "Examples: Create AS Path Regular Expressions in Routing Policies" on page 68.
<i>color preference</i> <i>color2 preference</i>	Preference value. You can specify a primary preference value (<i>preference</i>); a secondary preference value (<i>preference2</i>); and even finer-grained preference values (<i>color</i> and <i>color2</i>). The preference value can be a number in the range 0 through 4294967295 ($2^{32} - 1$), with a lower number indicating a more-preferred route. For more information about preference values, see Table 1 on page 8.
<i>community [names]</i>	(BGP only) Name of one or more BGP communities. If you list more than one name, the communities are logically ORed. For examples, see "" on page 76. When creating policies that apply to BGP communities, you also must create named communities and define their members as described in "Define Communities to Use in Routing Policies" on page 69.
<i>external</i>	External routes, including routes exported from one level to another.
<i>interface interface-name</i>	Router interface or interfaces specified by name or IP address. Do not use this qualifier with protocols that are not interface specific, such as internal BGP. In a from statement, match a route learned from one of the specified interfaces. In a to statement, match a route to be advertised over one of the specified interfaces.
<i>level level</i>	(IS-IS only) IS-IS level.
<i>local-preference value</i>	(BGP only) BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range 0 through 4294967295 ($2^{32} - 1$).
<i>metric metric</i> <i>metric2 metric</i> <i>metric3 metric</i> <i>metric4 metric</i>	Metric value. You can specify up to four metric values, starting with <i>metric</i> (for the first metric value) and continuing with <i>metric2</i> , <i>metric3</i> , and <i>metric4</i> . (BGP only) <i>metric</i> corresponds to the MED, and <i>metric2</i> corresponds to the IGP metric if the BGP next hop recurses through another route.
<i>neighbor address</i>	Neighbor (peer) address or addresses. A peer can be remote, because it can be an interior BGP peer, an OSPF advertising router, or an IS-IS router that advertised the prefix. In a from statement, match a route learned from one of the specified peers. In a to statement, match a route being advertised to one of the specified peers.
<i>next-hop address</i>	In a from statement only, match a route whose next hop matches one of the specified addresses.
<i>origin value</i>	(BGP only) BGP origin attribute, which is the origin of the AS path information. The value can be one of the following: <i>igp</i> —Path information originated within the local AS. <i>egp</i> —Path information originated in another AS. <i>incomplete</i> —Path information was learned by some other means.
<i>policy [policy-names]</i>	Name of a policy to evaluate. For more information, see "Evaluate One Routing Policy within Another" on page 50.

Match Condition	Description
preference <i>preference</i> preference2 <i>preference</i>	Preference value. You can specify a primary preference value (preference); a secondary preference value (preference2); and even finer-grained preference values (color and color2). The preference value can be a number in the range 0 through 4294967295 ($2^{32} - 1$), with a lower number indicating a more-preferred route. For more information about preference values, see Table 1 on page 8.
protocol <i>protocol</i>	Name of the protocol from which the route was learned. It can be one of the following: aggregate, bgp, direct, dvmrp, isis, local, ospf, pim-dense, pim-sparse, rip, or static.
rib inet <i>number</i>	Name of a routing table. The default is the default unicast routing table, inet.0.
tag <i>tag</i> tag2 <i>tag</i>	(OSPF only) 32-bit tag field in OSPF external LSA packets. You can specify two tag strings: tag (for the first string) and tag2.

Policy Actions

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

```
[edit]
policy-options {
  policy-statement policy-name {
    term term-name {
      from {
        (match-conditions | (route-filter destination-prefix match-type <actions>));
      }
      to {
        match-conditions;
      }
      then actions;
    }
  }
}
```

You can specify one or more of the actions listed in Table 3 and Table 4. There are three types of policy actions:

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

Tracing actions, which log route matches.

Actions that set properties associated with the route's information.

All conditions in the from and to statements must match for the action to be taken. If you do not specify from and to statements, all routes are considered to match and the actions apply to them all.



Note

When you apply an action that sets the properties associated with the route's information, the changes occur in a copy of the source route. The source route 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 policy has been applied, use the show route receive-protocol command.

The then statement is optional. If you omit it, one of the following occurs:

If you do not specify a flow control action (such as accepting or rejecting the route), the next term in the policy is evaluated.

If there are no more terms in the policy, the next policy is evaluated.

If there are no more terms or policies, the default action is taken, as described in “Default Routing Policy Actions” on page 29.

If, in the from statement, you specify route lists (routes that are grouped so that a common action can be applied to them), 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 “Examples: Routing Policy Configuration” on page 52.

