

## Chapter 9

# Firewall Filter Configuration

To configure firewall filters, you include statements at the [edit firewall] hierarchy level of the configuration:

```
[edit firewall]
family family-name {
  filter filter-name {
    accounting-profile name;
    interface-specific;
    term term-name {
      from {
        match-conditions;
      }
      then {
        action;
        action-modifiers;
      }
    }
  }
  service-filter filter-name {
    term term-name {
      from {
        match-conditions;
      }
      then {
        action;
        action-modifiers;
      }
    }
  }
}
```

This chapter describes the following tasks for configuring firewall filters:

Minimum Firewall Filter Configuration on page 158

Configuring Firewall Filters on page 159

Configuring Service Filters on page 187

Applying Firewall Filters to Interfaces on page 188

Configuring Accounting on page 191

Configuring Filter-Based Forwarding on page 193

Configuring Forwarding Table Filters on page 195

Configuring Firewall Filter System Logging on page 198



**NOTE:** Stateless firewall filtering is not supported on the interfaces you configure as tunnel sources. This affects only the transit packets exiting the tunnel.

## Minimum Firewall Filter Configuration

To configure a firewall filter, you must perform at least the following tasks:

Configure firewall filters—To configure firewall filters, include the family *family-name* statement and one or more filter statements at the [edit firewall] hierarchy level:

```
[edit firewall]
family family-name {
  filter filter-name {
    term term-name {
      from {
        match-conditions;
      }
      then {
        action;
        action-modifiers;
      }
    }
  }
}
```

Apply firewall filters to interfaces—Firewall filters control local packets to and from the Routing Engine if they are applied to the loopback interface, lo0. With the Internet Processor II application-specific integrated circuit (ASIC), firewall filters can control data packets through the routing platform when they are applied to an external interface. To have a firewall filter take effect, you must apply it to an interface by including the filter statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name]
filter {
  input filter-name;
  output filter-name;
}
```

## Configuring Firewall Filters

---

To configure firewall filters, include the firewall statement at the [edit] hierarchy level:

```
[edit]
firewall {
  family family-name {
    filter filter-name {
      accounting-profile name;
      interface-specific;
      term term-name {
        from {
          match-conditions;
        }
        then {
          action;
          action-modifiers;
        }
      }
    }
  }
}
```



**NOTE:** You can specify IP version 4 (IPv4) filters at either the [edit firewall] hierarchy level or the [edit firewall family inet] hierarchy level.

---

The following sections describe the components of the firewall statement and provide examples of configuring firewall filters:

Configuring the Family Address Type on page 159

Configuring the Filter Name on page 160

Configuring the Filter Terms on page 160

Configuring a Filter Match Statement on page 160

Configuring a Filter Action Statement on page 161

How Firewall Filters Are Evaluated on page 165

Filter Match Conditions on page 166

How Firewall Filters Test a Packet's Protocol on page 179

Examples: Defining Firewall Filters on page 180

### Configuring the Family Address Type

To configure the family address type for a firewall filter, include the family statement at the [edit firewall] hierarchy level:

```
[edit firewall]
family family-name { ... }
```

Specify `inet` to filter IPv4 packets. Specify `inet6` to filter IP version 6 (IPv6) packets. Specify `mpls` to filter Multiprotocol Label Switching (MPLS) packets. Specify `vpls` to filter virtual private LAN service (VPLS) packets.

### Configuring the Filter Name

To configure the filter name, include the filter statement:

```
filter filter-name { ... }
```

For IPv4 traffic, configure the filter name at the [edit firewall family inet] hierarchy level. For IPv6 traffic, configure the filter name at the [edit firewall family inet6] hierarchy level. The filter name can contain letters, numbers, and hyphens (–) and can be up to 24 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

### Configuring the Filter Terms

Each firewall filter consists of one or more *terms*. To configure a term, include the term statement:

```
term term-name { ... }
```

For IPv4 traffic, configure the filter terms at the [edit firewall family inet filter *filter-name*] hierarchy level. For IPv6 traffic, configure the filter terms at the [edit firewall family inet6 filter *filter-name*] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name*] hierarchy level.

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 (“ ”).

Each term name must be unique within a filter.

You can specify multiple terms in a filter, effectively chaining together a series of match–action operations to apply to the packets on an interface. You can also use the next term action so that, when a match condition is met, the evaluation continues to the next term, rather than terminating.

Firewall filter terms are evaluated in the order in which you specify them in the configuration. To reorder terms, use the configuration mode `insert` command. For example, the command `insert term up before term start` places the term up before the term start.

### Configuring a Filter Match Statement

In a firewall filter term, you can define conditions used to match the components of a packet. To configure match conditions, include the `from` statement:

```
from {
    match-conditions;
}
```

For IPv4 traffic, configure the match conditions at the [edit firewall family inet filter *filter-name* term *term-name*] hierarchy level. For IPv6 traffic, configure the match conditions at the [edit firewall family inet6 filter *filter-name* term *term-name*] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name*] hierarchy level.

You can specify zero or more match conditions in a single from statement. For a match to occur, the packet must match all the conditions in the term. For more information about match conditions, see “Filter Match Conditions” on page 166.

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

### Configuring a Filter Action Statement

In a firewall filter term, you can specify the action to take if the packet matches the conditions you have configured in the term. To configure a filter action, include the then statement:

```
then {
    action;
    action-modifiers;
}
```

For IPv4 traffic, configure the filter action at the [edit firewall family inet filter *filter-name* term *term-name*] hierarchy level. For IPv6 traffic, configure the filter action at the [edit firewall family inet6 filter *filter-name* term *term-name*] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name*] hierarchy level.

You can specify zero or one then statement in a filter term. If you omit the then statement or do not specify an action, the packets that match the conditions in the from statement are accepted.



**NOTE:** We strongly recommend that you always explicitly configure an action in the then statement.

---

You can specify one of the following filter actions:

accept—The packet is accepted and is sent to its destination.

discard—The packet is not accepted and is not processed further. Discarded packets cannot be logged or sampled.

next term—Evaluate the next term in the firewall filter.

reject—The packet is not accepted and a rejection message is returned. Rejected packets can be logged or sampled.

routing-instance—The packet is accepted and routed by the specified routing instance. For more information, see “Configuring Filter-Based Forwarding” on page 193.

In the filter action statement, you can also specify one or more of the following action modifiers:

count—Add packet to a count total.

forwarding-class—Specify the packet forwarding class name.

log—The packet's header information is stored on the Routing Engine or sent to a server.



**NOTE:** The firewall filter stops logging discard and reject actions at a high traffic rate.

---

loss-priority—Set the packet loss priority (PLP) to low or high.

policer—Apply rate-limiting procedures to the traffic. For more information, see “Policer Configuration” on page 203.

sample—Sample the packet traffic. Apply this option only if you have enabled traffic sampling. For more information, see “Traffic Sampling and Forwarding Configuration” on page 235.

syslog—Log an alert for the packet.

ipsec-sa *sa-name*—Specify an IP Security (IPSec) security association (SA) for the packet. This is used with the source-address and destination-address match conditions.

You can include zero or one action statement, but any combination of action modifiers. For the action or action modifier to take effect, all conditions in the from statement must match. If you specify log as one of the actions in a term, this constitutes a termination action; whether any additional terms in the filter are processed depends on the traffic through the filter.

The action modifier operations carry a default accept action. For example, if you specify an action modifier and do not specify an action, the specified action modifier is implemented and the packet is accepted.

Policing uses a specific type of action, known as a policer action. For more information, see “Policer Configuration” on page 203.

For more information about forwarding classes and loss priority, see the *JUNOS Network Interfaces and Class of Service Configuration Guide*.

Table 22 describes the filter actions and action modifiers.

Table 22: Firewall Filter Actions and Action Modifiers

Action or Action Modifier	Description
<b>Actions</b>	
accept	Accept a packet. This is the default.
discard	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are not available for logging or sampling.
next term	Continue to the next term for evaluation.
reject <message-type>	Discard a packet, sending an ICMP destination unreachable message. Rejected packets can be logged or sampled if you configure either of those action modifiers. You can specify one of the following message codes: administratively-prohibited (default), bad-host-tos, bad-network-tos, host-prohibited, host-unknown, host-unreachable, network-prohibited, network-unknown, network-unreachable, port-unreachable, precedence-cutoff, precedence-violation, protocol-unreachable, source-host-isolated, source-route-failed, or tcp-reset. If you specify tcp-reset, a Transmission Control Protocol (TCP) reset is returned if the packet is a TCP packet. Otherwise, nothing is returned.
routing-instance <i>routing-instance</i>	Specify a routing instance to which packets are forwarded.
<b>Action Modifiers</b>	
count <i>counter-name</i>	The number of the packets passing this filter/term/policer. The name can contain letters, numbers, and hyphens (-), and can be up to 24 characters long. A counter name is specific to the filter that uses it, so all interfaces that use the same filter increment the same counter.
forwarding-class <i>class-name</i>	A particular forwarding class.
ipsec-sa <i>sa-name</i>	An IPsec SA for the packet. Used with the source-address and destination-address match conditions.
log	Log the packet's header information in the Routing Engine. You can access this information by issuing the show log command at the command-line interface (CLI).
loss-priority <i>priority</i>	Set the PLP to low or high.
policer <i>policer-name</i>	Apply rate limits to the traffic using the named policer.
sample	Sample the traffic on the interface. Use this modifier only when traffic sampling is enabled. For more information, see "Traffic Sampling and Forwarding Configuration" on page 235.
syslog	Log an alert for this packet. The log can be sent to a server for storage and analysis.

**Example: Configure a Filter Action Statement**

Count, sample, and accept the traffic:

