

Chapter 23

Firewall Filter Configuration Guidelines

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

```
firewall {
  filter filter-name {
    term term-name {
      from {
        match-conditions;
      }
      then {
        action;
        action-modifiers;
      }
    }
  }
}
```

The following RFCs define the standards supported by certain aspects of the filtering software:

RFC 792, *Internet Control Message Protocol (ICMP)*

RFC 2474, *Definition of the Differentiated Services (DS) Field*

RFC 2475, *An Architecture for Differentiated Services*

RFC 2597, *Assured Forwarding PHB*

RFC 2598, *An Expedited Forwarding PHB*

This chapter describes the following tasks for configuring firewall filters:

Minimum Firewall Filter Configuration on page 268

Configure Firewall Filters on page 268

Apply Firewall Filters on page 287

Define Interface Groups on page 288

Configure Policing on page 289

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 one or more filter statements at the [edit firewall] hierarchy level:

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

Apply Firewall Filters—Firewall filters will control traffic to and from the Routing Engine if they are applied to the loopback interface, lo0. With the Internet Processor II ASIC, firewall filters can control traffic through the router 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 inet] hierarchy level:

```
interfaces {
  interface-name {
    unit logical-unit-number {
      family inet {
        filter {
          input filter-name;
          output filter-name;
        }
      }
    }
  }
}
```

Configure Firewall Filters

To configure firewall filters, include one or more filter statements at the [edit firewall] hierarchy level:

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

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

Filter Name on page 269

Filter Terms on page 269

Filter Match Statement on page 269

Filter Action Statement on page 270

How Firewall Filters are Evaluated on page 273

Filter Match Conditions on page 273

How Firewall Filters Test a Packet's Protocol on page 282

Examples: Define Firewall Filters on page 284

Filter Name

Each firewall filter is introduced by the keyword `filter` and is identified by a name, *filter-name*:

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

The 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 (" ").

Filter Terms

Each filter statement consists of one or more *terms*. Each term is introduced by the keyword `term` and identified by a name, *term-name*:

```
filter filter-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 (" ").

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.

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.

Filter Match Statement

In a firewall filter term, the `from` statement defines conditions used to match a packet's contents:

```

filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
  }
}

```

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.

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

Policing uses a specific type of match statement called an if-exceeding statement. This is discussed in “Rate-Limiting Statement” on page 290.

Filter Action Statement

In the then statement of a firewall filter term, you specify the action to take if the packet matches the conditions in the from statement. In addition, you can specify action modifiers:

```

filter filter-name {
  term term-name {
    then {
      action;
      action-modifiers;
    }
  }
}

```

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.

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.

`reject`—The packet is not accepted and a rejection message is returned.

In the filter action statement, you can also specify one or more action modifiers, as follows:

`alert`—Log an alert for the packet.

`count`—The packet is counted.

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

`output-queue`—Set the output queue to a specified value.

`plp`—Set the packet loss priority (PLP) bit to a specified value

`sample`—The packet traffic is sampled. Apply this option only if you have enabled traffic sampling; for more information, see “Configure Traffic Sampling” on page 299.

The action modifier operations are independent of the accept, discard, and reject actions. You can specify 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.

For example, the following configuration statement both counts and samples the traffic:

```
term all {
  then {
    accept;
    count sam-1;
    sample;
  }
}
```

To display the packet counter:

```
user@host> show firewall filter sam
```

Filter/Counter	Packet count	Byte count
sam		
sam-1	98	8028

To 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	211.211.211.1:80
23:09:07	-	A at-2/0/0.301	TCP 10.2.0.25	211.211.211.1:56
23:09:07	-	A at-2/0/0.301	ICM 10.2.0.25	211.211.211.1:49552
23:02:27	-	A at-2/0/0.301	TCP 10.2.0.25	211.211.211.1:56
23:02:25	-	A at-2/0/0.301	TCP 10.2.0.25	211.211.211.1:80
23:01:22	-	A at-2/0/0.301	ICM 10.2.2.101	211.211.211.1:23251
23:01:21	-	A at-2/0/0.301	ICM 10.2.2.101	211.211.211.1:16557
23:01:20	-	A at-2/0/0.301	ICM 10.2.2.101	211.211.211.1:29471
23:01:19	-	A at-2/0/0.301	ICM 10.2.2.101	211.211.211.1:26873

This output file contains the following fields:

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

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

A—Filter action:

A—Accept

D—Discard

R—Reject

Interface—Interface on which the filter is configured.

Pro—Packet's protocol name or number.

Source address—Source IP address in the packet.

Destination address—Destination IP address in the packet.

To 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
    
```

For more information on sampling output, see “Configure the Files to Contain Traffic Sampling Output” on page 305.