One of the actions—damping—has complex arguments. For this action, you define the arguments separately, giving them a name, and then you specify the name in the action. For information about defining these arguments, see “Examples: Define Damping Parameters” on page 80.

Table 3: Policy Control and Tracing Actions

Policy Action	Description
Flow Control Actions	
accept	Accept the route and propagate it. After a route is accepted, no other terms in the policy and no other policies are evaluated.
next policy	Skip to and evaluate the next policy. Any actions in the then statement preceding the next policy action that set route information are applied to the route. Any that follow it are ignored. next policy is the default control action if a match occurs, if you do not specify a flow control action, and if there are no further terms in the current policy.
next term	Skip to and evaluate the next term in the same policy. Any actions in the then statement preceding the next term action that set route information are applied to the route. Any that follow it are ignored. next term is the default control action if a match occurs and if you do not specify a flow control action.
reject	Reject the route and do not propagate it. After a route is rejected, no other terms in the policy and no other policies are evaluated.
Tracing Action	
trace	Log the match using the trace file specified in the global traceoptions statement. The default file is rpd. This action does not affect the flow control during policy evaluation.

Table 4: Policy Actions That Set Properties Associated with Route Information

Policy Action	Description
<code>as-path-prepend as-path</code>	(BGP only) Affix one or more AS numbers at the beginning of the AS path. To specify 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 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.
<code>class class-name</code>	(Class of service only) Apply class-of-service parameters to routes installed into the routing table.
<code>color preference</code> <code>color2 preference</code>	Set the preference value to a specific value. You can specify a primary preference value (preference); a secondary preference value (preference2); and even finer-grained preference values (color and color2). The preference value can be a number in the range 0 through 4294967295 ($2^{32} - 1$), with a lower number indicating a more preferred route. If you set the preference with the preference action, the new preference remains associated with the 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 Table 1 on page 8.
<code>color (add + subtract -) number</code> <code>color2 (add + subtract -) number</code>	Change the color preference value by the specified amount. If an addition operation results in a value that is greater than 4294967295 ($2^{32} - 1$), 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 no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.
<code>community (+ add) [names]</code>	(BGP only) Add communities to the set of communities in the route.
<code>community (- delete) [names]</code>	(BGP only) Delete communities from the set of communities in the route.
<code>community (= set) [names]</code>	(BGP only) Set the communities in the route, replacing any communities that were in the route.
<code>damping name</code>	(BGP only) Apply 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 “Enable Route Flap Damping” on page 333, and you must create a named list of parameters as described in “Examples: Routing Policy Configuration” on page 52.
<code>install-nexthop lsp lsp-name</code>	Choose which, among a set of equal-cost label-switched path (LSP) next hops, are installed in the forwarding table. To choose, use the export policy for the forwarding table to specify the LSP next hop to be used for the desired routes.
<code>load-balance per-packet</code>	(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.
<code>local-preference value</code>	(BGP only) Set the BGP local preference (LOCAL_PREF) attribute. The preference value can be a number in the range 0 through 4294967295.
<code>local-preference</code> <code>(add + subtract -) number</code>	Change the local preference value by the specified amount. If an addition operation results in a value that is greater than 4294967295 ($2^{32} - 1$), 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 no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value. For BGP, if the attribute value is not known, the local preference attribute value is initialized to 100 before the policy is applied.

Policy Action	Description
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 recurses 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 4294967295 ($2^{32} - 1$), 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 no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.
next-hop <i>address</i>	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.
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.
preference <i>preference</i> preference2 <i>preference</i>	Set the preference value. You can specify a primary preference value (preference); a secondary preference value (preference2); and even finer-grained preference values (color and color2). The preference value can be a number in the range 0 through 4294967295 ($2^{32} - 1$), with a lower number indicating a more-preferred route. If you set the preference with the preference action, the new preference remains associated with the 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 Table 1 on page 8.
preference (add + subtract -) <i>number</i> preference2 (add + subtract -) <i>number</i>	Change the preference value by the specified amount. If an addition operation results in a value that is greater than 4294967295 ($2^{32} - 1$), 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 no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.
tag <i>tag</i> tag2 <i>tag</i>	(OSPF only) Set the 32-bit tag field in OSPF external LSA packets. You can specify two tag strings: tag (for the first string) and tag2.
tag (add + subtract -) <i>number</i> tag2 (add + subtract -) <i>number</i>	Change the tag value by the specified amount. If an addition operation results in a value that is greater than 4294967295 ($2^{32} - 1$), 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 no attribute value is set before the addition or subtraction operation, the attribute value defaults to 0. If you perform an addition to an attribute with a value of 0, that number becomes the constant value.

