JUNIPER NETWORKS

Junos® OS

Firewall Filters Feature Guide for Routing Devices

Release
13.2

Published: 2013-09-17

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS Firewall Filters Feature Guide for Routing Devices
13.2

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

List of Figures

List of Tables

About the Documentation

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at http://www.juniper.net/techpubs/.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at http://www.juniper.net/books.

Supported Platforms

For the features described in this document, the following platforms are supported:

- ACX Series
- M Series
- MX Series
- T Series
- PTX Series

Using the Examples in This Manual

If you want to use the examples in this manual, you can use the **load merge** or the **load merge relative** command. These commands cause the software to merge the incoming

configuration into the current candidate configuration. The example does not become active until you commit the candidate configuration.

If the example configuration contains the top level of the hierarchy (or multiple hierarchies), the example is a *full example*. In this case, use the **load merge** command.

If the example configuration does not start at the top level of the hierarchy, the example is a *snippet*. In this case, use the **load merge relative** command. These procedures are described in the following sections.

Merging a Full Example

To merge a full example, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration example into a text file, save the file with a name, and copy the file to a directory on your routing platform.

   For example, copy the following configuration to a file and name the file **ex-script.conf**. Copy the **ex-script.conf** file to the **/var/tmp** directory on your routing platform.

   ```
   system {
     scripts {
       commit {
         file ex-script.xsl;
       }
     }
   }
   interfaces {
     fxp0 {
       disable;
       unit 0 {
         family inet {
           address 10.0.0.1/24;
         }
       }
     }
   }
   ```

2. Merge the contents of the file into your routing platform configuration by issuing the **load merge** configuration mode command:

   ```
   [edit]
   user@host# load merge /var/tmp/ex-script.conf
   load complete
   ```

Merging a Snippet

To merge a snippet, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration snippet into a text file, save the file with a name, and copy the file to a directory on your routing platform.

   For example, copy the following snippet to a file and name the file **ex-script-snippet.conf**. Copy the **ex-script-snippet.conf** file to the **/var/tmp** directory on your routing platform.

```
commit {
  file ex-script-snippet.xsl; }
```

2. Move to the hierarchy level that is relevant for this snippet by issuing the following configuration mode command:

```
[edit]
user@host# edit system scripts
[edit system scripts]
```

3. Merge the contents of the file into your routing platform configuration by issuing the **load merge relative** configuration mode command:

```
[edit system scripts]
user@host# load merge relative /var/tmp/ex-script-snippet.conf
load complete
```

For more information about the **load** command, see the *CLI User Guide*.

Documentation Conventions

Table 1 on page xix defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Table 2 on page xix defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the**configure** command: user@host> **configure**
`Fixed-width text like this`	Represents output that appears on the terminal screen.	`user@host>` **show chassis alarms** `No alarms currently active`

Table 2: Text and Syntax Conventions *(continued)*

Convention	Description	Examples
Italic text like this	• Introduces or emphasizes important new terms. • Identifies guide names. • Identifies RFC and Internet draft titles.	• A policy *term* is a named structure that defines match conditions and actions. • *Junos OS CLI User Guide* • RFC 1997, *BGP Communities Attribute*
Italic text like this	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# **set system domain-name** *domain-name*
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	• To configure a stub area, include the **stub** statement at the **[edit protocols ospf area area-id]** hierarchy level. • The console port is labeled **CONSOLE**.
< > (angle brackets)	Enclose optional keywords or variables.	**stub <default-metric** ***metric*****>;**
| (pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	**broadcast | multicast** **(*string1* | *string2* | *string3*)**
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	**rsvp { # Required for dynamic MPLS only**
[ ] (square brackets)	Enclose a variable for which you can substitute one or more values.	**community name members [** ***community-ids*** **]**

Indention and braces ({ })	Identify a level in the configuration hierarchy.	[edit]
		routing-options {
		  static {
		  route default {
		  nexthop *address*;
		  retain;
		  }
		  }
		}

; (semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
GUI Conventions		
Bold text like this	Represents graphical user interface (GUI) items you click or select.	• In the Logical Interfaces box, select **All Interfaces**. • To cancel the configuration, click **Cancel**.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select **Protocols>Ospf**.

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to techpubs-comments@juniper.net, or fill out the documentation feedback form at https://www.juniper.net/cgi-bin/docbugreport/ . If you are using e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable)

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf.
- Product warranties—For product warranty information, visit http://www.juniper.net/support/warranty/.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: http://www.juniper.net/customers/support/
- Search for known bugs: http://www2.juniper.net/kb/
- Find product documentation: http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base: http://kb.juniper.net/
- Download the latest versions of software and review release notes: http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://www.juniper.net/alerts/

- Join and participate in the Juniper Networks Community Forum: http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Management tool: http://www.juniper.net/cm/

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://tools.juniper.net/SerialNumberEntitlementSearch/

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at http://www.juniper.net/cm/.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see http://www.juniper.net/support/requesting-support.html.

PART 1

Overview

CHAPTER 1

Introduction to Stateless Firewall Filters

Router Data Flow Overview

The Junos® operating system (Junos OS) provides a *policy framework*, which is a collection of Junos OS policies that enable you to control flows of routing information and packets within the router.

Flow of Routing Information

Routing information is the information about routes learned by the routing protocols from a router's neighbors. This information is stored in routing tables. The routing protocols advertise active routes only from the routing tables. An *active route* is a route that is chosen from all routes in the routing table to reach a destination.

To control which routes the routing protocols place in the routing tables and which routes the routing protocols advertise from the routing tables, you can configure *routing policies*, which are sets of rules that the policy framework uses to preempt default routing policies.

The Routing Engine, which is the router's control plane, handles the flow of routing information between the routing protocols and the routing tables and between the routing tables and the forwarding table. The Routing Engine runs the Junos OS and routing policies and stores the active router configuration, the master routing table, and the master forwarding table,

Flow of Data Packets

Data packets are chunks of data that transit the router as they are being forwarded from a source to a destination. When a router receives a data packet on an interface, it determines where to forward the packet by looking in the forwarding table for the best route to a destination. The router then forwards the data packet toward its destination through the appropriate interface.

The Packet Forwarding Engine, which is the router's forwarding plane, handles the flow of data packets in and out of the router's physical interfaces. Although the Packet Forwarding Engine contains Layer 3 and Layer 4 header information, it does not contain the packet data itself (the packet's payload).

Flow of Local Packets

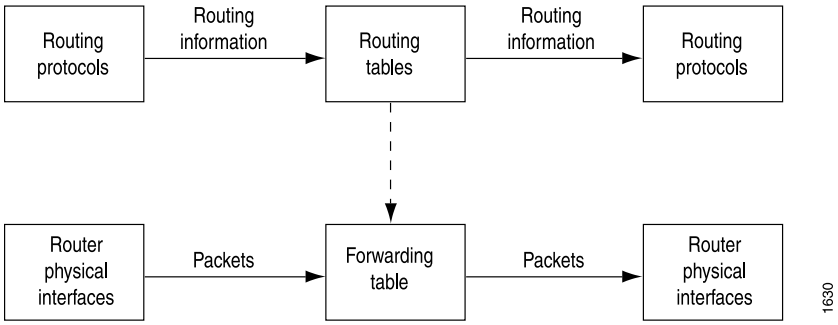
Local packets are chunks of data that are destined for or sent by the router. Local packets usually contain routing protocol data, data for IP services such as Telnet or SSH, and data for administrative protocols such as the Internet Control Message Protocol (ICMP). When the Routing Engine receives a local packet, it forwards the packet to the appropriate process or to the kernel, which are both part of the Routing Engine, or to the Packet Forwarding Engine.

The Routing Engine handles the flow of local packets from the router's physical interfaces and to the Routing Engine.

Interdependent Flows of Routing Information and Packets

Figure 1 on page 4 illustrates the flow of data through a router. Although routing information flows and packet flows are very different from one another, they are also interdependent.

Figure 1: Flows of Routing Information and Packets



Routing policies determine which routes the Routing Engine places in the forwarding table. The forwarding table, in turn, has an integral role in determining the appropriate physical interface through which to forward a packet.

Related Documentation

- Stateless Firewall Filter Overview on page 5
- *Packet Flow Within Routers Overview*
- *Understanding BGP Path Selection*

- *Understanding Route Preference Values*
- *Understanding Routing Policies*

Stateless Firewall Filter Overview

This topic covers the following information:

Packet Flow Control

To influence which packets are allowed to transit the system and to apply special actions to packets as necessary, you can configure *stateless firewall filters*. A stateless firewall specifies a sequence of one or more packet-filtering rules, called *filter terms*. A filter term specifies *match conditions* to use to determine a match and *actions* to take on a matched packet. A stateless firewall filter enables you to manipulate any packet of a particular protocol family, including fragmented packets, based on evaluation of Layer 3 and Layer 4 header fields. You typically apply a stateless firewall filter to one or more interfaces that have been configured with protocol family features. You can apply a stateless firewall filter to an ingress interface, an egress interface, or both.

Data Packet Flow Control

To control the flow of data packets transiting the device as the packets are being forwarded from a source to a destination, you can apply stateless firewall filters to the input or output of the router's or switch's physical interfaces.

To enforce a specified bandwidth and maximum burst size for traffic sent or received on an interface, you can configure *policers*. Policers are a specialized type of stateless firewall filter and a primary component of the Junos OS *class-of-service* (CoS).

Local Packet Flow Control

To control the flow of local packets between the physical interfaces and the Routing Engine, you can apply stateless firewall filters to the input or output of the *loopback interface*. The loopback interface (**lo0**) is the interface to the Routing Engine and carries no data packets.

Stateless and Stateful Firewall Filters

A stateless firewall filter, also known as an *access control list* (ACL), does not statefully inspect traffic. Instead, it evaluates packet contents statically and does not keep track of the state of network connections. In contrast, a *stateful firewall filter* uses connection state information derived from other applications and past communications in the data flow to make dynamic control decisions.

