

Chapter 6

Configure Route Lists

You can use routing policy to filter routes. You do this with route lists, which allow you to evaluate and process routes quickly. Using route lists, you can match groups of routes and then perform a common action on them, or you can perform an action on each individual route immediately.

To specify route prefixes in policies, include one or more route-filter options in the from statement of the policy-statement statement:

```
[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;
    }
  }
}
```

destination-prefix is the IP prefix specified as *prefix/prefix-length*. If you omit *prefix-length*, the default is /32.

match-type is the type of match to apply to the destination prefix. It can be one of the match types listed in Table 6. For examples of using the match types, see “Examples: Specify Route Lists in Policies” on page 59.

prefix-list *name* is the name of a list of IP address prefixes. For information on creating a named list of IP address prefixes, see “Define a List of IP Address Prefixes” on page 59.

actions is the action to take if the destination prefix matches. It can be one or more of the actions listed in Table 3 and Table 4.

In route lists, you can specify actions in two ways:

In the route-filter option—These action are taken immediately, and the then statement in the policy term is not evaluated.

In the then statement—These actions are taken after all the routes in the list are evaluated.

Table 6: Route List Match Types

Match Type	Match If ...
exact	<i>prefix</i> matches the route's prefix exactly, and <i>prefix-length</i> is equal to the route's prefix length.
longer	<i>prefix</i> matches the route's prefix exactly, and <i>prefix-length</i> is greater than the route's prefix length.
orlonger	<i>prefix</i> matches the route's prefix exactly, and <i>prefix-length</i> is equal to or greater than the route's prefix length.
upto <i>prefix-length2</i>	The route shares the same most-significant bits (described by <i>prefix-length</i>) and the route's prefix length falls between <i>prefix-length</i> and <i>prefix-length2</i> .
through <i>destination-prefix</i>	Matches all the following: Route matches first destination prefix Route matches second destination prefix for the number of bits in the prefix length Number of bits in the route's prefix length is less than or equal to the number of bits in the second prefix The through match type covers a corner case and you do not use it in most policy configurations.

To determine whether a route matches a prefix in the route list, all the prefixes in the route list are examined and a longest-match lookup is performed. (The order of different prefixes within the route list is not significant.) If a match occurs, the action associated with that prefix is taken. If the lookup does not yield a match, the policy term does not match the route.



If you specify multiple prefixes in the route list, only one prefix need match for a match to occur. The route list matching is effectively a logical OR operation.

Within a route list, you can configure the same destination prefix multiple times. In this case, the order of the prefixes is important, because the list of identical prefixes is searched linearly, and the first match type that matches the candidate route applies. In the following example, different match types are specified on the same prefix. The route 0.0.0.0/0 would be rejected, while the route 0.0.0.0/8 would be marked with next-hop self.

```
0.0.0.0/0 exact reject;  
0.0.0.0/0 longer next-hop self;
```

Define a List of IP Address Prefixes

You can define and name a set of IP address prefixes and use them in the configuration for routing policy statements and firewall filters. To define an IP address prefix list, include the `prefix-list` statement at the [edit policy options] hierarchy level. You can specify one or more addresses.

```
[edit policy-options]
prefix-list name {
  ip-addresses;
}
```

You can use a defined prefix list to support route filters in a policy for IP addresses that are used for exact matches. To do this, include the name of a defined prefix list with the `from` statement at the [edit policy options policy-statement *policy-name* term *term-name*] hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name]
from {
  prefix-list name;
}
```



Note

Per-prefix policy actions cannot be applied to individual prefixes in the list or to the collection of prefixes in the list.

For information about firewall filters, see the *JUNOS Internet Software Configuration Guide: Interfaces and Chassis*.

Examples: Specify Route Lists in Policies

The examples in this section show only fragments of routing policies. Normally, you would combine these fragments with other terms or policies.

In all these examples, remember that the default actions apply to non-matching routes:

Evaluate next term, if present

Evaluate next policy, if present

Take default action, as described in “Default Routing Policy Actions” on page 29

Reject routes with a destination prefix of 0.0.0.0 and a mask length from 0 through 8, and accept all other routes:

```
[edit]
policy-options {
  policy-statement from-hall2 {
    term 1 {
      from {
        route-filter 0.0.0.0/0 upto 0.0.0.0/8 reject;
      }
    }
    then accept;
  }
}
```

Reject routes with a mask of /8 and greater (that is, /8, /9, /10, and so on) and accept all nonmatching routes:

```
[edit]
policy-options {
  policy-statement from-hall3 {
    term term1 {
      from {
        route-filter 0/0 upto /7 accept;
        route-filter 0/0 orlonger;
      }
    }
    then accept;
  }
}
```

Accept host routes (128.125.0.0/32) from USC (128.125.0.0/16):

```
[edit]
policy-options {
  policy-statement usc-hosts-only {
    from {
      route-filter 128.125.0.0/16 upto /31 reject;
    }
    then accept;
  }
}
```

The through match type covers a corner case and you do not use it in most policy configurations. You should think of through as a tool to group a contiguous set of exact matches. For example, instead of specifying four exact matches:

```
from route-filter 0.0.0.0/1 exact
from route-filter 0.0.0.0/2 exact
from route-filter 0.0.0.0/3 exact
from route-filter 0.0.0.0/4 exact
```

You could represent them with the following single match:

```
from route-filter 0.0.0.0/1 through 0.0.0.0/4
```

Explicitly accept a limited set of prefixes (in the first term) and reject all others (in the second term):

```
[edit policy-options]
policy-statement internet-in {
  term 1 {
    from {
      route-filter 192.168.231.0/24 exact accept;
      route-filter 192.168.244.0/24 exact accept;
      route-filter 192.200.198.0/24 exact accept;
      route-filter 192.200.160.0/24 exact accept;
      route-filter 192.200.59.0/24 exact accept;
    }
  }
  term 2 {
    then {
      reject;
    }
  }
}
```

Reject a few groups of prefixes, then accept the remaining prefixes:

```
[edit policy-options]
policy-statement drop-routes {
  term 1{
    from {
      route-filter default exact reject;           # first, reject a number of prefixes:
      route-filter 0.0.0.0/8 orlonger reject;      # reject 0.0.0.0/0 exact
      route-filter 127.0.0.0/8 orlonger reject;    # reject prefix 0, mask /8 or longer
      route-filter 128.105.0.0/16 exact {         # reject loopback addresses
        as-path-prepend "1 2 3";                 # accept 128.105.0.0/16 exact
      }
      route-filter 192.0.2.0/24 orlonger reject;   # reject test network packets
      route-filter 224.0.0.0/3 orlonger reject;   # reject multicast and higher
      route-filter 0.0.0.0/0 upto /24 next policy; # accept everything up to /24
      route-filter 0.0.0.0/0 orlonger;           # accept everything else
    }
  }
}
```

Ignore or reject routes learned from various routing protocols, then define a route list to evaluate the remaining routes:

```
[edit]
policy-options {
  community internal666 members 666:1;
  policy-statement redistributions {
    term bgp {
      from protocol bgp;           # ignore routes learned from BGP and skip to next policy
      then next policy;
    }
    term ospf {
      from protocol ospf;         # reject routes learned from OSPF
      then reject;
    }
  }
  term isis {
    from protocol isis;         # reject routes learned from IS-IS
    then reject;
  }
}
```

```

term dvmrp {                               # reject routes learned from DVMRP
  from protocol dvmrp;
  then reject;
}
term other-routes {                         # evaluate the remaining routes
  from {                                    # then evaluate no other terms or policies
    route-filter 137.39.0.0/16 longer;
    route-filter 153.36.0.0/14 longer;
    route-filter 153.34.0.0/15 longer;
    route-filter 157.130.0.0/16 longer;
    route-filter 158.43.0.0/16 longer;
    route-filter 198.3.0.0/16 longer;
    route-filter 198.4.0.0/15 longer;
    route-filter 198.6.0.0/16 longer;
    route-filter 199.170.0.0/15 longer;
    route-filter 199.172.0.0/15 longer;
    route-filter 204.176.0.0/14 longer;
    route-filter 204.252.0.0/14 longer;
    route-filter 205.228.0.0/14 longer;
    route-filter 206.64.0.0/14 longer;
    route-filter 206.112.0.0/14 longer;
    route-filter 206.136.0.0/14 longer;
    route-filter 207.16.0.0/14 longer;
    route-filter 207.76.0.0/14 longer;
    route-filter 207.108.0.0/16 longer;
    route-filter 207.115.160.0/19 longer;
    route-filter 207.176.128.0/18 longer;
    route-filter 207.247.0.0/16 longer;
    route-filter 208.192.0.0/11 longer;
    route-filter 192.168.0.0/16 longer;
    route-filter 208.224.0.0/12 longer;
    route-filter 208.252.0.0/14 longer;
  }
  then {                                    # apply actions to all matching routes; then skip to the next policy
    next-hop self;
    community add internal666;
    next policy;
  }
}
term last {                                # apply action to the routes not matching previous terms
  then {
    next-hop self;
  }
}
}

```

Reject all prefixes longer than 24 bits. You would install this policy in a sequence of policies in an export statement. The first term in this filter passes on all 0.0.0.0/0 through 0.0.0.0/24 routes unaltered. The second, unnamed term rejects everything else.

```

[edit policy-options]
policy-statement 24bit-filter {
  term acl20 {
    from {
      route-filter 0.0.0.0/0 upto 0.0.0.0 /24;
    }
    then next policy;
  }
  then reject;
}

```

If, in this example, you were to specify route-filter 0.0.0.0/0 upto /24 accept, matching prefixes would be accepted immediately and the next policy in the export statement would never get evaluated.

If you were to include the then reject statement in the term acl20, prefixes greater than 24 bits would never get rejected because the policy software, when evaluating the term, would move on to evaluating the next statement before reaching the then reject statement.