Examples: Define Routing Policies

IS-IS to BGP

Accept only IS-IS routes advertised by the peer 128.125.1.1. If a route matches, it is accepted, and no further evaluation is performed on that route. If a route does not match, the default action is taken, as described in “Default Routing Policy Actions” on page 29. If you apply this policy to exported BGP routes, only the routes learned from the peer 128.125.1.1 are advertised to BGP peers.

```
[edit]
policy-options {
  policy-statement isis-to-bgp {
    term term1 {
      from {
        protocol isis;
        neighbor 128.125.1.1;
      }
      then {
        accept;
      }
    }
  }
}
```

Set Preference

Define a policy that matches routes from specific next hops that are being advertised into specific areas 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 policy (if configured) is evaluated, then the default action is taken, as described in “Default Routing Policy Actions” on page 29.

```
[edit]
policy-options {
  policy-statement set-preference {
    term term1 {
      from {
        next-hop [10.0.0.1 10.0.0.2];
      }
      to {
        area 23;
      }
      then {
        preference 10;
      }
    }
  }
}
```

```

    term term2 {
      from {
        next-hop 10.0.0.3;
      }
      to {
        area 24;
      }
      then {
        preference 15;
      }
    }
  }
}

```

Configure Forwarding LSP Next Hop

Configure the forwarding next hop LSP to learn about more specific prefixes of specified routes that are using BGP:

```

[edit routing-options]
router-id 10.10.20.101;
autonomous-system 2;
forwarding-table {
  export forwarding-policy;
}

[edit 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;
    }
  }
  term two {
    from {
      protocol bgp;
      route-filter 10.2.0.0/16 orlonger;
    }
    then {
      install-nexthop lsp mc-c-lsp-2;
    }
  }
  term three {
    from {
      protocol bgp;
      route-filter 10.3.0.0/16 orlonger;
    }
    then {
      install-nexthop lsp mc-c-lsp-3;
    }
  }
}

```

```
[edit 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;
}
```

For additional examples, see “Examples: Routing Policy Configuration” on page 52.

Apply Routing Policies

For a routing policy to take effect, you must apply it in the following ways:

Apply Routing Policies to a Routing Protocol on page 45

Apply Routing Policies to the Forwarding Table on page 49

Evaluate One Routing Policy within Another on page 50

Apply Routing Policies to a Routing Protocol

To apply policies to a routing protocol, include the import and export statements when configuring the routing protocol:

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

In the import statement, list the names of one or more policies to be evaluated when routes are imported into the routing table from the routing protocol. You can define one or more import policies. If no match for a route is found, the default action is for the routing table to import the route.

In the export statement, list the name of one or more 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 define one or more export policies. If no match for a route is found, all protocols export the routes that were learned from that same protocol. In addition, IGP export the set of direct routes matching the interfaces on which they are explicitly configured.

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

Specify Multiple Policy Names when Applying Routing Policies

In the import and export statements, you can specify one or multiple policy names. You can list the policy names sequentially to have them evaluated linearly, and you can use logical operators to create more complex statements.

When you list the policy names sequentially, the policies are evaluated in the order you specify them, from left to right, as follows:

1. The terms in each policy are evaluated in order, from first to last.
2. The first matching policy is applied to the route.
3. If the policy action contains an accept or reject control action, no more policy terms or policies are evaluated.
4. If there is no accept or reject action and if there are more terms in the policy, the next term is evaluated.
5. If there are no further terms in the current policy, the next policy listed in the import or export statement is evaluated.

When you list policies sequentially, enter them in the following format:

```
[policy1 policy2 ...]
```

You can create more complex import and export statements by using logical operators to create boolean expressions that define how to evaluate multiple policies. Table 5 describes the logical operators you can use.

Table 5: Logical Operators for Specifying Multiple Policy Names

Logical Operator (Highest to Lowest Precedence)	Description
!	Logical NOT. If the first policy does not match, the next policy is evaluated.
&&	Logical AND. If the first policy matches, the next policy is evaluated. If the first policy does not match, the next policy is skipped.
	Logical OR. If the first policy matches, the next policy is skipped. If the first policy does not match, the next policy is evaluated.
()	Group operators to override default precedence order.

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

Logical OR—In the following example, policy1 is evaluated first. If a destination prefix matches this policy, policy2 is skipped. If policy1 does not match, policy2 is evaluated.