```

term all {
  then {
    count sam-1;
    sample;                # default action is accept
  }
}

```

Display the packet counter:

```
user@host> show firewall filter sam

Filter:
Counters:
Name           Bytes           Packets
sam
sam-1          98              8028
```

Display the firewall log output:

```
user@host> show firewall log

Time  Filter  A Interface  Pro Source address  Destination address
23:09:09 -   A at-2/0/0.301  TCP 10.2.0.25  10.211.211.1:80
23:09:07 -   A at-2/0/0.301  TCP 10.2.0.25  10.211.211.1:56
23:09:07 -   A at-2/0/0.301  ICM 10.2.0.25  10.211.211.1:49552
23:02:27 -   A at-2/0/0.301  TCP 10.2.0.25  10.211.211.1:56
23:02:25 -   A at-2/0/0.301  TCP 10.2.0.25  10.211.211.1:80
23:01:22 -   A at-2/0/0.301  ICM 10.2.2.101  10.211.211.1:23251
23:01:21 -   A at-2/0/0.301  ICM 10.2.2.101  10.211.211.1:16557
23:01:20 -   A at-2/0/0.301  ICM 10.2.2.101  10.211.211.1:29471
23:01:19 -   A at-2/0/0.301  ICM 10.2.2.101  10.211.211.1:26873
```

This output file contains the following fields:

Time—Time at which the packet was received (not shown in the default).

Filter—Name of a filter that has been configured with the filter statement at the [edit firewall] hierarchy level. A hyphen (-) or the abbreviation pfe indicates that it was handled by the Packet Forwarding Engine. A space (no hyphen) indicates that the packet was handled by the Routing Engine.

A—Filter action:

A—Accept (or next term)

D—Discard

R—Reject

Interface—Interface on which the filter is configured.



**NOTE:** When general routing encapsulation (GRE) packets are rejected, the firewall log output might display an incorrect interface.

Pro—Packet’s protocol name or number.

Source address—Source IP address in the packet.

Destination address—Destination IP address in the packet.

Display the sampling output:

```
user@host> show log /var/tmp/sam
```

```
# Apr 7 15:48:50
Time          Dest      Src Dest Src Proto TOS Pkt Intf IP  TCP
          addr      addr port port   len num frag flags
Apr 7 15:48:54 192.168.9.194 192.168.9.195 0 0 1 0x0 84 8 0x0 0x0
Apr 7 15:48:55 192.168.9.194 192.168.9.195 0 0 1 0x0 84 8 0x0 0x0
Apr 7 15:48:56 192.168.9.194 192.168.9.195 0 0 1 0x0 84 8 0x0 0x0
```



**NOTE:** When you enable reverse-path forwarding (RPF) on an interface with an input filter for firewall log and count, the input firewall filter does not log the packets rejected by RPF, although the rejected packets are counted. To log the rejected packets, use an RPF check fail filter to log the rejected packets.

For more information about sampling output, see “Configuring a Forwarding Table Filter” on page 240.

## How Firewall Filters Are Evaluated

When a firewall filter consists of a single term, the filter is evaluated as follows:

If the packet matches all the conditions, the action in the then statement is taken.

If the packet matches all the conditions, and if there is no action specified in the then statement, the default action accept is used.

If the packet does not match all the conditions, it is discarded.

When a firewall filter consists of more than one term, the filter is evaluated sequentially:

The packet is evaluated against the conditions in the from statement in the first term.

If the packet matches, the action in the then statement is taken and, if the next term action is not used, the evaluation ends. Subsequent terms in the filter are not evaluated.

If the packet matches, the action in the then statement is taken; if the next term action is present, the evaluation continues to the next term.

If the packet does not match, it is evaluated against the conditions in the from statement in the second term.

This process continues until either the packet matches the from conditions in one of the subsequent terms or there are no more terms.

If a packet passes through all the terms in the filter without matching any of them, it is discarded.

If a term does not contain a from statement, the packet is considered to match and the action in the term's then statement is taken.

If a term does not contain a then statement or if you do not configure an action in the then statement, and if the packet matches the conditions in the term's from statement, the packet is accepted.

Each firewall filter has an implicit discard action at the end of the filter, which is equivalent to the following explicit filter term:

```
term implicit-rule {
    then discard;
}
```

Therefore, if a packet matches none of the terms in the filter, it is discarded.

### **Filter Match Conditions**

In the from statement in the firewall filter term, you specify conditions that the packet must match for the action in the then statement to be taken. All conditions in the from statement must match for the action to be taken. The order in which you specify match conditions is not important, because a packet must match all the conditions in a term for a match to occur.

If you specify no match conditions in a term, that term matches all packets.

An individual condition in a from statement can contain a list of values. For example, you can specify numeric ranges or multiple source or destination addresses. When a condition defines a list of values, a match occurs if one of the values in the list matches the packet.

Individual conditions in a from statement can be negated. When you negate a condition, you are defining an explicit mismatch. If a packet matches a negated condition, it is immediately considered not to match the from statement, and the next term in the filter is evaluated, if there is one; if there are no more terms, the packet is discarded.

Match conditions are grouped into categories depending upon how you specify the condition. You can specify the following conditions:

Specifying Numeric Range Filter Match Conditions on page 167

Specifying Address Filter Match Conditions on page 171

Specifying Multiple Match Conditions on page 175

Specifying Bit-Field Filter Match Conditions (IPv4 Traffic Only) on page 176

Specifying Class-Based Filter Match Conditions on page 178

Filtering Smaller Packets on page 179

## Specifying Numeric Range Filter Match Conditions

Numeric range filter conditions match packet fields that can be identified by a numeric value, such as port and protocol numbers. For numeric range filter match conditions, you specify a keyword that identifies the condition and a single value or a range of values that a field in a packet must match. Table 23 describes the numeric range filter match conditions for IPv4 addresses, and Table 24 on page 170 describes them for IPv6 addresses.

You can specify the numeric range value in one of the following ways:

**Single number.** A match occurs if the value of the field matches the number. For example:

```
source-port 25;
```

**Range of numbers.** A match occurs if the value of the field falls within the specified range. The following example matches source ports 1024 through 65,535, inclusive:

```
source-port 1024-65535;
```

**Text synonym for a single number.** A match occurs if the value of the field matches the number that corresponds to the synonym. For example:

```
source-port smtp;
```

To specify multiple values in a single match condition, group the values within square brackets following the keyword. For example:

```
source-port [smtp ftp-data 25 1024-65535];
```

To exclude a numeric value, append the string `-except` to the match keyword. For example, the following condition would match only if the source port is not 25:

```
source-port-except 25;
```

The following condition would match only if the port number is not one of those in the list:

```
source-port-except [smtp ftp-data 666 1024-65535];
```

**Table 23: Numeric Range IPv4 Firewall Filter Match Conditions**

Match Condition	Description
<code>keyword-except</code>	Negate a match. For example, <code>destination-port-except number</code> .
<code>ah-spi spi-value</code>	IPSec authentication header (AH) security parameter index (SPI) value. Match on this specific SPI value.
<code>ah-spi-except spi-value</code>	IPSec AH SPI value. Do not match on this specific SPI value.

Match Condition	Description
destination-port <i>number</i>	<p>TCP or User Datagram Protocol (UDP) destination port field. You cannot specify both the port and destination-port match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “How Firewall Filters Test a Packet’s Protocol” on page 179.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the port numbers are also listed): afs (1483), bgp (179), biff (512), bootpc (68), bootps (67), cmd (514), cvspserver (2401), dhcp (67), domain (53), eklogin (2105), ekshell (2106), exec (512), finger (79), ftp (21), ftp-data (20), http (80), https (443), ident (113), imap (143), kerberos-sec (88), klogin (543), kpasswd (761), krb-prop (754), krbupdate (760), kshell (544), ldap (389), login (513), mobileip-agent (434), mobilip-mn (435), msdp (639), netbios-dgm (138), netbios-ns (137), netbios-ssn (139), nfsd (2049), nntp (119), ntalk (518), ntp (123), pop3 (110), pptp (1723), printer (515), radacct (1813), radius (1812), rip (520), rkinit (2108), smtp (25), snmp (161), snmptrap (162), snpp (444), socks (1080), ssh (22), sunrpc (111), syslog (514), tacacs-ds (65), talk (517), telnet (23), tftp (69), timed (525), who (513), xdmcp (177), zephyr-clt (2103), or zephyr-hm (2104).</p>
destination-mac-address <i>address</i>	Destination media access control (MAC) address of a VPLS packet.
dscp <i>number</i>	<p>Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant six bits of this byte form the DSCP. For more information, see the <i>JUNOS Network Interfaces and Class of Service Configuration Guide</i>.</p> <p>You can specify DSCP in either hexadecimal, binary, or decimal form.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):</p> <p>RFC 2598, <i>An Expedited Forwarding, PHB</i> defines one code point: ef (46).</p> <p>RFC 2597, <i>Assured Forwarding PHB</i>, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:</p> <p>af11 (10), af12 (12), af13 (14),  af21 (18), af22 (20), af23 (22),  af31 (26), af32 (28), af33 (30),  af41 (34), af42 (36), af43 (38)</p>
ether-type <i>value</i>	Match on Ethernet type field of a VPLS packet.
ether-type-except <i>value</i>	Do not match on Ethernet type field of a VPLS packet.
esp-spi <i>spi-value</i>	IPSec encapsulating security payload (ESP) SPI value. Match on this specific SPI value. You can specify the ESP SPI value in either hexadecimal, binary, or decimal form.
esp-spi-except <i>spi-value</i>	IPSec ESP SPI value. Do not match on this specific SPI value.
forwarding-class <i>class</i>	Match on forwarding class. Specify assured-forwarding, best-effort, expedited-forwarding, or network-control.
forwarding-class-except <i>class</i>	Do not match on forwarding class. Specify assured-forwarding, best-effort, expedited-forwarding, or network-control.
fragment-offset <i>number</i>	Fragment offset field.

Match Condition	Description
<i>icmp-code number</i>	<p>ICMP code field. This value or keyword provides more specific information than <i>icmp-type</i>. Because the value's meaning depends upon the associated <i>icmp-type</i>, you must specify <i>icmp-type</i> along with <i>icmp-code</i>. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 179.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated:</p> <ul style="list-style-type: none"> <li>parameter-problem: <i>ip-header-bad</i> (0), <i>required-option-missing</i> (1)</li> <li>redirect: <i>redirect-for-host</i> (1), <i>redirect-for-network</i> (0), <i>redirect-for-tos-and-host</i> (3), <i>redirect-for-tos-and-net</i> (2)</li> <li>time-exceeded: <i>ttl-eq-zero-during-reassembly</i> (1), <i>ttl-eq-zero-during-transit</i> (0)</li> <li>unreachable: <i>communication-prohibited-by-filtering</i> (13), <i>destination-host-prohibited</i> (10), <i>destination-host-unknown</i> (7), <i>destination-network-prohibited</i> (9), <i>destination-network-unknown</i> (6), <i>fragmentation-needed</i> (4), <i>host-precedence-violation</i> (14), <i>host-unreachable</i> (1), <i>host-unreachable-for-TOS</i> (12), <i>network-unreachable</i> (0), <i>network-unreachable-for-TOS</i> (11), <i>port-unreachable</i> (3), <i>precedence-cutoff-in-effect</i> (15), <i>protocol-unreachable</i> (2), <i>source-host-isolated</i> (8), <i>source-route-failed</i> (5)</li> </ul>
<i>icmp-type number</i>	<p>ICMP packet type field. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 179.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <i>echo-reply</i> (0), <i>echo-request</i> (8), <i>info-reply</i> (16), <i>info-request</i> (15), <i>mask-request</i> (17), <i>mask-reply</i> (18), <i>parameter-problem</i> (12), <i>redirect</i> (5), <i>router-advertisement</i> (9), <i>router-solicit</i> (10), <i>source-quench</i> (4), <i>time-exceeded</i> (11), <i>timestamp</i> (13), <i>timestamp-reply</i> (14), or <i>unreachable</i> (3).</p>
<i>interface-group group-number</i>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For information about configuration interface groups, see "Applying Firewall Filters to Interfaces" on page 188.
<i>packet-length bytes</i>	Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.
<i>port number</i>	<p>TCP or UDP source or destination port field. You cannot specify both the port match and either the <i>destination-port</i> or <i>source-port</i> match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 179.</p> <p>In place of the numeric value, you can specify one of the text synonyms listed under <i>destination-port</i>.</p>
<i>precedence ip-precedence-field</i>	IP precedence field. In place of the numeric field value, you can specify one of the following text synonyms (the field values are also listed): <i>critical-ecp</i> (0xa0), <i>flash</i> (0x60), <i>flash-override</i> (0x80), <i>immediate</i> (0x40), <i>internet-control</i> (0xc0), <i>net-control</i> (0xe0), <i>priority</i> (0x20), or <i>routine</i> (0x00). You can specify precedence in either hexadecimal, binary, or decimal form.
<i>protocol number</i>	IP protocol field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <i>ah</i> , <i>egp</i> (8), <i>esp</i> (50), <i>gre</i> (47), <i>icmp</i> (1), <i>igmp</i> (2), <i>ipip</i> (4), <i>ipv6</i> (41), <i>ospf</i> (89), <i>pim</i> (103), <i>rsvp</i> (46), <i>tcp</i> (6), or <i>udp</i> (17).
<i>source-mac-address address</i>	Source MAC address of a VPLS packet.
<i>source-port number</i>	<p>TCP or UDP source port field. You cannot specify the port and <i>source-port</i> match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 179.</p> <p>In place of the numeric field, you can specify one of the text synonyms listed under <i>destination-port</i>.</p>
<i>vlan-ether-type value</i>	Match on virtual local area network (VLAN) Ethernet type field of a VPLS packet.
<i>vlan-ether-type-except value</i>	Do not match on VLAN Ethernet type field of a VPLS packet.

**Table 24: Numeric Range IPv6 Firewall Filter Match Conditions**

Match Condition	Description
address <i>address</i>	A 128-bit address that supports the standard syntax for IPv6 addresses. For more information, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
destination-address <i>address</i>	A 128-bit address that is the final destination node address for the packet. The filter description syntax supports the text representations for IPv6 addresses as described in RFC 2373, <i>IP Version 6 Addressing Architecture</i> . For more information about IPv6 address syntax, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
destination-port <i>number</i>	<p>TCP or UDP destination port field. You cannot specify both the port and destination-port match conditions in the same term.</p> <p>Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “How Firewall Filters Test a Packet’s Protocol” on page 179.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the port numbers are also listed): afs (1483), bgp (179), biff (512), bootpc (68), bootps (67), cmd (514), cvspserver (2401), dhcp (67), domain (53), eklogin (2105), ekshell (2106), exec (512), finger (79), ftp (21), ftp-data (20), http (80), https (443), ident (113), imap (143), kerberos-sec (88), klogin (543), kpasswd (761), krb-prop (754), krbupdate (760), kshell (544), ldap (389), login (513), mobileip-agent (434), mobilip-mn (435), msdp (639), netbios-dgm (138), netbios-ns (137), netbios-ssn (139), nfsd (2049), nntp (119), ntalk (518), ntp (123), pop3 (110), pptp (1723), printer (515), radacct (1813), radius (1812), rip (520), rkinit (2108), smtp (25), snmp (161), snmptrap (162), snpp (444), socks (1080), ssh (22), sunrpc (111), syslog (514), tacacs-ds (65), talk (517), telnet (23), tftp (69), timed (525), who (513), xdmcp (177), zephyr-clt (2103), or zephyr-hm (2104).</p>
icmp-code <i>number</i>	<p>ICMP code field. This value or keyword provides more specific information than icmp-type. Because the value’s meaning depends upon the associated icmp-type, you must specify icmp-type along with icmp-code. For more information, see “How Firewall Filters Test a Packet’s Protocol” on page 179.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed). The keywords are grouped by the ICMP type with which they are associated:</p> <ul style="list-style-type: none"> <li>parameter-problem: ip-header-bad (0), required-option-missing (1)</li> <li>redirect: redirect-for-host (1), redirect-for-network (0), redirect-for-tos-and-host (3), redirect-for-tos-and-net (2)</li> <li>time-exceeded: ttl-eq-zero-during-reassembly (1), ttl-eq-zero-during-transit (0)</li> <li>unreachable: communication-prohibited-by-filtering (13), destination-host-prohibited (10), destination-host-unknown (7), destination-network-prohibited (9), destination-network-unknown (6), fragmentation-needed (4), host-precedence-violation (14), host-unreachable (1), host-unreachable-for-TOS (12), network-unreachable (0), network-unreachable-for-TOS (11), port-unreachable (3), precedence-cutoff-in-effect (15), protocol-unreachable (2), source-host-isolated (8), source-route-failed (5)</li> </ul>
icmp-type <i>number</i>	<p>ICMP packet type field. Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “How Firewall Filters Test a Packet’s Protocol” on page 179.</p> <p>In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): echo-reply (0), echo-request (8), info-reply (16), info-request (15), mask-request (17), mask-reply (18), parameter-problem (12), redirect (5), router-advertisement (9), router-solicit (10), source-quench (4), time-exceeded (11), timestamp (13), timestamp-reply (14), or unreachable (3).</p>
interface-group <i>group-number</i>	Interface group on which the packet was received. An interface group is a set of one or more logical interfaces. For information about configuration interface groups, see “Applying Firewall Filters to Interfaces” on page 188.

Match Condition	Description
next-header <i>bytes</i>	An eight-bit IP protocol field that identifies the type of header immediately following the IPv6 header. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>egp (8)</code> , <code>esp (50)</code> , <code>gre (47)</code> , <code>icmp (1)</code> , <code>icmpv6 (1)</code> , <code>igmp (2)</code> , <code>ipip (4)</code> , <code>ipv6 (41)</code> , <code>ospf (89)</code> , <code>pim (103)</code> , <code>rsvp (46)</code> , <code>tcp (6)</code> , or <code>udp (17)</code> .
packet-length <i>bytes</i>	Length of the received packet, in bytes. The length refers only to the IP packet, including the packet header, and does not include any Layer 2 encapsulation overhead.
port <i>number</i>	TCP or UDP source or destination port field. You cannot specify both the port match and either the destination-port or source-port match conditions in the same term.  Typically, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “How Firewall Filters Test a Packet’s Protocol” on page 179.  In place of the numeric value, you can specify one of the text synonyms listed under destination-port.
source-address <i>address</i>	Address of the source node sending the packet; 128 bits in length. The filter description syntax supports the text representations for IPv6 addresses as described in RFC 2373. For more information about IPv6 address syntax, see the <i>JUNOS Routing Protocols Configuration Guide</i> .
source-port <i>number</i>	TCP or UDP source port field. You cannot specify the port and source-port match conditions in the same term.  Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more information, see “How Firewall Filters Test a Packet’s Protocol” on page 179.  In place of the numeric field, you can specify one of the text synonyms listed under destination-port.
traffic-class <i>number</i>	An eight-bit field that specifies the class-of-service (CoS) priority of the packet. This field was previously used as the ToS field in IPv4. However, the semantics of this field (for example, DiffServ code points) are identical to IPv4.
ttl <i>type</i>	IPv4 TTL type to match. Specify a TTL value between 1 and 255. This match condition is supported only on M320 and T-series routing platforms.
ttl-except <i>type</i>	IPv4 TTL type to avoid matching. Specify a TTL value between 1 and 255. This match condition is supported only on M320 and T-series routing platforms.

### Specifying Address Filter Match Conditions

Address filter conditions match prefix values in a packet, such as IP source and destination prefixes. For address filter match conditions, you specify a keyword that identifies the field and one or more prefixes of that type that a packet must match. Table 25 on page 173 describes the address filter match conditions.

You can specify the address in one of the following ways:

Single prefix. A match occurs if the value of the field matches the prefix. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address 10.0.0.0/8;
```