The *Junos OS Firewall Filters and Traffic Policers Library for Routing Devices* describes *stateless firewall filters* supported on T Series, M Series, MX Series routers and EX Series switches.

Purpose of Stateless Firewall Filters

The basic purpose of a stateless firewall filter is to enhance security through the use of packet filtering. Packet filtering enables you to inspect the components of incoming or outgoing packets and then perform the actions you specify on packets that match the criteria you specify. The typical use of a stateless firewall filter is to protect the Routing Engine processes and resources from malicious or untrusted packets.

Related Documentation

- Router Data Flow Overview on page 3
- Stateless Firewall Filter Types on page 6
- *Traffic Policing Overview*
- *Packet Flow Through the CoS Process Overview*

Stateless Firewall Filter Types

This topic covers the following information:

Firewall Filters

The Junos OS standard stateless firewall filters support a rich set of packet-matching criteria that you can use to match on specific traffic and perform specific actions, such as forwarding or dropping packets that match the criteria you specify. You can configure firewall filters to protect the local router or to protect another device that is either directly or indirectly connected to the local router. For example, you can use the filters to restrict the local packets that pass from the router's physical interfaces to the Routing Engine. Such filters are useful in protecting the IP services that run on the Routing Engine, such as Telnet, SSH, and BGP, from denial-of-service attacks.

NOTE: If you configured targeted broadcast for virtual routing and forwarding (VRF) by including the **forward-and-send-to-re** statement, any firewall filter that is configured on the Routing Engine loopback interface (**lo0**) cannot be applied to the targeted broadcast packets that are forwarded to the Routing Engine. This is because broadcast packets are forwarded as flood next hop traffic and not as local next hop traffic, and you can only apply a firewall filter to local next hop routes for traffic directed toward the Routing Engine.

Service Filters

A service filter defines packet-filtering (a set of match conditions and a set of actions) for IPv4 or IPv6 traffic. You can apply a service filter to the inbound or outbound traffic at an adaptive services interface to perform packet filtering on traffic before it is accepted for service processing. You can also apply a service filter to the traffic that is returning to the services interface after service processing to perform postservice processing.

Service filters filter IPv4 and IPv6 traffic only and can be applied to logical interfaces on Adaptive Services PICs, MultiServices PICs, and MultiServices DPCs only. Service filters are not supported on J Series devices and Branch SRX devices.

Simple Filters

Simple filters are supported on Gigabit Ethernet intelligent queuing (IQ2) and Enhanced Queuing Dense Port Concentrator (EQ DPC) interfaces only. Unlike standard filters, simple filters support IPv4 traffic only and have a number of restrictions. For example, you cannot configure a terminating action for a simple filter. Simple filters always accept packets. Also, simple filters can be applied only as input filters. They are not supported on outbound traffic. Simple filters are recommended for metropolitan Ethernet applications.

Related Documentation

- Stateless Firewall Filter Overview on page 5
- Stateless Firewall Filter Components on page 7

Stateless Firewall Filter Components

This topic covers the following information:

Protocol Family

Under the **firewall** statement, you can specify the protocol family for which you want to filter traffic.

Table 3 on page 7 describes the firewall filter protocol families.

Table 3: Firewall Filter Protocol Families

Type of Traffic to Be Filtered	Configuration Statement	Comments
Protocol Independent	**family any**	All protocol families configured on a logical interface.

Table 3: Firewall Filter Protocol Families *(continued)*

Type of Traffic to Be Filtered	Configuration Statement	Comments
Internet Protocol version 4 (IPv4)	**family inet**	The **family inet** statement is optional for IPv4.
Internet Protocol version 6 (IPv6)	**family inet6**	
MPLS	**family mpls**	
MPLS-tagged IPv4	**family mpls**	Supports matching on IP addresses and ports, up to five MPLS stacked labels.
MPLS-tagged IPv6	**family mpls**	Supports matching on IP addresses and ports, up to five MPLS stacked labels.
Virtual private LAN service (VPLS)	**family vpls**	
Layer 2 Circuit Cross-Connection	**family ccc**	
Layer 2 Bridging	**family bridge** (for MX Series routers) and **family ethernet-switching** (for EX Series switches)	MX Series routers and EX Series switches only.

Filter Type

Under the **family *family-name*** statement, you can specify the type and name of the filter you want to configure.

Table 4 on page 8 describes the firewall filter types.

Table 4: Filter Types

Filter Type	Configuration Statement	Description
Standard Firewall Filter	**filter *filter-name***	Filters the following traffic types: • Protocol independent • IPv4 • IPv6 • MPLS • MPLS-tagged IPv4 • MPLS-tagged IPv6 • VPLS • Layer 2 CCC • Layer 2 bridging (MX Series routers and EX Series switches only)

Table 4: Filter Types *(continued)*

Filter Type	Configuration Statement	Description
Service Filter	**service-filter** *service-filter-name*	Defines packet-filtering to be applied to ingress or egress before it is accepted for service processing or applied to returning service traffic after service processing has completed. Filters the following traffic types: • IPv4 • IPv6 Supported at logical interfaces configured on the following hardware only: • Adaptive Services (AS) PICs on M Series and T Series routers • Multiservices (MS) PICs on M Series and T Series routers • Multiservices (MS) DPCs on MX Series routers (and EX Series switches)
Simple Filter	**simple-filter** *simple-filter-name*	Defines packet filtering to be applied to ingress traffic only. Filters the following traffic type: • IPv4 Supported at logical interfaces configured on the following hardware only: • Gigabit Ethernet Intelligent Queuing (IQ2) PICs installed on M120, M320, or T Series routers • Enhanced Queuing Dense Port Concentrators (EQ DPCs) installed on MX Series routers (and EX Series switches)

Terms

Under the **filter**, **service-filter**, or **simple-filter** statement, you must configure at least one firewall filter *term*. A term is a named structure in which match conditions and actions are defined. Within a firewall filter, you must configure a unique name for each term.

TIP: For each protocol family on an interface, you can apply no more than one filter in each direction. If you try to apply additional filters for the same protocol family in the same direction, the last filter overwrites the previous filter. You can, however, apply filters from the same protocol family to the input and output direction of the same interface.

All stateless firewall filters contain one or more terms, and each term consists of two components—match conditions and actions. The match conditions define the values or fields that the packet must contain to be considered a match. If a packet is a match, the corresponding action is taken. By default, a packet that does not match a firewall filter is discarded.

If a packet arrives on an interface for which no firewall filter is applied for the incoming traffic on that interface, the packet is accepted by default.

NOTE: A firewall filter with a large number of terms can adversely affect both the configuration commit time and the performance of the Routing Engine.

Additionally, you can configure a stateless firewall filter within the term of another filter. This method enables you to add common terms to multiple filters without having to modify all filter definitions. You can configure one filter with the desired common terms, and configure this filter as a term in other filters. Consequently, to make a change in these common terms, you need to modify only one filter that contains the common terms, instead of multiple filters.

Match Conditions

A firewall filter term must contain at least one packet-filtering criteria, called a *match condition*, to specify the field or value that a packet must contain in order to be considered a match for the firewall filter term. For a match to occur, the packet must match all the conditions in the term. If a packet matches a firewall filter term, the router (or switch) takes the configured action on the packet.

If a firewall filter term contains multiple match conditions, a packet must meet *all* match conditions to be considered a match for the firewall filter term.

If a single match condition is configured with multiple values, such as a range of values, a packet must match only *one* of the values to be considered a match for the firewall filter term.

The scope of match conditions you can specify in a firewall filter term depends on the protocol family under which the firewall filter is configured. You can define various match conditions, including the IP source address field, IP destination address field, TCP or UDP source port field, IP protocol field, Internet Control Message Protocol (ICMP) packet type, IP options, TCP flags, incoming logical or physical interface, and outgoing logical or physical interface.

Each protocol family supports a different set of match conditions, and some match conditions are supported only on certain routing devices. For example, a number of match conditions for VPLS traffic are supported only on the MX Series 3D Universal Edge Routers.

In the **from** statement in a firewall filter term, you specify characteristics that the packet must have for the action in the subsequent **then** statement to be performed. The characteristics are referred to as *match conditions*. The packet must match all conditions in the **from** statement for the action to be performed, which also means that the order of the conditions in the **from** statement is not important.

If an individual match condition can specify a list of values (such as multiple source and destination addresses) or a range of numeric values, a match occurs if any of the values matches the packet.

If a filter term does not specify match conditions, the term accepts all packets and the actions specified in the term's **then** statement are optional.

NOTE:

Some of the numeric range and bit-field match conditions allow you to specify a text synonym. For a complete list of synonyms:

- If you are using the J-Web interface, select the synonym from the appropriate list.
- If you are using the CLI, type a question mark (?) after the **from** statement.

Actions

The actions specified in a firewall filter term define the actions to take for any packet that matches the conditions specified in the term.

Actions that are configured within a single term are all taken on traffic that matches the conditions configured.

BEST PRACTICE: We strongly recommend that you explicitly configure one or more actions per firewall filter term. Any packet that matches all the conditions of the term is automatically accepted unless the term specifies other or additional actions.

Firewall filter actions fall into the following categories:

Filter-Terminating Actions

A filter-terminating action halts all evaluation of a firewall filter for a specific packet. The router (or switch) performs the specified action, and no additional terms are examined.

Nonterminating Actions

Nonterminating actions are used to perform other functions on a packet, such as incrementing a counter, logging information about the packet header, sampling the packet data, or sending information to a remote host using the system log functionality.

The presence of a nonterminating action, such as **count**, **log**, or **syslog**, without an explicit terminating action, such as **accept**, **discard**, or **reject**, results in a default terminating action of **accept**. If you do not want the firewall filter action to terminate, use the **next term** action after the nonterminating action.