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

Logical AND—In the following example, policy1 is evaluated first. If a destination prefix matches this policy, policy2 is evaluated. If policy1 does not match, policy2 is skipped.

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

Logical NOT—In the following example, policy1 is evaluated first. If a destination prefix matches this policy, policy2 is not evaluated. If policy1 does not match, policy2 is evaluated.

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

The sequential list [policy1 policy2 policy3] is not the same as the expression [policy1 && policy2 && policy3] or [policy1 || policy2 || policy3]:

[policy1 policy2 policy3]—If policy1 matches and the action is accept or reject, no more policies are evaluated.

[policy1 && policy2 && policy3]—If policy1 matches and the action is accept, policy2 is evaluated.

[policy1 || policy2 || policy3]—If policy1 matches and the action is accept, no more policies are evaluated.

[policy1 policy2 policy3]—If policy2 matches and the action is reject, policy3 is not evaluated.

[policy1 || policy2 || policy3]—If policy2 matches and the action is reject, policy3 is evaluated.

You can also combine policy expressions and sequential lists. In the following example, if policy1 is accepted, policy2 is evaluated. If policy2 is accepted and contains a next policy action, policy3 is evaluated. If policy2 is accepted but does not contain a next policy action, policy3 is never evaluated.

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

Apply Routing Policies to Routing Protocols

You can apply routing policies to the following routing protocols:

BGP—Global import and export statements, group import and export statements, and peer import statements (peers in a group share a common export policy). A peer import statement overrides a group import statement. A group import or export statement overrides a global BGP import or export statement. For more information, see “Configure BGP Routing Policy” on page 336.

DVMRP—DVMRP global import and export statements. For more information, see “Configure DVMRP Routing Policy” on page 394.

IS-IS—IS-IS global export statement. For more information, see “Configure IS-IS Routing Policy” on page 188.

OSPF—OSPF global export statement. For more information, see “Configure OSPF Routing Policy” on page 233.

RIP—RIP default and neighbor import statements and group export statement. For more information, see “Apply Import Policy” on page 263 and “Apply Export Policy” on page 264.



Note

You cannot apply import policies to link-state protocols because doing so could lead to an inconsistent topology database.

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

In export policies, omitting the from statement in a policy 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 policy term with an empty from statement, these routes inadvertently could be exported because they matched the from statement. For example, the following 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 1.2.2.4;
    }
  }
}
policy-options {
  policy-statement statics-policy {
    term term1 {
      from {
        route-filter 192.168.0.0/16 orlonger;
        route-filter 224.0.0.0/3 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 policy results in BGP advertising static and direct routes to its peers because:

term1 rejects the destination prefixes enumerated in the route list.

term2, which has no from statement, matches all other routes, including static and direct routes (because you omit the from statement, so all routes are considered to match), and accepts all these routes (by virtue of including the accept statement).

To modify the policy shown above so that an IGP does not export unwanted routes, you need to specify additional policy terms. For example:

```
[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 224.0.0.0/3 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

You can apply export policies to routes being exported from the routing table into the forwarding table. For more information, see “Configure Per-Packet Load Balancing” on page 125.

Evaluate One Routing Policy within Another

It is possible to evaluate one routing policy within another. One use for this is to evaluate a set of route filters within another policy.

To evaluate one policy within another, create the subroutine policy and specify its name using the policy match condition in another policy's from or to statement:

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



Note

Do not evaluate a policy within itself. If you attempt to do so, no prefixes will ever match the policy.

If the subroutine policy matches, the actions specified in the then statement of the subroutine policy are executed immediately, resulting in unexpected side effects in the execution of the policy. Plan your policies carefully.

Examples: Apply Routing Policies

Configure IS-IS to export routes that match the dmz and localcustomers policies:

```
[edit]
protocols {
  isis {
    export [dmz localcustomers];
  }
}
```

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

```
[edit]
protocols {
  bgp {
    group 1 {
      type external;
      peer-as 47;
      neighbor 1.2.2.4;
      neighbor 1.2.2.5;
      export localcustomers;
      import [martianfilter longprefixfilter as47filter];
    }
    group 2 {
      type external;
      peer-as 42;
      neighbor 2.1.2.4;
      neighbor 2.1.2.5;
      export localcustomers;
      import [martianfilter longprefixfilter as42filter];
    }
    group 3 {
      type internal;
      neighbor 10.1.1.1;
      export localcustomers;
    }
  }
}
```

Apply the longprefixfilter prefix only to routes learned from a particular peer within a group:

```
[edit]
protocols {
  bgp {
    group 4 {
      type external;
      peer-as 47;
      export localcustomers;
      neighbor 1.2.2.4;
      neighbor 1.2.2.5;
      neighbor 1.2.2.6 {
        import [martianfilter as47filter longprefixfilter];
      }
      import [martianfilter as47filter];
    }
  }
}
```

Create the subroutine policy is-customer and call it from within the policies export-customer and import-customer. In import-customer, the action is taken only on BGP packets that match the route filters defined in is-customer.