Multiple prefixes. A match occurs if any one of the prefixes in the list matches the packet. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
    10.0.0.0/8;
    192.168.0.0/32;
}
```

The order in which you list prefixes in the list is not significant. They are all evaluated to determine whether a match occurs. If prefixes overlap, longest-match rules are used to determine whether a match occurs. Each list of prefixes contains an implicit 0/0 except statement, which means that any prefix that does not match any prefix in the list is explicitly considered not to match.

To specify the address prefix, use the notation *prefix/prefix-length*. If you omit *prefix-length*, it defaults to /32. For example:

```
[edit firewall family family-name filter filter-name term term-name from]
user@host# set destination-address 10
[edit firewall family family-name filter filter-name term term-name from]
user@host# show
destination-address {
    10.0.0.0/32;
}
```

To exclude a prefix, specify the string *except* after the prefix. In the following example, any addresses that fall under 192.168.10.0/8 match, except for addresses that fall under 192.168.0.0/16. All other addresses implicitly do not match this condition.

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
    192.168.0.0/16 except;
    192.168.10.0/8;
}
```

To match all destinations except one, in this example 10.1.1.0/24, configure the match conditions as follows:

```
[edit firewall family family-name filter filter-name term term-name from]
destination-address {
    0.0.0.0/0;
    10.1.1.0/24 except;
}
```

Because the prefixes are order-independent and use longest-match rules, longer prefixes subsume shorter ones as long as they are the same type (whether you specify *except* or not). This is because anything that would match the longer prefix would also match the shorter one. In the following example:

172.16.1.2 matches the 172.16.0.0/10 prefix, and thus the action in the *then* statement is taken.

172.16.2.2 matches the 172.16.2.0/16 prefix. Because this prefix is negated (that is, marked as *except*), an explicit mismatch occurs. The next term in the filter is evaluated, if there is one. If there are no more terms, the packet is discarded.