In this example, term 2 is never evaluated, because term 1 has the implicit default **accept** terminating action.

```
[edit firewall filter test]
term 1 {
  from {
    source-address {
      0.0.0.0/0;
    }
  }
```

```
    then {
      log;
      <accept> #By default if not specified
    }
  }
  term 2 {
    then {
      reject;
    }
  }
```

In this example, term 2 is evaluated, because term 1 has the explicit **next term** flow control action.

```
[edit firewall filter test]
term 1 {
  from {
    source-address {
      0.0.0.0/0;
    }
  }
  then {
    log;
    next term;
  }
}
term 2 {
  then {
    reject;
  }
}
```

Flow Control Action

For standard stateless firewall filters only, the action **next term** enables the router (or switch) to perform configured actions on the packet and then evaluate the following term in the filter, rather than terminating the filter.

A maximum of 1024 **next term** actions are supported per standard stateless firewall filter configuration. If you configure a standard filter that exceeds this limit, your candidate configuration results in a commit error.

Related Documentation

- Stateless Firewall Filter Types on page 6
- *"Inserting a New Identifier in a Junos Configuration"* in the *CLI User Guide*

Stateless Firewall Filter Application Points

After you define the firewall filter, you must apply it to an application point. These application points include logical interfaces, physical interfaces, routing interfaces, and routing instances.

In most cases, you can apply a firewall filter as an *input* filter or an *output* filter, or both at the same time. Input filters take action on packets being received on the specified

interface, whereas output filters take action on packets that are transmitted through the specified interface.

You typically apply one filter with multiple terms to a single logical interface, to incoming traffic, outbound traffic, or both. However, there are times when you might want to chain together multiple firewall filters (with single or multiple terms) and apply them to an interface. You use an *input list* to apply multiple firewall filters to the incoming traffic on an interface. You use an *output list* to apply multiple firewall filters to the outbound traffic on an interface. You can include up to 16 filters in an input list or an output list.

There is no limit to the number of filters and counters you can set, but there are some practical considerations. More counters require more terms, and a large number of terms can take a long time to process during a commit operation. However, filters with more than 4000 terms and counters have been implemented successfully.

Table 5 on page 13 describes each point to which you can apply a firewall filter. For each application point, the table describes the types of firewall filters supported at that point, the router (or switch) hierarchy level at which the filter can be applied, and any platform-specific limitations.

Table 5: Stateless Firewall Filter Configuration and Application Summary

Filter Type	Application Point	Restrictions

Stateless firewall filter
 	
Configure by including the **filter** *filter-name* statement the **[edit firewall]** hierarchy level:
 	
filter *filter-name*;
 	
NOTE: If you do not include the **family** statement, the firewall filter processes IPv4 traffic by default.
	Logical interface
	
	Apply at the **[edit interfaces** *interface-name* **unit** *unit-number* **family inet]** hierarchy level by including the **input** *filter-name* or **output** *filter-name* statements:
	
	filter {
	input *filter-name*;
	output *filter-name*;
	}
	
	NOTE: A filter configured with the implicit **inet** protocol family cannot be included in an input filter list or an output filter list.
	
	NOTE: On T4000 Type 5 FPCs, a filter attached at the Layer 2 application point (that is, at the logical interface level) is unable to match with the forwarding class of a packet that is set by a Layer 3 classifier such as DSCP, DSCP V6, **inet-precedence**, and **mpls-exp**.
		Supported on the following routers:
		
		• T Series routers
		• M320 routers
		• M7i routers with the enhanced CFEB (CFEB-e)
		• M10i routers with the enhanced CFEB-e
		
		Also supported on the following Modular Port Concentrators (MPCs) on MX Series routers:
		
		• 10-Gigabit Ethernet MPC
		• 60-Gigabit Ethernet Queuing MPC
		• 60-Gigabit Ethernet Enhanced Queuing MPC
		• 100-Gigabit Ethernet MPC
		
		• Also supported on EX Series switches

Table 5: Stateless Firewall Filter Configuration and Application Summary *(continued)*

Filter Type	Application Point	Restrictions

Stateless firewall filter
 	
Configure at the **[edit firewall family *family-name*]** hierarchy level by including the following statement:
 	
filter *filter-name*;
 	
The ***family-name*** can be any of the following protocol families:
 	
• **any**
• **bridge**
• **ethernet-switching**
• **ccc**
• **inet**
• **inet6**
• **mpls**
• **vpls**
	Protocol family on a logical interface
	
	Apply at the **[edit interfaces *interface-name* unit *unit-number* family *family-name*]** hierarchy level by, including the **input**, **input-list**, **output**, or **output-list** statements:
	
	filter {
	input *filter-name*;
	input-list [*filter-names*];
	output *filter-name*;
	output-list [*filter-names*];
	}
		The protocol family **bridge** is supported only on MX Series routers.

Stateless firewall filter	**Routing Engine loopback interface**	

Service filter
 	
Configure at the **[edit firewall family (inet | inet6)]** hierarchy level by including the following statement:
 	
service-filter *service-filter-name*;
	Family inet or inet6 on a logical interface
	
	Apply at the **[edit interfaces *interface-name* unit *unit-number* family (inet | inet6)]** hierarchy level by using the **service-set** statement to apply a service filter as an input or output filter to a service set:
	
	service {
	input {
	service-set *service-set-name*
	service-filter *filter-name;*
	}
	output {
	service-set *service-set-name*
	service-filter *filter-name;*
	}
	}
	
	Configure a service set at the **[edit services]** hierarchy level by including the following statement:
	
	service-set *service-set-name*;
		Supported only on Adaptive Services (AS) and Multiservices (MS) PICs.

Table 5: Stateless Firewall Filter Configuration and Application Summary *(continued)*

Filter Type	Application Point	Restrictions

Postservice filter
 	
Configure at the **[edit firewall family (inet | inet6)]** hierarchy level by including the following statement:
 	
service-filter *service-filter-name*;
	Family inet or inet6 on a logical interface
	
	Apply at the **[edit interfaces** *interface-name* **unit** *unit-number* **family (inet | inet6)]** hierarchy level by including the **post-service-filter** statement to apply a service filter as an input filter:
	
	service {
	  input {
	  post-service-filter *filter-name*;
	  }
	}
		A postservice filter is applied to traffic returning to the services interface after service processing. The filter is applied only if a service set is configured and selected.

Simple filter Configure at the **[edit firewall family inet]** hierarchy level by including the following statement: simple-filter *filter-name*	**Family inet on a logical interface** Apply at the **[edit interfaces** *interface-name* **unit** *unit-number* **family inet]** hierarchy level by including the following statement: simple-filter *simple-filter-name*;	Simple filters can only be applied as input filters. Supported on the following platforms only: • Gigabit Ethernet intelligent queuing (IQ2) PICs on the M120, M320, and T Series routers. • Enhanced Queuing Dense Port Concentrators (EQ DPC) on MX Series routers (and EX Series switches).

Reverse packet forwarding (RPF) check filter
 	
Configured at the **[edit firewall family (inet | inet6)]** hierarchy level by including the following statement:
 	
filter *filter-name*;
	Family inet or inet6 on a logical interface
	
	Apply at the **[edit interfaces** *interface-name* **unit** *unit-number* **family (inet | inet6)]** hierarchy level by including the following statement:
	
	rpf-check fail-filter *filter-name*
	
	to apply the stateless firewall filter as an RPF check filter.
	
	rpf-check {
	  fail-filter *filter-name*;
	  mode loose;
	}
		Supported on MX Series routers and EX Series switches only.

Related Documentation

- Stateless Firewall Filter Components on page 7
- Supported Standards for Filtering on page 317

CHAPTER 2

Understanding Firewall Filters

Firewall Filters Overview

Firewall filters provide a means of protecting your router (and switch) from excessive traffic transiting the router (and switch) to a network destination or destined for the Routing Engine. Firewall filters that control local packets can also protect your router (and switch) from external incidents.

You can configure a firewall filter to do the following:

- Restrict traffic destined for the Routing Engine based on its source, protocol, and application.
- Limit the traffic rate of packets destined for the Routing Engine to protect against flood, or denial-of-service (DoS) attacks.
- Address special circumstances associated with fragmented packets destined for the Routing Engine. Because the device evaluates every packet against a firewall filter (including fragments), you must configure the filter to accommodate fragments that do not contain packet header information. Otherwise, the filter discards all but the first fragment of a fragmented packet.

Related Documentation

- Stateless Firewall Filter Types on page 6
- Guidelines for Configuring Firewall Filters on page 21
- Guidelines for Applying Firewall Filters on page 25
- Understanding How to Use Firewall Filters on page 29

How Firewall Filters Evaluate Packets

This topic covers the following information:

Firewall Filter Packet Evaluation Overview

The following sequence describes how the device evaluates a packet entering or exiting an interface if the input or output traffic at a device interface is associated with a firewall filter. Packet evaluation proceeds as follows:

1. The device evaluates the packet against the terms in the firewall filter sequentially, beginning with the first term in the filter.

   - If the packet matches all the conditions specified in a term, the device performs all the actions specified in that term.
   - If the packet does not match all the conditions specified in a term, the device proceeds to the next term in the filter (if a subsequent term exists) and evaluates the packet against that term.
   - If the packet does not match any term in the firewall filter, the device implicitly discards the packet.

2. Unlike service filters and simple filters, firewall filters support the **next term** action, which is neither a terminating action nor a nonterminating action but a flow control action.

   - If the matched term includes the **next term** action, the device continues evaluation of the packet at the next term within the firewall filter.
   - If the matched term does not include the **next term** action, evaluation of the packet against the given firewall filter ends at this term. The device does not evaluate the packet against any subsequent terms in this filter.

   A maximum of 1024 **next term** actions are supported per firewall filter configuration. If you configure a firewall filter that exceeds this limit, your candidate configuration results in a commit error.

3. The device stops evaluating a packet against a given firewall filter when either the packet matches a term without the **next term** action or the packet fails to match the last term in the firewall filter.

Packet Evaluation at a Single Firewall Filter

Table 6 on page 19 describes packet-filtering behaviors at a device interface associated with a single firewall filter.