Policing employs a specific type of policer actions. These are discussed in “Policer Action Statement” on page 291.

Table 10 describes the filter actions and action modifiers.

Table 10: Firewall Filter Actions and Action Modifiers

Action or Action Modifier	Description
Actions	
accept	Accept a packet. This is the default.
discard	Discard a packet silently, without sending an ICMP message.
reject <message-type>	Discard a packet, sending an ICMP destination unreachable message. 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 TCP reset is returned if the packet is a TCP packet. Otherwise, nothing is returned.
Action Modifiers	
alert	Log an alert for this packet. The log can be sent to a server for storage and analysis.
count counter-name	Increment a counter for this filter. 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 count into the same counter.
log	Log the packet’s header information in the Routing Engine. You can access this information from the CLI, but it is not available from network management.
output-queue (0..3)	Set the output queue bits in the packet header to a specified value between 0 and 3.
plp (0..1)	Set the packet loss priority (PLP) bit in the packet header to a specified value, 0 or 1.
sample	Sample the traffic on the interface. Use this modifier only when traffic sampling is enabled; for more information, see “Configure Traffic Sampling” on page 299.

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 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. If it 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, or, if there are no more terms, the packet is discarded.

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

Specify Numeric Range Filter Match Conditions on page 274

Specify Address Filter Match Conditions on page 276

Specify Bit-Field Filter Match Conditions on page 280

Specify 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 11 describes the numeric range filter match conditions.

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 65535 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 were not 25:

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

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

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

Table 11: Numeric Range Firewall Filter Match Conditions

Match Condition	Description
<i>keyword-except</i>	Negate a match. For example, <i>destination-port-except number</i> .
<i>destination-port number</i>	<p>TCP or UDP destination port field. You cannot specify 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 282.</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>
<i>dscp number</i>	<p>Differentiated Services codepoint. The Diffserv protocol uses the ToS byte in the IP header. The most significant six bits of this byte form the Diffserv codepoint (DSCP).</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>The Expedited Forwarding RFC defines one codepoint: ef (46).</p> <p>The Assured Forwarding RFC defines four classes, with three drop precedences in each class, for a total of twelve codepoints:</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>
<i>fragment-offset number</i>	Fragment offset field.
<i>icmp-code number</i>	<p>ICMP code field. This value or keyword provides more specific information than the <i>icmp-type</i>. Because the value's meaning depends upon the associated <i>icmp-type</i>, you must specify the <i>icmp-type</i> along with the <i>icmp-code</i>. For more information, see "How Firewall Filters Test a Packet's Protocol" on page 282.</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> <p>parameter-problem: ip-header-bad (0), required-option-missing (1)</p> <p>redirect: redirect-for-host (1), redirect-for-network (0), redirect-for-tos-and-host (3), redirect-for-tos-and-net (2)</p> <p>time-exceeded: ttl-eq-zero-during-reassembly (1), ttl-eq-zero-during-transit (0)</p> <p>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)</p>
<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 282.</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>

Match Condition	Description
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 "Apply Firewall Filters" on page 287.
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. 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 282. In place of the numeric value, you can specify one of the text synonyms listed under destination-port.
precedence <i>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): critical-ecp (0xa0), flash (0x60), flash-override (0x80), immediate (0x40), internet-control (0xc0), net-control (0xe0), priority (0x20), or routine (0x00).
protocol <i>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): egp (8), gre (47), icmp (1), igmp (2), ipip (4), ospf (89), pim (103), rsvp (46), tcp (6), or udp (17).
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 282. In place of the numeric field, you can specify one of the text synonyms listed under destination-port.