10.1.2.3 does not match any of the prefixes included in the source-address condition. Instead, it matches the implicit 0.0.0.0/0 *except* at the end of the list, and is considered to be a mismatch.

The 172.16.3.0/16 statement is ignored because it falls under the address 172.16.0.0/10—both are the same type.

The 10.2.2.2 except statement is ignored because it is subsumed by the implicit 0.0.0.0/0 except statement at the end of the list.

```
[edit firewall family family-name filter filter-name term term-name from]
source-address {
    172.16.0.0/10;
    172.16.2.0/16 except;
    192.168.1.0;
    192.168.1.192/26 except;
    192.168.1.254;
    172.16.3.0/16;           # ignored
    10.2.2.2 except;       # ignored
}
```

**Table 25: Address Firewall Filter Match Conditions**

Match Condition	Description
address <i>prefix</i>	IP source or destination address field. You cannot specify both the address and the destination-address or source-address match conditions in the same term.
destination-address <i>prefix</i>	IP destination address field. You cannot specify the destination-address and address match conditions in the same term.
destination-prefix-list <i>prefix-list</i>	IP destination prefix list field. You cannot specify the destination-prefix-list and prefix-list match conditions in the same term.
prefix-list <i>prefix-list</i>	IP source or destination prefix list field. You cannot specify both the prefix-list and the destination-prefix-list or source-prefix-list match conditions in the same term.
source-address <i>prefix</i>	IP source address field. You cannot specify the source-address and address match conditions in the same rule.
source-prefix-list <i>prefix-list</i>	IP source prefix list field. You cannot specify the source-prefix-list and prefix-list match conditions in the same term.

You can also define a list of IP address prefixes under a *prefix-list* alias for frequent reference. You make this definition at the [edit policy-options] hierarchy level:

```
[edit policy-options]
prefix-list prefix-list {
    address;
    address;
    address;
}
}
```

Once you have defined a prefix list, you can use it when defining firewall filters:

```
[edit firewall family family-name filter filter-name term term-name]
from {
  source-prefix-list {
    prefix-list1;
    prefix-list2;
  }
  destination-prefix-list {
    prefix-list1;
  }
}
```

You can specify noncontiguous address prefixes in a filter term for firewall filters. Noncontiguous address prefixes are prefixes that are not adjacent or neighboring to one another. For example, in the following example, the following prefixes are noncontiguous: 0.0.0.10/0.0.0.255, 0.10.0.10/0.255.0.255, and 0.12.10.9/0.255.255.255:

```
[edit firewall family inet filter filter-name]
term term-name {
  address 0.0.0.10/0.0.0.255;
  destination-address 0.10.0.10/0.255.0.255;
  source-address 0.12.10.9/0.255.255.255 except;
}
```



**NOTE:** Noncontiguous address prefixes are valid only for IPv4 filters. IPv6 filters do not support noncontiguous address prefixes.

---

You can also specify a netmask value rather than a prefix length, for example:

```
[edit firewall family inet filter filter-name]
term term-name {
  address 10.0.0.10/255.0.0.255;
}
```

The prefix notation shown matches any address with a first and last octet of 10. The address and netmask are separated by a forward slash (/). The second and third bytes of the prefix can be any value from 0 through 255.



**NOTE:** When a firewall filter term includes the from address *address* match condition and a subsequent term includes the from source-address *address* match condition for the same address, packets may be processed by the latter term before they are evaluated by any intervening terms. Therefore, packets that should be rejected by the intervening terms may be accepted, or packets that should be accepted may be rejected.

To prevent this from occurring, we recommend you do the following. For every firewall filter term that contains the from address *address* match condition, replace that term with two separate terms: one that contains the from source-address *address* match condition, and another that contains the from destination-address *address* match condition.

---

For more information about prefixes, see the *JUNOS Routing Protocols Configuration Guide*.

### Specifying Multiple Match Conditions

A complication arises with filters that specify both address and port matches:

An address match occurs if either the source or destination address in the packet matches one of the prefixes in the list.

A port match occurs if either the source or destination port in the packet matches one of the port ranges in the list.

For example, you could apply the following terms within a firewall filter:

```
[edit firewall family family-name filter filter-name]
term 1 {
  from {
    address {
      10.0.0.0/12;
    }
    protocol tcp;
  }
  then {
    count lf1-1;
    accept;
  }
}
term 2 {
  from {
    address {
      10.26.0.0/15;
    }
    protocol tcp;
  }
  then {
    count lf1-2;
    accept;
  }
}
```

A packet whose source address is 10.0.1.1 and whose destination address is 10.27.1.1 should increment the first counter, as should a packet with source address 10.27.1.1 and destination address 10.0.1.1. Sometimes, however, one of these packets increments the second counter rather than the first. The problem is that the address matches are seen as mutually exclusive alternatives, which are compiled into a form that evaluates the match in parallel without regard to term ordering. This works for single-field matches such as source-address or destination-address, but not for multiple-field matches where there is no mutual exclusivity. It still produces correct matches in this case (the packet always matches the from condition in the term whose action it takes), but loses the term ordering that should be used to distinguish multiple matches.



**NOTE:** As a workaround in this situation, avoid using address and port for match conditions. You can use source-address, destination-address, source-port, or destination-port instead.

If the application absolutely requires matching the same prefix against either source-address or destination-address, write two terms in sequence.

### Specifying Bit-Field Filter Match Conditions (IPv4 Traffic Only)

Bit-field filter conditions match packet fields if particular bits in those fields are or are not set. You can match the IP options, TCP flags, and IP fragmentation fields. For bit-field filter match conditions, you specify a keyword that identifies the field and tests to determine that the option is present in the field. Table 26 describes the bit-field match conditions.



**NOTE:** The JUNOS software does not automatically check the first fragment bit when matching TCP flags. To include the first fragment bit, include the fragment-offset match condition described in Table 23 on page 167.

To specify the bit-field value to match, enclose the value in quotation marks (double quotes). For example, a match occurs if the RST bit in the TCP flags field is set:

```
tcp-flags "rst";
```

Generally, you specify the bits being tested using keywords. Bit-field match keywords always map to a single bit value. You also can specify bit fields as hexadecimal or decimal numbers.

To negate a match, precede the value with an exclamation point. For example, a match occurs only if the RST bit in the TCP flags field is *not* set:

```
tcp-flags "!rst";
```

To match multiple bit-field values, use the logical operators list in Table 27 on page 178. The operators are listed in order, from highest precedence to lowest precedence. Operations are left-associative.

As an example of a logical AND operation, in the following, a match occurs if the packet is the initial packet on a TCP session:

```
tcp-flags "syn & !ack";
```

As an example of a logical OR operation, in the following, a match occurs if the packet is *not* the initial packet on a TCP session:

```
tcp-flags "!syn | ack";
```

As an example of grouping, in the following, a match occurs for any packet that is either a TCP reset or is not the initial packet in the session:

```
tcp-flags "!(syn & !ack) | rst";
```

When you specify a numeric value that has more than one bit set, the value is treated as a logical AND of the set bits. For example, the following two values are the same and a match occurs only if either bit 0x01 or 0x02 is not set:

```
tcp-flags "!0x3";
tcp-flags "!(0x01 & 0x02)";
```

You can use text synonyms to specify some common bit-field matches. You specify these matches as a single keyword. For example:

```
tcp-established;
```

**Table 26: Bit-Field Firewall Filter Match Conditions**

Match Condition	Description
<b>Conditions with Variables</b>	
fragment-flags <i>number</i>	IP fragmentation flags. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): dont-fragment (0x4000), more-fragments (0x2000), or reserved (0x8000).
ip-options <i>number</i>	IP options. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): loose-source-route (131), record-route (7), router-alert (148), strict-source-route (137), or timestamp (68).
tcp-flags <i>number</i>	TCP flags.  Normally, you specify this match in conjunction with the protocol match statement to determine which protocol is being used on the port. For more details, see "How Firewall Filters Test a Packet's Protocol" on page 179.  In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): ack (0x10), fin (0x01), push (0x08), rst (0x04), syn (0x02), or urgent (0x20).
<b>Text Synonyms</b>	
first-fragment	First fragment of a fragmented packet. This condition does not match unfragmented packets.
is-fragment	This condition matches if the packet is a trailing fragment; it does not match the first fragment of a fragmented packet. To match both first and trailing fragments, you can use two terms, or you can use "fragment-range 0-8191".
tcp-established	TCP packets other than the first packet of a connection. This is a synonym for "(ack   rst)".  This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.
tcp-initial	First TCP packet of a connection. This is a synonym for "(syn & !ack)".  This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol tcp match condition.