Table 6: Packet Evaluation at a Single Firewall Filter

Firewall Filter Event	Action	Subsequent Action
The firewall filter term does not specify any match conditions.	The term matches all packets by default, and so the device performs the actions specified by that term.	If the term actions include the **next term** action, the device continues evaluation of the packet against the next term within the firewall filter (if a subsequent term exists).
The packet matches all conditions specified by the firewall filter term.	The device performs the actions specified by that term.	If the term actions include the **next term** action, the device continues evaluation of the packet against the next term within the firewall filter (if a subsequent term exists).
The packet matches all conditions specified by the firewall filter term, but the term does not specify any actions.	The device implicitly accepts the packet.	If the term actions include the **next term** action, the device continues evaluation of the packet against the next term within the firewall filter (if a subsequent term exists).
The packet does not match all conditions specified by the firewall filter term.	The device does not perform the actions specified by that term.	The device continues evaluation of the packet against the next term within the filter (if a subsequent term exists).

The packet does not match any term in the filter	The device implicitly discards the packet
	
	Every firewall filter configuration includes an implicit **discard** action at the end of the filter. This implicit terminating action is equivalent to including the following example term **t_explicit_discard** as the final term in the firewall filter:
	
	`term t_explicit_discard {`
	`  then discard;`
	`}`	

Best Practice: Explicitly Accept Any Traffic That Is Not Specifically Discarded

You might want a firewall filter to accept any traffic that the filter does not specifically discard. In this case, we recommend that you configure the firewall filter with a final term that specifies the **accept** terminating action.

In the following example snippet, configuring the **t_allow_all_else** term as the final term in the firewall filter explicitly configures the firewall filter to accept any traffic that the filter did not specifically discard :

```
term t_allow_all_else {
  then accept;
}
```

Following this best practice can simplify troubleshooting of the firewall filter.

Best Practice: Explicitly Reject Any Traffic That Is Not Specifically Accepted

On the other hand, you might want a firewall filter to reject any traffic that the firewall filter does not specifically accept. In this case, we recommend that you configure the firewall filter with a final term that specifies the **reject** terminating action.

In the following example snippet, configuring the **t_deny_all_else** term as the final term in the firewall filter explicitly configures the firewall filter to reject any traffic that the filter did not specifically accept:

```
term t_deny_all_else {
   then reject;
}
```

Following this best practice can simplify troubleshooting of the firewall filter.

Multiple Firewall Filters Attached to a Single Interface

On supported device interfaces, you can attach multiple firewall filters to a single interface. For more information, see "Understanding Multiple Firewall Filters Applied as a List" on page 54.

NOTE: On supported interfaces, you can attach a protocol-independent (**family any**) firewall filter and a protocol-specific (**family inet** or **family inet6**) firewall filter to the same interface. The protocol-independent firewall filter executes first. For more information, see "Guidelines for Applying Firewall Filters" on page 25.

Single Firewall Filter Attached to Multiple Interfaces

On supported interfaces, you can associate a single firewall filter with multiple interfaces, and Junos OS creates an *interface-specific instance* of that firewall filter for each associated interface.

- Junos OS associates each interface-specific instantiation of a firewall filter with a system-generated, interface-specific name.
- For any **count** actions in the filter terms, the Packet Forwarding Engine maintains separate, interface-specific counters, and Junos OS associates each counter with a system-generated, interface-specific name.
- For any **policer** actions in the filter terms, Junos OS creates separate, interface-specific instances of the policer actions.

For more information, see "Interface-Specific Firewall Filter Instances Overview" on page 61.

Related Documentation

- Firewall Filter Match Conditions for Protocol-Independent Traffic on page 339
- Firewall Filter Terminating Actions on page 384
- Firewall Filters Overview on page 17

Guidelines for Configuring Firewall Filters

This topic covers the following information:

Statement Hierarchy for Configuring Firewall Filters

To configure a firewall filter, you can include the following statements. For an IPv4 firewall filter, the **family inet** statement is optional.

```
firewall {
    family family-name {
        filter filter-name {
            accounting-profile name;
            interface-specific;
            physical-interface-filter;
            term term-name {
                filter filter-name;
            }
            term term-name {
                from {
                    match-conditions;
                    ip-version ipv4 {
                        match-conditions;
                        protocol (tcp | udp) {
                            match conditions;
                        }
                    }
                }
                then {
                    actions;
                }
            }
        }
    }
}
```

You can include the firewall configuration at one of the following hierarchy levels:

- **[edit]**

- **[edit logical-systems *logical-system-name*]**

NOTE: For stateless firewall filtering, you must allow the output tunnel traffic through the firewall filter applied to input traffic on the interface that is the next-hop interface toward the tunnel destination. The firewall filter affects only the packets exiting the router (or switch) by way of the tunnel.

Firewall Filter Protocol Families

A firewall filter configuration is specific to a particular protocol family. Under the **firewall** statement, include one of the following statements to specify the protocol family for which you want to filter traffic:

- **family any**—To filter protocol-independent traffic.
- **family inet**—To filter Internet Protocol version 4 (IPv4) traffic.
- **family inet6**—To filter Internet Protocol version 6 (IPv6) traffic.
- **family mpls**—To filter MPLS traffic.
- **family vpls**—To filter virtual private LAN service (VPLS) traffic.
- **family ccc**—To filter Layer 2 circuit cross-connection (CCC) traffic.
- **family bridge**—To filter Layer 2 bridging traffic for MX Series 3D Universal Edge Routers only.
- **family ethernet-switching**—To filter Layer 2 (Ethernet) traffic.

The **family *family-name*** statement is required only to specify a protocol family other than IPv4. To configure an IPv4 firewall filter, you can configure the filter at the **[edit firewall]** hierarchy level without including the **family inet** statement, because the **[edit firewall]** and **[edit firewall family inet]** hierarchy levels are equivalent.

Firewall Filter Names and Options

Under the **family *family-name*** statement, you can include **filter *filter-name*** statements to create and name firewall filters. The filter name can contain letters, numbers, and hyphens (-) and be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

At the **[edit firewall family *family-name* filter *filter-name*]** hierarchy level, the following statements are optional:

- **accounting-profile**
- **interface-specific**
- **physical-interface-filter**

Firewall Filter Terms

Under the **filter *filter-name*** statement, you can include **term *term-name*** statements to create and name filter terms.

- You must configure at least one term in a firewall filter.
- You must specify a unique name for each term within a firewall filter. The term name can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).
- The order in which you specify terms within a firewall filter configuration is important. Firewall filter terms are evaluated in the order in which they are configured. By default, new terms are always added to the end of the existing filter. You can use the **insert** configuration mode command to reorder the terms of a firewall filter.

At the **[edit firewall family *family-name* filter *filter-name* term *term-name*]** hierarchy level, the **filter *filter-name*** statement is not valid in the same term as **from** or **then** statements. When included at this hierarchy level, the **filter *filter-name*** statement is used to *nest* firewall filters.

Firewall Filter Match Conditions

Firewall filter match conditions are specific to the type of traffic being filtered.

With the exception of MPLS-tagged IPv4 or IPv6 traffic, you specify the term’s match conditions under the **from** statement. For MPLS-tagged IPv4 traffic, you specify the term’s IPv4 address-specific match conditions under the **ip-version ipv4** statement and the term’s IPv4 port-specific match conditions under the **protocol (tcp | udp)** statement.

For MPLS-tagged IPv6 traffic, you specify the term’s IPv6 address-specific match conditions under the **ip-version ipv6** statement and the term’s IPv6 port-specific match conditions under the **protocol (tcp | udp)** statement.

Table 7 on page 23 describes the types of traffic for which you can configure firewall filters.

Table 7: Firewall Filter Match Conditions by Protocol Family

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
Protocol-independent	**[edit firewall family any filter *filter-name* term *term-name*]** For the complete list of match conditions, see “Firewall Filter Match Conditions for Protocol-Independent Traffic” on page 339.
IPv4	**[edit firewall family inet filter *filter-name* term *term-name*]** For the complete list of match conditions, see “Firewall Filter Match Conditions for IPv4 Traffic” on page 341.

Table 7: Firewall Filter Match Conditions by Protocol Family *(continued)*

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
IPv6	**[edit firewall family inet6 filter *filter-name* term *term-name*]** For the complete list of match conditions, see "Standard Firewall Filter Match Conditions for IPv6 Traffic" on page 350.
MPLS	**[edit firewall family mpls filter *filter-name* term *term-name*]** For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS Traffic" on page 358.
IPv4 addresses in MPLS flows	**[edit firewall family mpls filter *filter-name* term *term-name* ip-version ipv4]** For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 360.
IPv4 ports in MPLS flows	**[edit firewall family mpls filter *filter-name* term *term-name* ip-version ipv4 protocol (tcp | udp)]** For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 360.
IPv6 addresses in MPLS flows	**[edit firewall family mpls filter *filter-name* term *term-name* ip-version ipv6]** For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 360.
IPv6 ports in MPLS flows	**[edit firewall family mpls filter *filter-name* term *term-name* ip-version ipv6 protocol (tcp | udp)]** For the complete list of match conditions, see "Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic" on page 360.
VPLS	**[edit firewall family vpls filter *filter-name* term *term-name*]** For the complete list of match conditions, see "Firewall Filter Match Conditions for VPLS Traffic" on page 362.
Layer 2 CCC	**[edit firewall family ccc filter *filter-name* term *term-name*]** For the complete list of match conditions, see "Firewall Filter Match Conditions for Layer 2 CCC Traffic" on page 369.
Layer 2 Bridging (MX Series routers and EX Series switches only)	**[edit firewall family bridge filter *filter-name* term *term-name*]** **[edit firewall family ethernet-switching filter *filter-name* term *term-name*]** (for EX Series switches only) For the complete list of match conditions, see "Firewall Filter Match Conditions for Layer 2 Bridging Traffic" on page 372.

If you specify an IPv6 address in a match condition (the **address**, **destination-address**, or **source-address** match conditions), use the syntax for text representations described in RFC 2373, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see *IPv6 Overview* and *Supported IPv6 Standards*.