Specify 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 12 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:

```
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:

```
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 filter filter-name term term-name from]
user@host# set destination-address 10
[edit firewall 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 131.0.0.0/8 match, except for addresses that fall under 131.108.0.0/16. All other addresses implicitly do not match this condition.

```
destination-address {
    131.108.0.0/16 except;
    131.0.0.0/8;
}
```

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

```
destination-address {
    0.0.0.0/0;
    10.1.1.0/24 except;
}
```

Because the prefixes are order-independent and use longest-match rules, shorter prefixes subsume longer 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:

8.4.1.2 matches the 8.0.0.0/10 prefix, and thus the action in the *then* statement is taken.

8.2.1.2 matches the 8.2.0.0/16 prefix. Because this prefix is negated (that is, marked as *except*), an explicit mismatch occurs, and either the next term in the filter is evaluated, if there is one, or, if there are no more terms, the packet is discarded.

1.2.3.4 does not match anything 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, so it is considered to be a mismatch.

The 8.3.0.0/16 statement is ignored because it falls under the address 8.0.0.0/10—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.

```
source-address {
    8.0.0.0/10;
    8.2.0.0/16 except;
    192.168.1.0;
    192.168.1.192/26 except;
    192.168.1.254;
    8.3.0.0/16;           # ignored
    10.2.2.2 except;     # ignored
}
```

Table 12: Address Firewall Filter Match Conditions

Match Condition	Description
<i>keyword prefix except</i>	Negate an individual prefix. For example, <i>address prefix except</i> .
<i>address 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.
<i>destination-address prefix</i>	IP destination address field. You cannot specify the destination-address and address match conditions in the same term.
<i>source-address prefix</i>	IP source address field. You cannot specify the source-address and address match conditions in the same rule.

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:

```
from {
  source-prefix-list {
    prefix-list1;
    prefix-list2;
  }
  destination-prefix-list {
    prefix-list1;
  }
}
```

For further information, see the *JUNOS Internet Software Configuration Guide: Routing and Routing Protocols*.

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, if you apply the following terms within a firewall filter:

```
term 1 {
  from {
    address {
      1.0.0.0/12;
    }
    protocol tcp;
  }
  then {
    count If1-1;
    accept;
  }
}
term 2{
  from {
    address {
      1.26.0.0/15;
    }
    protocol tcp;
  }
  then {
    count If1-2;
    accept;
  }
}
```

A packet whose source address is 1.0.1.1 and whose destination address is 1.27.1.1 should increment the first counter, as should a packet with source address 1.27.1.1 and destination address 1.0.1.1. 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, for example:

```
term 1 {
    from {
        source-address 192.168/16;
    }
    then accept;
}
term 2 {
    from {
        destination-address 192.168/16;
    }
    then accept;
}
```

Specify Bit-Field Filter Match Conditions

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 13 describes the bit-field match conditions.

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 14. 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 either is 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 of the bits 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 13: Bit-Field Firewall Filter Match Conditions

Match Condition	Description
Conditions with Variables	
<code>fragment-flags <i>number</i></code>	IP fragmentation flags. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): <code>dont-fragment (0x4000)</code> , <code>more-fragments (0x2000)</code> , or <code>reserved (0x8000)</code> .
<code>ip-options <i>number</i></code>	IP options. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>loose-source-route (131)</code> , <code>record-route (7)</code> , <code>router-alert (148)</code> , <code>strict-source-route (137)</code> , or <code>timestamp (68)</code> .
<code>tcp-flags <i>number</i></code>	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 282. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): <code>ack (0x10)</code> , <code>fin (0x01)</code> , <code>push (0x08)</code> , <code>rst (0x04)</code> , <code>syn (0x02)</code> , or <code>urgent (0x20)</code> .
Text Synonyms	
<code>first-fragment</code>	First fragment of a fragmented packet. This condition does not match unfragmented packets.
<code>tcp-established</code>	TCP packets other than the first packet of a connection. This is a synonym for <code>"(act rst)"</code> . This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol <code>tcp</code> match condition.
<code>tcp-initial</code>	First TCP packet of a connection. This is a synonym for <code>"(syn & !ack)"</code> . This condition does not implicitly check that the protocol is TCP. To check this, specify the protocol <code>tcp</code> match condition.

Table 14: Bit-Field Logical Operators

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

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 firewall 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.

For example, if you specify a match of destination-port ssh, the firewall 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.