**Table 27: Bit-Field Logical Operators**

Logical Operator	Description
(...)	Grouping
!	Negation
& or +	Logical AND
or ,	Logical OR

### Specifying Class-Based Filter Match Conditions

Class-based filter conditions match packet fields based on source class or destination class. A source class is a set of source prefixes grouped together and given a class name. A destination class is a set of destination prefixes grouped together and given a class name.

You can specify the source class in the following way:

```
[edit firewall filter inet filter-name term term-name]
from {
    source-class class-name;
}
```

You can specify the destination class in the following way:

```
[edit firewall filter inet filter-name term term-name]
from {
    destination-class class-name;
}
```

You can specify a source class or destination class for an output firewall filter. Although you can specify a source class and destination class for an input firewall filter, the counters are incremented only if the firewall filter is applied on the output interface.

The class-based filter match condition works only for output filters because the source class usage (SCU) and destination class usage (DCU) is determined after route lookup.



**NOTE:** Source class usage and destination class usage are not supported on the interfaces you configure as tunnel sources. This affects only the transit packets exiting the tunnel.

---



**NOTE:** Class-based filter match conditions are supported for inet and inet6 address families on the M-series platforms.

---

### Filtering Smaller Packets

Firewall filtering is not supported for packets of 1-4 bytes length. To filter packets of 1-4 bytes length, include an additional term to match the packet size.

For example, let's consider the following filter term:

```
term 1 {
  from {
    fragment-offset-except 0;
  }
  then {
    reject;
  }
}
```

To consider packets with 1-4 bytes length, include an additional term to match the packet size:

```
term 2 {
  from {
    packet-length [ 21 - 24 ];
  }
  then {
    reject;
  }
}
```

### How Firewall Filters Test a Packet's Protocol

If you specify a port match condition or a match of the ICMP type or TCP flags field, there is no implied protocol match. If you use one of the following match conditions in a term, you should explicitly specify the protocol in the same term:

`destination-port`—Specify the match protocol `tcp` or protocol `udp` in the same term.

`icmp-code`—Specify the match protocol `icmp` in the same term.

`icmp-type`—Specify the match protocol `icmp` in the same term.

`port`—Specify the match protocol `tcp` or protocol `udp` in the same term.

`source-port`—Specify the match protocol `tcp` or protocol `udp` in the same term.

`tcp-flags`—Specify the match protocol `tcp` in the same term.

When examining match conditions, the policy framework software tests only the specified field itself. The software does not also test the IP header to determine that the packet is indeed an IP packet.

If you do not explicitly specify the protocol, when using the fields listed above, design your filters carefully to ensure that they are performing the expected matches. If you specify a match of destination-port ssh, the policy framework software deterministically matches any packets that have a value of 22 in the 2-byte field that is 2 bytes beyond the end of the IP header, without ever checking the IP protocol field.

### Example: Do Not Test Packet Protocol

The first term matches all packets except for TCP and UDP packets, so only TCP and UDP packets are evaluated by third term (term test-a-port):

```
[edit]
firewall {
  family inet {
    filter test-filter {
      term all-but-tcp-and-udp {
        from {
          protocol-except [tcp udp];
        }
        then accept;
      }
      term test-an-address {
        from {
          address 192.168/16;
        }
        then reject;
      }
      term test-a-port {
        from {
          destination-port [ssh dns];
        }
        then accept;
      }
      term dump-anything-else {
        then reject;
      }
    }
  }
}
```

### Examples: Defining Firewall Filters

The following examples illustrate how to define firewall filters:

Example 1 on page 181

Example 2 on page 181

Example 3 on page 182

Example 4 on page 183

Example 5 on page 183

Example 6 on page 184

Example 7 on page 185

Example 8 on page 185

Example 9 on page 186

### Example 1

Block telnet and secure shell (ssh) access to all but the 192.168.1.0/24 subnet. This filter also logs any ssh or telnet traffic attempts from other subnets to the firewall log buffer:

```
[edit]
firewall {
  family inet {
    filter local-access-control {
      term terminal-access {
        from {
          address {
            192.168.1.0/24;
          }
          protocol tcp;
          port [ssh telnet];
        }
        then accept;
      }
      term terminal-access-denied {
        from {
          protocol tcp;
          port [ssh telnet];
        }
        then {
          log;
          reject;
        }
      }
      term default-term {
        then accept;
      }
    }
  }
}
```

### Example 2

Block Trivial File Transfer Protocol (TFTP) access, logging any attempts to establish TFTP connections:

```
[edit]
firewall {
  family inet {
    filter tftp-access-control {
      from {
        protocol udp;
        port tftp;
      }
    }
  }
}
```

```

        then {
            log;
            discard;
        }
    }
}

```

By default, to decrease vulnerability to denial-of-service (DoS) attacks, the JUNOS software filters and discards Dynamic Host Configuration Protocol (DHCP) or Bootstrap Protocol (BOOTP) packets that have a source address of 0.0.0.0 and a destination address of 255.255.255.255. This default filter is known as a unicast RPF check. However, some vendors' equipment automatically accepts these packets. To interoperate with other vendors' equipment, you can configure a filter that checks for both these addresses and overrides the default RPF-check filter by accepting these packets.

### Example 3

Configure a filter (rpf-dhcp) that accepts DHCP packets with a source address of 0.0.0.0 and a destination address of 255.255.255.255:

```

[edit firewall family inet]
filter rpf-dhcp {
    term dhcp {
        from {
            source-address {
                0.0.0.0/32;
            }
            destination-address {
                255.255.255.255/32;
            }
        }
        then {
            accept;
        }
    }
}

```

To apply this filter to an interface, include the rpf-check fail-filter statement at the [edit interface *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```

[edit interface interface-name unit logical-unit-number family inet]
rpf-check fail-filter rpf-dhcp;

```

**Example 4**

Define a policer for a destination class class1:

```
[edit]
firewall {
  family inet {
    filter filter1 {
      policer police-class1
      if-exceeding {
        bandwidth-limit 25;
        burst-size-limit 1000;
      }
      then {
        discard;
      }
    }
    term term1 {
      from {
        destination-class class1;
      }
      then {
        policer police-class1;
      }
    }
  }
}
```

**Example 5**

Count individual IP option packets, but do not block any traffic. Also, log packets that have loose or strict source routing:

```
[edit]
firewall {
  family inet {
    filter ip-option-filter {
      term match-strictsource {
        from {
          ip-options strict-source-route;
        }
        then {
          count strict-source-route;
          log;
          accept;
        }
      }
      term match-loose-source {
        from {
          ip-options loose-source-route;
        }
        then {
          count loose-source-route;
          log;
          accept;
        }
      }
    }
  }
}
```



```

        term default-term {
            then {
                reject administratively-prohibited;# default reject action
            }
        }
    }
}

```

### Example 7

Match packets that are either OSPF packets or packets that come from an address in the prefix 10.108/16, and send an administratively-prohibited ICMP message for all packets that do not match:

```

[edit]
firewall {
    family inet {
        filter ospf-or-131 {
            term protocol-match {
                from {
                    protocol ospf;
                }
            }
            term address-match {
                from {
                    source-address {
                        10.108.0.0/16;
                    }
                }
            }
        }
    }
}

```

### Example 8

Reject all addresses except 192.168.5.0/24. In the first term, the statement 192.168.5.2/24 except causes this address to be considered a mismatch and this address is passed to the next term in the filter. The address 0.0.0.0/0 in the first term matches all other packets, and these are counted, logged, and rejected. In the second term, all packets that passed through the first term (that is, packets whose address matches 192.168.5.2/24) are counted, logged, and accepted.