Firewall Filter Actions

Under the **then** statement for a firewall filter term, you can specify the actions to be taken on a packet that matches the term.

Table 8 on page 25 summarizes the types of actions you can specify in a firewall filter term.

Table 8: Firewall Filter Action Categories

Type of Action	Description	Comment
Terminating	Halts all evaluation of a firewall filter for a specific packet. The router (or switch) performs the specified action, and no additional terms are used to examine the packet. You can specify only one *terminating action* in a firewall filter. You can, however, specify one terminating action with one or more *nonterminating actions* in a single term. For example, within a term, you can specify **accept** with **count** and **syslog**.	See "Firewall Filter Terminating Actions" on page 384.
Nonterminating	Performs other functions on a packet (such as incrementing a counter, logging information about the packet header, sampling the packet data, or sending information to a remote host using the system log functionality), but any additional terms are used to examine the packet.	See "Firewall Filter Nonterminating Actions" on page 377.
Flow control	For standard firewall filters only, the **next term** action directs the router (or switch) to perform configured actions on the packet and then, rather than terminate the filter, use the next term in the filter to evaluate the packet. If the **next term** action is included, the matching packet is evaluated against the next term in the firewall filter. Otherwise, the matching packet is not evaluated against subsequent terms in the firewall filter. For example, when you configure a term with the nonterminating action **count**, the term's action changes from an implicit **discard** to an implicit **accept**. The **next term** action forces the continued evaluation of the firewall filter.	You cannot configure the **next term** action with a terminating action in the same filter term. However, you can configure the next term action with another nonterminating action in the same filter term. A maximum of 1024 **next term** actions are supported per standard firewall filter configuration. If you configure a standard firewall filter that exceeds this limit, your candidate configuration results in a commit error.

Related Documentation

- Guidelines for Applying Firewall Filters on page 25
- Understanding How to Use Firewall Filters on page 29

Guidelines for Applying Firewall Filters

This topic covers the following information:

- Applying Firewall Filters Overview on page 26
- Statement Hierarchy for Applying Firewall Filters on page 26
- Restrictions on Applying Firewall Filters on page 27

Applying Firewall Filters Overview

You can apply a standard firewall filter to a loopback interface on the router or to a physical or logical interface on the router. Table 9 on page 26 summarizes the behavior of firewall filters based on the point to which you attach the filter.

Table 9: Firewall Filter Behavior by Filter Attachment Point

Filter Attachment Point	Filter Behavior
Loopback interface	The router's loopback interface, **lo0**, is the interface to the Routing Engine and carries no data packets. When you apply a firewall filter to the loopback interface, the filter evaluates the local packets received or transmitted by the Routing Engine.
Physical interface or logical interface	When you apply a filter to a physical interface on the router or to a logical interface (or member of an aggregated Ethernet bundle defined on the interface), the filter evaluates all data packet that pass through that interface.
Multiple interfaces	You can use the same firewall filter one or more times. On M Series routers, except the M120 and M320 routers, if you apply a firewall filter to multiple interfaces, the filter acts on the sum of traffic entering or exiting those interfaces. On T Series, M120, M320, and MX Series routers, interfaces are distributed among multiple packet-forwarding components. On these routers, you can configure firewall filters and service filters that, when applied to multiple interfaces, act on the individual traffic streams entering or exiting each interface, regardless of the sum of traffic on the multiple interfaces. For more information, see "Interface-Specific Firewall Filter Instances Overview" on page 61.
Single interface with protocol-independent and protocol-specific firewall filters attached	For interfaces hosted on the following hardware only, you can attach a protocol-independent (**family any**) firewall filter and a protocol-specific (**family inet** or **family inet6**) firewall filter simultaneously. The protocol-independent firewall filter executes first. • ACX Series Universal Access Routers • Flexible PIC Concentrators (FPCs) in M7i and M10i Multiservice Edge Routers • Modular Interface Cards (MICs) and Modular Port Concentrators (MPCs) in MX Series 3D Universal Edge Routers • T Series Core Routers NOTE: Interfaces hosted on the following hardware do not support protocol-independent firewall filters: • Forwarding Engine Boards (FEBs) in M120 routers • Enhanced III FPCs in M320 routers • FPC2 and FPC3 modules in MX Series routers • Dense Port Concentrators (DPCs) in MX Series routers • PTX Series Packet Transport Routers For more information, see "Understanding Multiple Firewall Filters Applied as a List" on page 54.

Statement Hierarchy for Applying Firewall Filters

To apply a firewall filter to a logical interface, configure the filter statement for the logical interface defined under either the **[edit]** or **[edit logical-systems *logical-system-name*]**

hierarchy level. Under the **filter** statement, you can include one or more of the following statements: **group** *group-number*, **input** *filter-name*, **input-list** *filter-name*, **output** *filter-name*, or **output-list** *filter-name*. The hierarchy level at which you attach the **filter** statement depends on the filter type and device type you are configuring.

Protocol-Independent Firewall Filters on MX Series Routers

To apply a protocol-independent firewall filter to a logical interface on an MX Series router, configure the **filter** statement *directly* under the logical unit:

```
interfaces {
  interface-name {
    unit logical-unit-number {
      filter {
        group group-number;
        input filter-name;
        input-list [ filter-names ];
        output filter-name;
        output-list [ filter-names ];
      }
    }
  }
}
```

All Other Firewall Filters on Logical Interfaces

To apply a firewall filter to a logical interface for all cases *other than* a protocol-independent filter on an MX Series router, configure the **filter** statement under the protocol family:

```
interfaces {
  interface-name {
    unit logical-unit-number {
      family family-name {
        ...
        filter {
          group group-number;
          input filter-name;
          input-list [ filter-names ];
          output filter-name;
          output-list [ filter-names ];
        }
      }
    }
  }
}
```

Restrictions on Applying Firewall Filters

Number of Input and Output Filters Per Logical Interface

Input filters—Although you can use the same filter multiple times, you can apply only one input filter or one input filter list to an interface.

- To specify a single firewall filter to be used to evaluate packets received on the interface, include the **input *filter-name*** statement in the **filter** stanza.
- To specify an ordered list of firewall filters to be used to evaluate packets received on the interface, include the **input-list [*filter-names*]** statement in the **filter** stanza. You can specify up to 16 firewall filters for the filter input list.

Output filters—Although you can use the same filter multiple times, you can apply only one output filter or one output filter list to an interface.

- To specify a single firewall filter to be used to evaluate packets transmitted on the interface, include the **output *filter-name*** statement in the **filter** stanza.
- To specify an ordered list of firewall filters to be used to evaluate packets transmitted on the interface, include the **output-list [*filter-names*]** statement in the **filter** stanza. You can specify up to 16 firewall filters in a filter output list.

MPLS and Layer 2 CCC Firewall Filters in Lists

The **input-list *filter-names*** and **output-list *filter-names*** statements for firewall filters for the **ccc** and **mpls** protocol families are supported on all interfaces with the exception of the following:

- Management interfaces and internal Ethernet interfaces (**fxp** or **em0**)
- Loopback interfaces (**lo0**)
- USB modem interfaces (**umd**)

Layer 2 CCC Firewall Filters on MX Series Routers

On MX Series routers only, you cannot apply a Layer 2 CCC firewall filter (a firewall filter configured at the **[edit firewall filter family ccc]** hierarchy level) as an output filter. On MX Series routers, firewall filters configured for the **family ccc** statement can be applied only as input filters.

Related Documentation

- family (Firewall) on page 299
- *family (Interfaces)*
- filter (Applying to a Logical Interface) on page 301
- filter (Configuring) on page 302
- Guidelines for Configuring Firewall Filters on page 21
- Understanding How to Use Firewall Filters on page 29

Understanding How to Use Firewall Filters

This topic covers the following information:

Using Firewall Filters to Affect Local Packets

On a router (or switch), you can configure one physical loopback interface, **lo0**, and one or more addresses on the interface. The loopback interface is the interface to the Routing Engine, which runs and monitors all the control protocols. The loopback interface carries local packets only. Firewall filters applied to the loopback interface affect the local packets destined for or transmitted from the Routing Engine.

NOTE: **When you create an additional loopback interface, it is important to apply a filter to it so the Routing Engine is protected. We recommend that when you apply a filter to the loopback interface, you include the apply-groups statement. Doing so ensures that the filter is automatically inherited on every loopback interface, including lo0 and other loopback interfaces.**

Trusted Sources

The typical use of a firewall filter is to protect the Routing Engine processes and resources from malicious or untrusted packets. To protect the processes and resources owned by the Routing Engine, you can use a firewall filter that specifies which protocols and services, or applications, are allowed to reach the Routing Engine. Applying this type of filter to the loopback interface ensures that the local packets are from a trusted source and protects the processes running on the Routing Engine from an external attack.

Flood Prevention

You can create standard stateless firewall filters that limit certain TCP and ICMP traffic destined for the Routing Engine. A router (or switch) without this kind of protection is vulnerable to TCP and ICMP flood attacks, which are also called denial-of-service (DoS) attacks. For example:

- A TCP flood attack of SYN packets initiating connection requests can overwhelm the device until it can no longer process legitimate connection requests, resulting in denial of service.
- An ICMP flood can overload the device with so many echo requests (ping requests) that it expends all its resources responding and can no longer process valid network traffic, also resulting in denial of service.

Applying the appropriate firewall filters to the Routing Engine protects against these types of attacks.

Using Firewall Filters to Affect Data Packets

Firewall filters that you apply to your router's (or switch's) transit interfaces evaluate only the user data packets that transit the router from one interface directly to another as they are being forwarded from a source to a destination. To protect the network as a whole from unauthorized access and other threats at specific interfaces, you can apply firewall filters router transit interfaces .