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.

There are cases in which not testing the protocol field is reasonable. In the following example, the first term matches of 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 {
  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: Define Firewall Filters

Block Telnet and 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]
firewall {
  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;
    }
  }
}
```

Block TFTP access, logging any attempts to establish TFTP connections:

```
[edit firewall]
firewall {
  filter tftp-access-control {
    from {
      protocol udp;
      port tftp;
    }
    then {
      log;
      discard;
    }
  }
}
```

The following filter does not block any traffic, but it does count individual IP option packets. It also logs packets that have loose or strict source routing.

```
[edit firewall]
filter ip-option-filter {
  term match-ssrr {
    from {
      ip-options strict-source-route;
    }
    then {
      count strict-source-route;
      log;
      accept;
    }
  }
  term match-lsrr {
    from {
      ip-options loose-source-route;
    }
    then {
      count loose-source-route;
      log;
      accept;
    }
  }
  term match-rr {
    from {
      ip-options record-route;
    }
    then {
      count record-route;
      accept;
    }
  }
  term match-ts {
    from {
      ip-options timestamp;
    }
    then {
      count timestamp;
      accept;
    }
  }
  term match-ra {
    from {
      ip-options router-alert;
    }
    then {
      count router-alert;
      accept;
    }
  }
  term match-all {
    then accept;
  }
}
```

Accept only OSPF packets from an address in the prefix 131.108.0.0/16, discarding all other packets with an administratively-prohibited ICMP message:

```
[edit firewall]
filter ospf-filter {
  term term1 {
    from {
      source-address {
        131.108.0.0/16;
      }
      protocol ospf;
    }
  }
  term default-term {
    then {
      reject administratively-prohibited;# this is the default reject action
    }
  }
}
```

Match packets that are either OSPF packets or packets that come from an address in the prefix 131.108/16. Send an administratively-prohibited ICMP message for all packets that do not match.

```
[edit firewall]
filter ospf-or-131{
  term protocol-match {
    from {
      protocol ospf;
    }
  }
  term address-match {
    from {
      source-address {
        131.108.0.0/16;
      }
    }
  }
}
```

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 {
  filter fire1 {
    term 1 {
      from {
        address {
          192.168.5.0/24 except;
          0.0.0.0/0;
        }
      }
      then {
        count rejpref1-1;
        log;
        reject;
      }
    }
    term 2 {
      then {
        count rejpref1-2;
        log;
        accept;
      }
    }
  }
}
```

Apply Firewall Filters

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

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

In the input statement, you 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, you 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 use the same filter one or more times.

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

Filters apply to all packets entering an interface, not just the packets destined to the Routing Engine. To filter packets destined to the Routing Engine, use the interface group keyword in “Define Interface Groups” on page 288.

For filters applied to transit traffic to function, the router must contain an Internet Processor II ASIC.

Define 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 11 on page 275.

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

```
[edit interfaces]
interfaces interface-name {
  unit logical-unit-number {
    family inet {
      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: Define Interface Filters

Create a filter that contains an interface group:

```
[edit firewall]
filter gnome {
  term g1 {
    from {
      interface-group 1;
      address {
        207.79.80.114/32;
      }
      protocol tcp;
      port finger;
    }
    then {
      count gnome-counter1;
      log;
      reject;
    }
  }
  term g2 {
    then {
      count gnome-counter2;
      log;
      accept;
    }
  }
}
```

Apply the filter that contains an interface group. You must apply the filter to a physical interface; it does not work on the loopback interface, lo0.

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

Configure Policing

Policing, or rate-limiting, enables you to limit the amount of traffic that passes into or out of an interface. Policing applies two types of rate limits on the traffic:

Bandwidth—The number of bits per second permitted, on average.

Maximum burst size—Bursts of data that exceed the given bandwidth limit, but only up to a total number of bytes given by this value.

Policing employs the *token-bucket algorithm*, which enforces a limit on average bandwidth while allowing bursts up to a specified maximum value. It offers more flexibility than the *leaky bucket algorithm* (see “Configure Receive and Transmit Leaky Bucket Properties” on page 36) in regulating bursty traffic and never discards packets.

You can define specific classes of traffic on an interface, to which you can apply a set of rate limits. To do this, you define *policers* within a filter statement. For example, suppose you want to limit all ftp traffic from a particular source to certain rate limits. The specification consists of the following:

1. You configure one or more policer statements in a section of the filter definition that precedes the term definitions.
2. Once they are defined, you reference the policers in the then clause of a term.
3. You can add other components of an action such as accept/discard, count, or log.
4. Finally, you must apply the policers to an interface for them to be effective. You do this in exactly the same way as for firewall filters.

The policer is applied to the packet first, and if the packet exceeds the defined limits, the actions of the then clause of the policer are applied. If the result of the policing action was not a discard, the remaining components of the then clause of the term are applied.

Rate-Limiting Statement

To specify the rate-limiting part of a policer, include an if-exceeding statement:

```
if-exceeding {
    bandwidth-limit rate;
    burst-size-limit bytes;
}
```

For example, you can define a bandwidth limit of 20 mbps with a burst size limit of 500 kb:

```
if-exceeding {
    bandwidth-limit 20m;
    burst-size-limit 500k;
}
```

You specify `bandwidth-limit` in bits per second, and `burst-size-limit` in bytes. You can specify the value as a complete decimal number or as a decimal number followed by the abbreviation k (1000), m (1,000,000), or g (1,000,000,000).

There is no absolute minimum value for `bandwidth-limit`, but any value below 61,040 bps will result in an effective rate of 30,520 bps. The maximum value is 4.29 Gbps.

The recommended minimum value for `burst-size-limit` is the Maximum Transmission Unit (MTU) of the IP packets being policed. The maximum value is 16.7 MB.

Policer Action Statement

If a packet does not exceed its rate limits, it will be processed further without being affected. If the packet exceeds its limits, it is handled in one of two ways, depending on what you specify:

The packet can be discarded.

The packet can be marked, by setting certain control bits. These bits are the packet loss priority (PLP) bit and the two queue bits. You can set either the PLP bit or the queue bits, or both.

To simply discard a packet that exceeds the rate limits:

```
then {
  discard;
}
```

To set the PLP bit equal to 1:

```
then {
  plp 1;
}
```

To set the queue bits to the value 2:

```
then {
  output-queue 2;
}
```

The range of values for plp is 0 or 1, and the range of values for output-queue is 0 through 3.

For both MPLS and IP traffic, if the least significant bit (LSB) of the precedence field is set, then the PLP bit is set by the incoming interface on downstream routers. This increases the probability of drop by RED on downstream routers. Therefore, whenever you explicitly set the PLP on a packet, you should also rewrite its precedence field to a value that has the LSB of the precedence field set as follows:

```
001
011
101
111
```

Conversely, with packets that should receive priority treatment, and any traffic that does not have its PLP bit set, you should rewrite the precedence field to values that do not contain a 1 in the LSB:

```
000
010
100
110
```

For MPLS packets, in addition to the LSB of the precedence field affecting downstream treatment, the first two bits identify the outbound transmission queue used on all downstream outbound interfaces. You can manipulate the MPLS CoS field through MPLS CoS configuration.

For more information, see “Rewrite the IP Precedence Bits” on page 334.

Combining the rate-limiting and action statements produces an example of a complete policer specification:

```
policer p1 {
  if-exceeding {
    bandwidth-limit 300k;
    burst-size-limit 500k;
  }
  then {
    discard;
  }
}
```

The general form is:

```
[edit firewall filter filter-name]
policer policer-name {
  if-exceeding {
    bandwidth-limit rate;
    burst-size-limit bytes;
  }
  then {
    policer-action
  }
}
```

Where *policer-action* is as defined above.

Example: Configure Policing

The following example shows a complete filter specification containing a policer. It instructs the router to limit all ftp traffic from a given source to certain rate limits. Traffic exceeding the limits is discarded; the remaining traffic is accepted and counted.

```
[edit firewall]
filter limit-ftp {
  policer p1{
    if-exceeding {
      bandwidth-limit 400k;
      burst-size-limit 100k;
    }
    then {
      discard;
    }
  }
  term t-ftp{
    from {
      source-address 1.2.3/24;
      protocol tcp;
      destination-port ftp;
    }
    then {
      policer p1;
      accept;
      count count-ftp;
    }
  }
}
```