```
[edit policy-options]
policy-statement is-customer {
  from {
    route-filter 172.100.1.0/24 exact;
    route-filter 171.186.100.0/24 exact;
  }
  then accept;
}
policy-statement export-customer {
  from policy is-customer;
  then accept;
}
policy-statement import-customer {
  from {
    protocol bgp;
    policy is-customer;
  }
  then {
    preference 10;
    accept;
  }
}
```

Examples: Routing Policy Configuration

Redistribute BGP routes from a community into IS-IS, changing the metric to 14:

```
[edit]
protocols {
  isis {
    export edu-to-isis;
  }
}
policy-options {
  community edu members 666:5;
  policy-statement edu-to-isis {
    from {
      protocol bgp;
      community edu;
    }
    then {
      metric 14;
      accept;
    }
  }
}
```

Advertise all static routes with a metric of 10 and do not advertise any directly connected routes:

```
[edit]
policy-options {
  policy-statement rip-export {
    term first {
      from protocol static;
      then {
        metric 10;
        then accept;
      }
    }
    term second {
      from protocol direct;
      then reject;
    }
  }
}
```

Redistribute OSPF routes into 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;
      then accept;
    }
    then reject;
  }
}
```

Configure BGP so that all but one peer receives routes learned by OSPF:

```
[edit]
routing-options {
  autonomous-system 56;
}
protocols {
  bgp {
    group group1 {
      export ospf-into-bgp;
      type external;
      peer-as 47;
      neighbor 1.2.2.4;
      neighbor 1.2.2.5;
      neighbor 1.2.2.6;
    }
  }
}
policy-options {
  policy-statement ospf-into-bgp {
    term first {
      from protocol ospf;
      to neighbor 1.2.2.6;
      then reject;
    }
    term second {
      from protocol ospf;
      then accept;
    }
  }
}
```

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

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

Delete a particular community from a route, leaving remaining communities untouched:

```
[edit]
protocols {
  bgp {
    import delete-dunedin;
  }
}
policy-options {
  community dunedin members 701:666;
  policy-statement delete-dunedin {
    then {
      community delete dunedin;
    }
  }
}
```

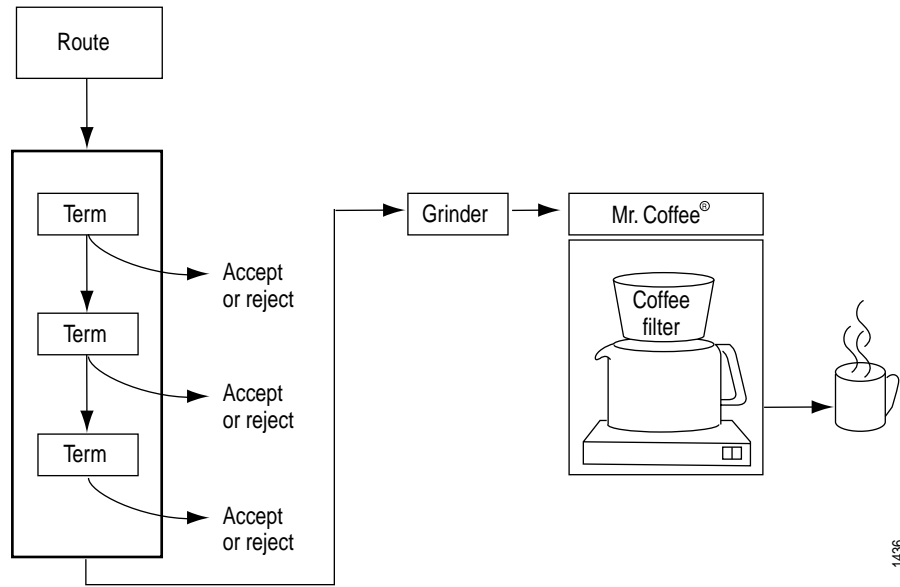
Define a policy to export IS-IS Level 1 internal-only routes into Level 2:

```
[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 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;
  }
}
```

Figure 6: Routing Policy as a Filter



1436