Related Documentation

- How Firewall Filters Evaluate Packets on page 18
- Guidelines for Configuring Firewall Filters on page 21
- Guidelines for Applying Firewall Filters on page 25

CHAPTER 3

Understanding Firewall Filters in Logical Systems

Firewall Filters in Logical Systems Overview

This topic covers the following information:

Logical Systems

With the Junos OS, you can partition a single physical router or switch into multiple logical devices that perform independent routing tasks. Because logical systems perform a subset of the tasks once handled by the physical router or switch, logical systems offer an effective way to maximize the use of a single router or switch.

Firewall Filters in Logical Systems

You can configure a separate set of firewall filters for each logical system on a router or switch. To configure a filter in a logical system, you must define the filter in the **firewall** stanza at the **[edit logical-systems *logical-system-name*]** hierarchy level, and you must apply the filter to a logical interface that is also configured at the **[edit logical-systems *logical-system-name*]** hierarchy level.

Identifiers for Firewall Objects in Logical Systems

To identify firewall objects configured under logical systems, operational **show** commands and firewall-related SNMP MIB objects include a **__logical-system-name/** prefix in the object name. For example, firewall objects configured under the **ls1** logical system include **__ls1/** as the prefix.

Related Documentation

- Stateless Firewall Filter Types on page 6
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32
- Unsupported Firewall Filter Statements for Logical Systems on page 409
- Unsupported Actions for Firewall Filters in Logical Systems on page 411
- Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods on page 183
- "*Introduction to Logical Systems*
- "*Logical Systems Operations and Restrictions*

Guidelines for Configuring and Applying Firewall Filters in Logical Systems

This topic covers the following information:

- Statement Hierarchy for Configuring Firewall Filters in Logical Systems on page 32
- Filter Types in Logical Systems on page 33
- Firewall Filter Protocol Families in Logical Systems on page 33
- Firewall Filter Match Conditions in Logical Systems on page 34
- Firewall Filter Actions in Logical Systems on page 34
- Statement Hierarchy for Applying Firewall Filters in Logical Systems on page 34

Statement Hierarchy for Configuring Firewall Filters in Logical Systems

To configure a firewall filter in a logical system, include the **filter**, **service-filter**, or **simple-filter** statement at the **[edit logical-systems *logical-system-name* firewall family *family-name*]** hierarchy level.

```
[edit]
logical systems {
  logical-system-name {
    firewall {
      family family-name {
        filter filter-name {
          interface-specific;
          physical-interface-filter;
          term term-name {
            filter filter-name;
            from {
              match-conditions;
            }
            then {
              actions;
            }
          }
        }
        service-filter filter-name { # For 'family inet' or 'family inet6' only.
          term term-name {
            from {
```

```
            match-conditions;
          }
          then {
            actions;
          }
        }
      }
      simple-filter filter-name { # For 'family inet' only.
        term term-name {
          from {
            match-conditions;
          }
          then {
            actions;
          }
        }
      }
    }
  }
}
```

Filter Types in Logical Systems

There are no special restrictions on the types of stateless firewall filter types that you can configure in logical systems.

In a logical system, you can use the same types of stateless firewall filters that are available on a physical router or switch:

- Standard stateless firewall filters
- Service filters
- Simple filters

Firewall Filter Protocol Families in Logical Systems

There are no special restrictions on the protocol families supported with stateless firewall filters in logical systems.

In a logical system, you can filter the same protocol families as you can on a physical router or switch.

- Standard stateless firewall filters—In logical systems, you can filter the following traffic types: protocol-independent, IPv4, IPv6, MPLS, MPLS-tagged IPv4 or IPv6, VPLS, Layer 2 circuit cross-connection, and Layer 2 bridging.
- Service filters—In logical systems, you can filter IPv4 and IPv6 traffic.
- Simple filters—In logical systems, you can filter IPv4 traffic only.

Firewall Filter Match Conditions in Logical Systems

There are no special restrictions on the match conditions supported with stateless firewall filters in logical systems.

Firewall Filter Actions in Logical Systems

There are no special restrictions on the actions supported with stateless firewall filters in logical systems.

Statement Hierarchy for Applying Firewall Filters in Logical Systems

To apply a firewall filter in a logical system, include the **filter** ***filter-name***, **service-filter** ***service-filter-name***, or **simple-filter** ***simple-filter-name*** statement to a logical interface in the logical system.

The following configuration shows the hierarchy levels at which you can apply the statements:

```
[edit]
logical-systems logical-system-name {
    interfaces {
        interface-name {
            unit logical-unit-number {
                family family-name {
                    filter {
                        group group-name;
                        input filter-name;
                        input-list [ filter-names ];
                        output filter-name;
                        output-list [ filter-names ]
                    }
                    rpf-check { # For 'family inet' or 'family inet6' only.
                        fail-filter filter-name;
                        mode loose;
                    }
                    service { # For 'family inet' or 'family inet6' only.
                        input {
                            service-set service-set-name <service-filter service-filter-name>;
                            post-service-filter service-filter-name;
                        }
                        output {
                            service-set service-set-name <service-filter service-filter-name>;
                        }
                    }
                    simple-filter { # For 'family inet' only.
                        input simple-filter-name;
                    }
                }
            }
        }
    }
}
```

Related Documentation

- Firewall Filters in Logical Systems Overview on page 31
- References from a Firewall Filter in a Logical System to Subordinate Objects on page 35
- References from a Firewall Filter in a Logical System to Nonfirewall Objects on page 36
- References from a Nonfirewall Object in a Logical System to a Firewall Filter on page 38
- Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods on page 183
- Unsupported Firewall Filter Statements for Logical Systems on page 409
- Unsupported Actions for Firewall Filters in Logical Systems on page 411

References from a Firewall Filter in a Logical System to Subordinate Objects

This topic covers the following information:

Resolution of References from a Firewall Filter to Subordinate Objects

If a firewall filter defined in a logical system references a subordinate object (for example, a policer or prefix list), that subordinate object must be defined within the **firewall** stanza of the same logical system. For example, if a firewall filter configuration references a policer, the firewall filter and the policer must be configured under the same **[edit logical-systems *logical-system-name* firewall]** hierarchy level.

This rule applies even if the same policer is configured under the main firewall configuration or if the same policer is configured as part of a firewall in another logical system.

Valid Reference from a Firewall Filter to a Subordinate Object

In this example, the firewall filter **filter1** references the policer **pol1**. Both **filter1** and **pol1** are defined under the same firewall object. This configuration is valid. If **pol1** had been defined under another firewall object, the configuration would not be valid.

```
[edit]
logical systems {
  ls-A {
    firewall {
      policer pol1 {
        if-exceeding {
          bandwidth-limit 401k;
          burst-size-limit 50k;
        }
        then discard;
      }
      filter filter1 {
        term one {
          from {
            source-address 12.1.0.0/16;
```

```
            }
            then {
                reject host-unknown;
            }
        }
        term two {
            from {
                source-address 12.2.0.0/16;
            }
            then policer pol1;
        }
    }
  }
 }
}
```

Related Documentation

- Firewall Filters in Logical Systems Overview on page 31
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32
- References from a Firewall Filter in a Logical System to Nonfirewall Objects on page 36
- References from a Nonfirewall Object in a Logical System to a Firewall Filter on page 38

References from a Firewall Filter in a Logical System to Nonfirewall Objects

This topic covers the following information:

- Resolution of References from a Firewall Filter to Nonfirewall Objects on page 36
- Valid Reference to a Nonfirewall Object Outside of the Logical System on page 37

Resolution of References from a Firewall Filter to Nonfirewall Objects

In many cases, a firewall configuration references objects outside the firewall configuration. As a general rule, the referenced object must be defined under the same logical system as the referencing object. However, there are cases when the configuration of the referenced object is not supported at the **[edit logical-systems *logical-system-name*]** hierarchy level.

Valid Reference to a Nonfirewall Object Outside of the Logical System

This example configuration illustrates an exception to the general rule that the objects referenced by a firewall filter in a logical system must be defined under the same logical system as the referencing object.

In the following scenario, the service filter **inetsf1** is applied to IPv4 traffic associated with the service set **fred** at the logical interface **fe-0/3/2.0**, which is on an adaptive services interface.

- Service filter **inetsf1** is defined in **ls-B** and references prefix list **prefix1**.
- Service set **fred** is defined at the main services hierarchy level, and the policy framework software searches the **[edit services]** hierarchy for the definition of the **fred** service set.

Because service rules cannot be configured in logical systems. firewall filter configurations in the **[edit logical-systems logical-system *logical-system-name*]** hierarchy are allowed to reference *service sets* outside the logical system hierarchy.

```
[edit]
logical-systems {
  ls-B {
    interfaces {
      fe-0/3/2 {
        unit 0 {
          family inet {
            service {
              input {
                service-set fred service-filter inetsf1;
              }
            }
          }
        }
      }
    }
    policy-options {
      prefix-list prefix1 {
        1.1.0.0/16;
        1.2.0.0/16;
        1.3.0.0/16;
      }
    }
    firewall { # Under logical-system 'ls-B'.
      family inet {
        filter filter1 {
          term one {
            from {
              source-address {
                12.1.0.0/16;
              }
            }
            then {
              reject host-unknown;
            }
          }
          term two {
```

```
                from {
                  source-address {
                    12.2.0.0/16;
                  }
                }
                then policer pol1;
              }
            }
            service-filter inetsf1 {
              term term1 {
                from {
                  source-prefix-list {
                    prefix1;
                  }
                }
                then count prefix1;
              }
            }
          }
          policer pol1 {
            if-exceeding {
              bandwidth-limit 401k;
              burst-size-limit 50k;
            }
            then discard;
          }
        }
      }
    } # End of logical systems configuration.
    services { # Main services hierarchy level.
      service-set fred {
        max-flows 100;
        interface-service {
          service-interface sp-1/2/0.0;
        }
      }
    }
```

Related Documentation

- Firewall Filters in Logical Systems Overview on page 31
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32
- References from a Firewall Filter in a Logical System to Subordinate Objects on page 35
- References from a Nonfirewall Object in a Logical System to a Firewall Filter on page 38

References from a Nonfirewall Object in a Logical System to a Firewall Filter

This topic covers the following information:

- Resolution of References from a Nonfirewall Object to a Firewall Filter on page 39
- Invalid Reference to a Firewall Filter Outside of the Logical System on page 39
- Valid Reference to a Firewall Filter Within the Logical System on page 40
- Valid Reference to a Firewall Filter Outside of the Logical System on page 42

Resolution of References from a Nonfirewall Object to a Firewall Filter

If a nonfirewall filter object in a logical system references an object in a firewall filter configured in a logical system, the reference is resolved using the following logic:

- If the nonfirewall filter object is configured in a logical system that includes firewall filter configuration statements, the policy framework software searches the **[edit logical-systems *logical-system-name* firewall]** hierarchy level. Firewall filter configurations that belong to *other* logical systems or to the main **[edit firewall]** hierarchy level are not searched.
- If the nonfirewall filter object is configured in a logical system that does not include any firewall filter configuration statements, the policy framework software searches the firewall configurations defined at the **[edit firewall]** hierarchy level.