```

[edit]
firewall {
    family inet {
        filter fire1 {
            term 1 {
                from {
                    address {
                        192.168.5.0/24 except;
                        0.0.0.0/0;
                    }
                }
            }
        }
    }
}

```



Apply the filter `bgp179` to interface `lo0`:

```
[edit interfaces lo0]
root@canopy# show
unit 0 {
  family inet {
    filter {
      input bgp179;
    }
    address 10.0.0.1/32;
  }
}
```

## Configuring Service Filters

---

A service filter identifies packets on which services need to be applied. The service filter also identifies the Physical Interface Card (PIC) performing the service. To configure service filters, include the firewall statement at the `[edit]` hierarchy level:

```
[edit]
firewall {
  family inet {
    service-filter filter-name {
      term term-name {
        from {
          match-conditions;
        }
        then {
          action;
          action-modifiers;
        }
      }
    }
  }
}
```



**NOTE:** You must specify `inet` as the address family in order to configure a service filter.

---

Service filters are configured the same way as firewall filters. Service filters have a subset of the match conditions and actions for firewall filters.

One of the actions must be `service` or `skip`:

Specifying the `service` action directs packets for stateful-firewall service.

Specifying the `skip` action let packets bypass stateful-firewall service.

For more information about configuring firewall filters, see “Configuring Firewall Filters” on page 159. For more information about services and service interfaces, see the *JUNOS Services Interfaces Configuration Guide*.

## Applying Firewall Filters to Interfaces

---

For a firewall filter to work, you must apply it to at least one interface. To do this, include the filter statement when configuring the logical interface at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name*] hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name]
filter {
    input filter-name;
    output filter-name;
}
```

In the input statement, list the name of one firewall filter to be evaluated when packets are received on the interface. Input filters applied to the loopback interface, lo0, affect only inbound traffic destined for the Routing Engine.

In the output statement, list the name of one firewall filter to be evaluated when packets are transmitted on the interface. Output filters applied to the loopback interface, lo0, affect only outbound traffic sent from the Routing Engine.

You can apply only one input and one output firewall filter to each interface. You can use the same filter one or more times.

When you apply a filter to an interface, it is evaluated against all the data packets passing through that interface. The exception is the loopback interface, lo0, which is the interface to the Routing Engine and carries no data packets. If you apply a filter to the lo0 interface, the filter affects the local packets received or transmitted by the Routing Engine.

Filters apply to all packets entering an interface, not just the packets destined for the Routing Engine. To filter packets destined for the Routing Engine, configure the group statement at the [edit interfaces *interface-name* unit *logical-unit-number* family *family-name* filter] hierarchy level. For more information, see “Defining Interface Groups” on page 190.

For filters applied to data packets to function, the routing platform must contain an Internet Processor II ASIC.

You can configure the following additional properties when applying filters to interfaces:

Configuring Interface-Specific Counters on page 188

Defining Interface Groups on page 190

### Configuring Interface-Specific Counters

When you configure a firewall filter that is applied to multiple interfaces, you can name individual counters specific to each interface. These counters enable you to easily maintain statistics on the traffic transiting the different interfaces.



**NOTE:** Configuration of interface-specific counters also creates separate instances of any policers you have configured for the same interface. For more information about policers, see “Policer Configuration” on page 203.

---

To configure interface-specific counters, include the interface-specific statement at the [edit firewall family *family-name* filter *filter-name*] hierarchy level:

```
[edit firewall family family-name filter filter-name]
interface-specific;
```



**NOTE:** The counter name is restricted to 24 bytes. If the renamed counter exceeds this maximum length, the policy framework software might reject it.

### Example: Configuring Interface-Specific Counters

Configure an interface-specific counter:

```
[edit firewall]
family inet {
  filter test {
    interface-specific;
    term 1 {
      from {
        address {
          10.0.0.0/12;
        }
        protocol tcp;
      }
      then {
        count sample1;
        accept;
      }
    }
  }
}
```

When you apply this filter to the input interface of at-1/1/1.0 and the output interface of so-2/2/2.2, the counters are named sample1-at-1/1/1.0-i and sample1-so-2/2/2/.2-o. The suffixes -i (input) and -o (output) are added to the counter names automatically.

The JUNOS software does not sample packets originating from the router. If you configure a sampling filter and apply it to the output side of an interface, then only the transit packets going through that interface are sampled. Packets that are sent from the Routing Engine to the Packet Forwarding Engine are not sampled.

## Defining Interface Groups

When applying a firewall filter, you can define an interface to be part of an *interface group*. Packets received on that interface are tagged as being part of the group. You then can match these packets using the `interface-group match` statement, as described in Table 23 on page 167.

To define an interface to be part of an interface group, include the `group` statement at the `[edit interfaces interface-name unit logical-unit-number family family-name filter]` hierarchy level:

```
[edit interfaces interface-name unit logical-unit-number family family-name filter]
group group-number;
input filter-name;
output filter-name;
```

In the `group` statement, specify the interface group number to be associated with the filter.

In the `input` statement, list the name of one firewall filter to be evaluated when packets are received on the interface.

In the `output` statement, list the name of one firewall filter to be evaluated when packets are transmitted on the interface.

### Example: Defining Interface Groups

Create a filter that contains an interface group:

```
[edit firewall]
family inet {
  filter if-group {
    term group1 {
      from {
        interface-group 1;
        address {
          192.168.80.114/32;
        }
        protocol tcp;
        port finger;
      }
      then {
        count if-group-counter1;
        log;
        reject;
      }
    }
    term group-2 {
      then {
        count if-group-counter2;
        log;
        accept;
      }
    }
  }
}
```

Assign one or more interfaces to the interface group referenced in the filter:

```
[edit interfaces]
fxp0 {
  unit 0 {
    family inet {
      filter {
        group 1;
      }
      address 192.168.5.38/24;
    }
  }
}
```

Apply the filter that contains an interface group:

```
[edit interfaces]
family inet {
  lo0 {
    unit 0 {
      family inet {
        filter {
          input if-group;
          group 1;
        }
        address 10.0.0.1/32;
        address 192.168.77.1/32;
      }
    }
  }
}
```

## Configuring Accounting

---

Juniper Networks routing platforms can collect various kinds of data about traffic passing through the routing platform. You can set up one or more *accounting profiles* that specify some common characteristics of this data, including the following:

- Fields used in the accounting records

- Number of files that the routing platform retains before discarding, and the number of bytes per file

- Polling period that the system uses to record the data

To implement an accounting profile, you must configure the profile and then apply it to an interface or firewall filter. For more information, see the *JUNOS Network Management Configuration Guide*.

## Configuring a Firewall Filter Accounting Profile

There are several types of accounting profiles: interface, firewall filter, destination class, and Routing Engine. To configure an accounting profile, include statements at the [edit accounting-options] hierarchy level. To activate firewall filter profiles, you must reference them at the [edit firewall family *family-name*] hierarchy level. If you reference the same profile name from both a firewall filter and an interface in the same configuration, it causes an error.