Invalid Reference to a Firewall Filter Outside of the Logical System

This example configuration illustrates an unresolvable reference from a nonfirewall object in a logical system to a firewall filter.

In the following scenario, the stateless firewall filters **filter1** and **fred** are applied to the logical interface **fe-0/3/2.0** in the logical system **ls-C**.

- Filter **filter1** is defined in **ls-C**.
- Filter **fred** is defined in the main firewall configuration.

Because **ls-C** contains firewall filter statements (for **filter1**), the policy framework software resolves references to and from firewall filters by searching the **[edit logical systems ls-C firewall]** hierarchy level. Consequently, the reference from **fe-0/3/2.0** in the logical system to **fred** in the main firewall configuration cannot be resolved.

```
[edit]
logical-systems {
  ls-C {
    interfaces {
      fe-0/3/2 {
        unit 0 {
          family inet {
            filter {
              input-list [ filter1 fred ];
            }
          }
        }
      }
    }
    firewall { # Under logical system 'ls-C'.
      family inet {
        filter filter1 {
          term one {
            from {
              source-address 12.1.0.0/16;
            }
            then {
              reject host-unknown;
```

```
            }
          }
          term two {
            from {
              source-address 12.2.0.0/16;
            }
            then policer pol1;
          }
        }
      }
      policer pol1 {
        if-exceeding {
          bandwidth-limit 401k;
          burst-size-limit 50k;
        }
        then discard;
      }
    }
  }
} # End of logical systems
firewall { # Under the main firewall hierarchy level
  family inet {
    filter fred {
      term one {
        from {
          source-address 11.1.0.0/16;
        }
        then {
          log;
          reject host-unknown;
        }
      }
    }
  }
} # End of main firewall configurations.
```

Valid Reference to a Firewall Filter Within the Logical System

This example configuration illustrates resolvable references from a nonfirewall object in a logical system to two firewall filter.

In the following scenario, the stateless firewall filters **filter1** and **fred** are applied to the logical interface **fe-0/3/2.0** in the logical system **ls-C**.

- Filter **filter1** is defined in **ls-C**.
- Filter **fred** is defined in **ls-C** and also in the main firewall configuration.

Because **ls-C** contains firewall filter statements, the policy framework software resolves references to and from firewall filters by searching the **[edit logical systems ls-C firewall]** hierarchy level. Consequently, the references from **fe-0/3/2.0** in the logical system to **filter1** and **fred** use the stateless firewall filters configured in **ls-C**.

```
[edit]
logical-systems {
  ls-C {
    interfaces {
```

```
        fe-0/3/2 {
            unit 0 {
                family inet {
                    filter {
                        input-list [ filter1 fred ];
                    }
                }
            }
        }
    }
    firewall { # Under logical system 'ls-C'.
        family inet {
            filter filter1 {
                term one {
                    from {
                        source-address 12.1.0.0/16;
                    }
                    then {
                        reject host-unknown;
                    }
                }
                term two {
                    from {
                        source-address 12.2.0.0/16;
                    }
                    then policer pol1;
                }
            }
            filter fred { # This 'fred' is in 'ls-C'.
                term one {
                    from {
                        source-address 10.1.0.0/16;
                    }
                    then {
                        log;
                        reject host-unknown;
                    }
                }
            }
        }
        policer pol1 {
            if-exceeding {
                bandwidth-limit 401k;
                burst-size-limit 50k;
            }
            then discard;
        }
    }
  }
} # End of logical systems configurations.
firewall { # Main firewall filter hierarchy level
    family inet {
        filter fred {
            term one {
                from {
                    source-address 11.1.0.0/16;
```

```
            }
            then {
              log;
              reject host-unknown;
            }
          }
        }
      }
    } # End of main firewall configurations.
```

Valid Reference to a Firewall Filter Outside of the Logical System

This example configuration illustrates resolvable references from a nonfirewall object in a logical system to two firewall filter.

In the following scenario, the stateless firewall filters **filter1** and **fred** are applied to the logical interface **fe-0/3/2.0** in the logical system **ls-C**.

- Filter **filter1** is defined in the main firewall configuration.
- Filter **fred** is defined in the main firewall configuration.

Because **ls-C** does not contain any firewall filter statements, the policy framework software resolves references to and from firewall filters by searching the **[edit firewall]** hierarchy level. Consequently, the references from **fe-0/3/2.0** in the logical system to **filter1** and **fred** use the stateless firewall filters configured in the main firewall configuration.

```
[edit]
logical-systems {
  ls-C {
    interfaces {
      fe-0/3/2 {
        unit 0 {
          family inet {
            filter {
              input-list [ filter1 fred ];
            }
          }
        }
      }
    }
  }
} # End of logical systems configurations.
firewall { # Main firewall hierarchy level.
  family inet {
    filter filter1 {
      term one {
        from {
          source-address 12.1.0.0/16;
        }
        then {
          reject host-unknown;
        }
      }
      term two {
        from {
          source-address 12.2.0.0/16;
```

```
            }
            then policer pol1;
          }
        }
        filter fred {
          term one {
            from {
              source-address 11.1.0.0/16;
            }
            then {
              log;
              reject host-unknown;
            }
          }
        }
      }
      policer pol1 {
        if-exceeding {
          bandwidth-limit 701k;
          burst-size-limit 70k;
        }
        then discard;
      }
    } # End of main firewall configurations.
```

Related Documentation

- Firewall Filters in Logical Systems Overview on page 31
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32
- References from a Firewall Filter in a Logical System to Subordinate Objects on page 35
- References from a Firewall Filter in a Logical System to Nonfirewall Objects on page 36

CHAPTER 4

Firewall Filter Accounting and Logging

Accounting for Firewall Filters Overview

Juniper Networks devices can collect various kinds of data about traffic passing through the device. 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

There are several types of accounting profiles: interface, firewall filter, source class and destination class usage, and Routing Engine. If you apply the same profile name to both a firewall filter and an interface, it causes an error.

Related Documentation

- Statement Hierarchy for Configuring Firewall Filter Accounting Profiles on page 279
- Statement Hierarchy for Applying Firewall Filter Accounting Profiles on page 280
- Example: Configuring Statistics Collection for a Firewall Filter on page 187

System Logging Overview

The Junos OS generates system log messages (also called *syslog messages*) to record *system events* that occur on the device. Events consist of routine operations, failure and error conditions, and critical conditions that might require urgent resolution. This system logging utility is similar to the UNIX **syslogd** utility.

Each Junos OS system log message belongs to a message category, called a *facility*, that reflects the hardware- or software-based source of the triggering event. A group of messages belonging to the same facility are either generated by the same software

process or concern a similar hardware condition or user activity (such as authentication attempts). Each system log message is also preassigned a *severity*, which indicates how seriously the triggering event affects router (or switch) functions. Together, the facility and severity of an event are known as the message *priority*. The content of a syslog message identifies the Junos OS *process* that generates the message and briefly describes the operation or error that occurred.

By default, syslog messages that have a severity of **info** or more serious are written to the main system log file **messages** in the **/var/log** directory of the local Routing Engine. To configure global settings and facility-specific settings that override these default values, you can include statements at the **[edit system syslog]** hierarchy level.

For all syslog facilities or for a specified facility, you can configure the syslog message utility to redirect messages of a specified severity to a specified file instead of to the main system log file. You can also configure the syslog message utility to write syslog messages of a specified severity, for all syslog facilities or for a specified facility, to additional destinations. In addition to writing syslog messages to a log file, you can write syslog messages to the terminal sessions of any logged-in users, to the router (or switch) console, or to a remote host or the other Routing Engine.

At the global level—for all system logging messages, regardless of facility, severity, or destination—you can override the default values for file-archiving properties and the default timestamp format.

Related Documentation

- System Logging of Events Generated for the Firewall Facility on page 46
- Logging of Packet Headers Evaluated by a Firewall Filter Term on page 48
- Example: Configuring Logging for a Firewall Filter Term on page 192

System Logging of Events Generated for the Firewall Facility

System log messages generated for firewall filter actions belong to the **firewall** facility. Just as you can for any other Junos OS system logging facility, you can direct **firewall** facility syslog messages to one or more specific destinations: to a specified file, to the terminal session of one or more logged in users (or to all users), to the router (or switch) console, or to a remote host or the other Routing Engine on the router (or switch).

When you configure a syslog message destination for **firewall** facility syslog messages, you include a statement at the **[edit system syslog]** hierarchy level, and you specify the **firewall** facility name together with a severity level. Messages from the **firewall** that are rated at the specified level or more severe are logged to the destination.

System log messages with the **DFWD_** prefix are generated by the firewall process (**dfwd**), which manages compilation and downloading of Junos OS firewall filters. System log messages with the **PFE_FW_** prefix are messages about firewall filters, generated by the Packet Forwarding Engine controller, which manages packet forwarding functions. For more information, see the *Junos OS System Log Messages Reference*.