The following example shows accounting profile `fw_profile` for the firewall filter `myfilter`. For more information about configuring accounting profiles, see the *JUNOS Network Management Configuration Guide*.

```
[edit]
accounting-options {
  filter-profile fw_profile {
    file fw_accounting;
    interval 60;
    counters {
      counter1;
      counter2;
      counter3;
    }
  }
}
```

To apply the `fw_profile` accounting profile to a firewall filter, include the `accounting-profile` statement as shown:

```
[edit]
firewall {
  family inet {
    filter myfilter {
      accounting-profile fw_profile;
      ...
      term accept-all {
        then {
          count counter1;
          accept;
        }
      }
    }
  }
}
```

## Configuring Filter-Based Forwarding

---

You can configure filters to classify packets based on source address and specify the forwarding path the packets take within the router. For example, you can use this filter for applications to differentiate traffic from two clients that have a common access layer (for example, a Layer 2 switch) but are connected to different Internet service providers (ISPs). When the filter is applied, the router can differentiate the two traffic streams and direct each to the appropriate network. Depending on the media type the client is using, the filter can use the source IP address to forward the traffic to the corresponding network through a tunnel. You can also configure filters to classify packets based on IP protocol type or IP precedence bits.



**NOTE:** You can forward packets based on input filters only; you cannot forward packets based on output filters.

---



**NOTE:** Source-class usage filter matching and unicast reverse-path forwarding checks are not supported on an interface configured with filter-based forwarding (FBF).

---

Filter-based forwarding is supported for IPv4 and IPv6.

To direct traffic meeting defined match conditions to a specific routing instance, include the routing-instance filter action:

```
routing-instance routing-instance;
```

For IPv4 traffic, include the action at the [edit firewall family inet filter *filter-name* term *term-name* then] hierarchy level. For IPv6 traffic, include the action at the [edit firewall family inet6 filter *filter-name* term *term-name* then] hierarchy level. For MPLS traffic, configure the filter terms at the [edit firewall family mpls filter *filter-name* term *term-name* then] hierarchy level.

The routing-instance filter action accepts the traffic meeting the match conditions and directs it to the routing instance named in *routing-instance*. For information about forwarding instances and routing instances, see the *JUNOS Routing Protocols Configuration Guide*.

To complete the configuration, you must also create a routing table group that adds interface routes to the following routing instances:

```
Routing instance named in the action
```

```
Default routing table inet.0
```

You create a routing table group to resolve the routes installed in the routing instance to directly connected next hops on that interface. For more information on routing table groups and interface routes, see the *JUNOS Routing Protocols Configuration Guide*.

## Examples: Configuring Filter-Based Forwarding

Configure a filter to direct traffic to ISP1 or ISP2 based on source address matching:

```
[edit firewall]
family inet {
  filter classify-customers {
    term isp1-customers {
      from {
        source-address 10.1.1.0/24;
        source-address 10.1.2.0/24;
      }
      then {
        routing-instance isp1-route-table;
      }
    }
    term isp2-customers {
      from {
        source-address 10.2.1.0/24;
        source-address 10.2.2.0/24;
      }
      then {
        routing-instance isp2-route-table;
      }
    }
    term default {
      then {
        accept;
      }
    }
  }
}
```

Configure a filter-based forwarding (FBF) filter for family inet6:

```
[edit]
firewall {
  family inet {
    filter ftf_fbf {
      term 0 {
        from {
          source-address {
            ::10.34.1.0/120;
          }
        }
        then {
          count ce1;
          log;
          routing-instance ce1;
        }
      }
      term 1 {
        from {
          source-address {
            ::10.34.2.0/120;
          }
        }
      }
    }
  }
}
```



## Configuring a Forwarding Table Filter

A forwarding table filter allows you to filter data packets based on their components and to perform an action on packets that match the filter.

To configure a forwarding table filter, do the following:

1. Define a forwarding table filter:
  - a. Configure the family address type: IPv4 (`inet`), IPv6 (`inet6`), or MPLS (`mpls`).
  - b. Define one or more *terms*, which are named structures in which match conditions and actions are defined.
  - c. Define a *match condition*, which is the criterion against which a bearer packet is compared; for example, the IP address of a source device or a destination device. You can specify multiple criteria in a match condition.
  - d. Define an *action*, which is what happens if all criteria match; for example, the gateway GPRS support node (GGSN) accepting the bearer packet, performing a lookup in the forwarding table, and forwarding the packet to its destination; discarding the packet; and discarding the packet and returning a rejection message. In addition to an action, you can define one or more *action modifier*s, which are actions that are taken in addition to the GGSN accepting or discarding a packet when all criteria match; for example, counting the packets and logging a packet.

For more information about configuring firewall filters, see “Configuring Firewall Filters” on page 159.

2. Apply the forwarding table filter as an input filter to a forwarding table. The forwarding table filter controls which bearer packets the router accepts and forwards.

To define a forwarding table filter, include the firewall statement at the [edit] hierarchy level:

```
[edit]
firewall {
  family family-name {
    filter filter-name {
      term term-name {
        from {
          match-conditions;
        }
        then {
          action;
          action-modifiers;
        }
      }
    }
  }
}
```

To create a forwarding table, include the `instance-type` statement at the [edit routing-instance *instance-name*] hierarchy level:

```
[edit]
routing-instance instance-name {
    instance-type forwarding;
}
```

To apply a forwarding table filter to a VPN routing and forwarding (VRF) table, include the filter input statement at the [edit routing-instance *instance-name* forwarding-options family *family-name*] hierarchy level:

```
[edit]
routing-instance routing-instance-name {
    instance-type forwarding;
    forwarding-options {
        family family-name {
            filter {
                input filter-name;
            }
        }
    }
}
```

To apply a forwarding table filter to a forwarding table, include the filter input statement at the [edit forwarding-options family *family-name*] hierarchy level:

```
[edit forwarding-options family family-name]
filter {
    input filter-name;
}
```

To apply a forwarding table filter to the default forwarding table `inet.0`, which is not associated with a specific routing instance, include the filter input statement at the [edit forwarding-options family `inet`] hierarchy level:

```
[edit]
filter {
    input filter-name;
}
```

For information about the `routing-instance` and `routing-options` statements, see the *JUNOS Routing Protocols Configuration Guide*.

## Configuring Firewall Filter System Logging

---

System logging can be configured for the firewall filter process. You can set system logging to record messages of a particular level or all levels. The messages are sent to a system logging file.

The following is a sample system logging configuration for the firewall filter icmp-syslog. For more information about configuring system logging, see the *JUNOS System Basics Configuration Guide*.

```
[edit]
system {
  syslog {
    file filter {
      firewall any;
      archive no-world-readable;
    }
  }
}
```

This causes the syslog daemon to write any messages with the syslog facility of firewall to the file /var/log/filter. This keeps the messages out of the main system log file and makes them easier to find.

### **Example: Configuring Firewall Filter System Logging**

Create a filter that logs and counts ICMP packets that have 192.168.207.222 as either their source or destination:

```
[edit]
firewall {
  family inet {
    filter icmp-syslog {
      term icmp-match {
        from {
          address {
            192.168.207.222/32;
          }
        }
        protocol icmp;
      }
      then {
        count packets;
        syslog;
        accept;
      }
    }
    term default {
      then accept;
    }
  }
}
```

Enter the show log filter command to display the results:

```
root@systech> show log filter
Mar 20 08:03:11 systech feb FW: so-0/1/0.0 A icmp 192.168.207.222 192.168.207.223 0 0 (1 packets)
```

This output file contains the following fields:

Date and Time—Date and time at which the packet was received (not shown in the default).

Filter action:

A—Accept (or next term)

D—Discard

R—Reject

Protocol—Packet's protocol name or number.

Source address—Source IP address in the packet.

Destination address—Destination IP address in the packet.



**NOTE:** If the protocol is ICMP, the ICMP type and code are displayed. For all other protocols, the source and destination ports are displayed.

---

The last two fields (both zero) are the source and destination TCP/UDP ports, respectively, and are shown for TCP or UDP packets only. This log message indicates that only one packet for this match has been detected in about a one-second interval. If packets arrive faster, the system log function compresses the information so that less output is generated, and displays an output similar to the following:

```
root@systech> show log filter
Mar 20 08:08:45 systech feb FW: so-0/1/0.0 A icmp 192.168.207.222 192.168.207.223 0 0 (515 packets)
```