Table 10 on page 47 lists the system log destinations you can configure for the **firewall** facility.

Table 10: Syslog Message Destinations for the Firewall Facility

Destination	Description	Configuration Statements Under [edit system syslog]
File	Configuring this option keeps the **firewall** syslog messages out of the main system log file. To include priority and facility with messages written to the file, include the **explicit-priority** statement. To override the default standard message format, which is based on a UNIX system log format, include the **structured-data** statement. When the **structured-data** statement is included, other statements that specify the format for messages written to the file are ignored (the **explicit-priority** statement at the **[edit system syslog file *filename*]** hierarchy level and the **time-format** statement at the **[edit system syslog]** hierarchy level).	**file *filename* {** **firewall *severity*;** **allow-duplicates;** **archive *archive-optios*;** **explicit-priority;** **structured-data;** **}** **allow-duplicates;** **archive *archive-options*;** **time-format (*option*);**
Terminal session	Configuring this option causes a copy of the **firewall** syslog messages to be written to the specified terminal sessions. Specify one or more user names, or specify * for all logged in users.	**user (*username* | *) {** **firewall *severity*;** **}** **time-format (*option*);**
Router (or switch) console	Configuring this option causes a copy of the **firewall** syslog messages to be written to the router (or switch) console.	**console {** **firewall *severity*;** **}** **time-format (*option*);**
Remote host or the other Routing Engine	Configuring this option causes a copy of the **firewall** syslog messages to be written to the specified remote host or to the other Routing Engine. To override the default alternative facility for forwarding **firewall** syslog messages to a remote machine (**local3**), include the **facility-override firewall** statement. To include priority and facility with messages written to the file, include the **explicit-priority** statement.	**host (*hostname* | other-routing-engine) {** **firewall *severity*;** **allow-duplicates;** **archive *archive-optios*;** **facility-override firewall;** **explicit-priority;** **}** **allow-duplicates; # All destinations** **archive *archive-options*;** **time-format (*option*);**

By default, the timestamp recorded in a standard-format system log message specifies the month, date, hour, minute, and second when the message was logged, as in the example:

Sep 07 08:00:10

To include the year, the millisecond, or both in the timestamp for all system logging messages, regardless of the facility, include one of the following statement at the **[edit system syslog]** hierarchy level:

- **time-format year**;
- **time-format millisecond**;
- **time-format year millisecond**;

The following example illustrates the format for a timestamp that includes both the millisecond (**401**) and the year (**2010**):

Sep 07 08:00:10.401.2010

Related Documentation

- System Logging Overview on page 45
- Logging of Packet Headers Evaluated by a Firewall Filter Term on page 48
- Example: Configuring Logging for a Firewall Filter Term on page 192
- *Junos OS System Logging Facilities and Message Severity Levels*
- *Junos OS System Log Configuration Hierarchy*
- *Junos OS Default System Log Settings*
- *Logging Messages in Structured-Data Format*
- *Including the Year or Millisecond in Timestamps*
- *Changing the Alternative Facility Name for Remote System Log Messages*
- *Junos OS System Log Alternate Facilities for Remote Logging*

Logging of Packet Headers Evaluated by a Firewall Filter Term

Built in to the stateless firewall filtering software is the capability to log packet-header information for the packets evaluated by a stateless firewall filter term. You can write the packet header information to the system log file on the local Routing Engine or to a firewall filter buffer in the Packet Forwarding Engine. Logging of packet headers evaluated by firewall filters is supported for standard stateless firewall filters for IPv4 or IPv6 traffic only. Service filters and simple filters do not support logging of packet headers.

Table 11 on page 49 lists the packet-header logs you can configure for a firewall filter action.

Table 11: Packet-Header Logs for Stateless Firewall Filter Terms

Log	Description	Configuration Statements

Syslog message destinations configured for the firewall facility	Configure this option by using the **syslog** nonterminating action.
	
	NOTE: Packet header information is interspersed with event messages.
	
	To list log files, enter the **show log** operational mode command without command options.
	
	To display log file contents for a specific file in the **/var/log** directory on the local Routing Engine, enter the **show log** *filename* operational mode command or the **file show /var/log/***filename* operational mode command.
	
	To clear log file contents, enter the **clear log** *filename* **<all>** operational mode command. If you include the **all** option, the specified log file is truncated, all archived versions of the log file are deleted.
		firewall {
		  family {
		  filter *filter-name* {
		  from {
		  match-conditions;
		  }
		  then {
		  ...
		  syslog;
		  terminating-action;
		  }
		  }
		  }
		}

Buffer in the Packet Forwarding Engine	Configure this option by using the **log** nonterminating action.
	
	NOTE: Restarting the router (or switch) causes the contents of this buffer to be cleared.
	
	To display the local log entries for firewall filters, enter the **show firewall log** operational mode command.
		firewall {
		  family {
		  filter *filter-name* {
		  from {
		  match-conditions;
		  }
		  then {
		  ...
		  log;
		  terminating-action;
		  }
		  }
		  }
		}

Related Documentation

- System Logging Overview on page 45
- System Logging of Events Generated for the Firewall Facility on page 46
- Example: Configuring Logging for a Firewall Filter Term on page 192

CHAPTER 5

Multiple Firewall Filters Attached to a Single Interface

Understanding Multiple Firewall Filters in a Nested Configuration

This topic covers the following information:

The Challenge: Simplify Large-Scale Firewall Filter Administration

Typically, you apply a single firewall filter to an interface in the input or output direction or both. This approach might not be practical, however, when you have a router (or switch) configured with many, even hundreds of interfaces. In an environment of this scale, you want the flexibility of being able to modify filtering terms common to multiple interfaces without having to reconfigure the filter of every affected interface.

In general, the solution is to apply an effectively “chained” structure of multiple stateless firewall filters to a single interface. You partition your filtering terms into multiple firewall filters configured so that you can apply a unique filter to each router (or switch) interface but also apply common filters to multiple router (or switch) interfaces as required. The Junos OS policy framework provides two options for managing the application of multiple separate firewall filters to individual router (or switch) interfaces. One option is to apply multiple filters as a single input list or output list. The other option is to reference a stateless firewall filter from within the term of another stateless firewall filter.

A Solution: Configure Nested References to Firewall Filters

The most structured way to avoid configuring duplicate filtering terms common to multiple firewall filters is to configure multiple firewall filters so that each filter includes the shared filtering terms by *referencing* a separate filter that contains the common filtering terms. The Junos OS uses the filter terms—in the order in which they appear in the filter definition—to evaluate packets that transit the interface. If you need to modify filtering terms shared across multiple interfaces, you only need to modify one firewall filter.

NOTE: **Similar to the alternative approach (applying a list of firewall filters), configuring a nested firewall filter combines multiple firewall filters into a new firewall filter definition.**

Configuration of Nested Firewall Filters

Configuring a nested firewall filter for each router (or switch) interface involves separating shared packet-filtering rules from interface-specific packet-filtering rules as follows:

- For each set of packet-filtering rules common across multiple interfaces, configure a separate firewall filter that contains the shared filtering terms.
- For each router (or switch) interface, configure a separate firewall filter that contains:
  - All the filtering terms unique to that interface.
  - An additional filtering term that includes a **filter** reference to the firewall filter that contains the common filtering terms.

Application of Nested Firewall Filters to a Router or Switch Interface

Applying nested firewall filters is no different from applying an unnested firewall filter. For each interface, you can include an **input** or **output** statement (or both) within the **filter** stanza to specify the appropriate nested firewall filter.

Applying nested firewall filters to an interface, the shared filtering terms and the interface-specific firewall filters are applied through *a single nested firewall filter* that includes other filters through the **filter** statement within a separate filtering term.

Related Documentation

- Guidelines for Nesting References to Multiple Firewall Filters on page 52
- Example: Nesting References to Multiple Firewall Filters on page 202

Guidelines for Nesting References to Multiple Firewall Filters

This topic covers the following information:

- Statement Hierarchy for Configuring Nested Firewall Filters on page 53
- Filter-Defining Terms and Filter-Referencing Terms on page 53
- Types of Filters Supported in Nested Configurations on page 53

Statement Hierarchy for Configuring Nested Firewall Filters

To reference a filter from within a filter, include the **filter *filter-name*** statement as a separate filter term:

```
firewall firewall-name {
    family family-name {
        filter filter-name {
            term term-name {
                filter filter-name;
            }
        }
    }
}
```

You can include the firewall configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

Filter-Defining Terms and Filter-Referencing Terms

You cannot configure a firewall filter term that both references another firewall filter and defines a match condition or action. If a firewall filter term includes the **filter** statement, then it cannot also include the **from** or **then** statement.

For example, the firewall filter term **term term1** in the configuration is *not* valid:

```
[edit]
firewall {
    family inet {
        filter filter_1 {
            term term1 {
                filter filter_2;
                from {
                    source-address 1.1.1.1/32;
                }
                then {
                    accept;
                }
            }
        }
    }
}
```

In order for **term term1** to be a valid filter term, you must either remove the **filter filter_2** statement or remove both the **from** and **then** stanzas.

Types of Filters Supported in Nested Configurations

Nested configurations of firewall filters support firewall filters only. You cannot use service filters or simple filters in a nested firewall filter configuration.

Number of Filter References in a Single Filter

The total number of filters referenced from within a filter cannot exceed 256.

Depth of Filter Nesting

The Junos OS supports a single level of firewall filter nesting. If **filter_1** references **filter_2**, you cannot configure a filter that references a filter that references **filter_1**.

Related Documentation

- Understanding Multiple Firewall Filters in a Nested Configuration on page 51
- Example: Nesting References to Multiple Firewall Filters on page 202

Understanding Multiple Firewall Filters Applied as a List

This topic covers the following information:

The Challenge: Simplify Large-Scale Firewall Filter Administration

Typically, you apply a single firewall filter to an interface in the input or output direction or both. However, this approach might not be practical when you have a device configured with many interfaces. In large environments, you want the flexibility of being able to modify filtering terms common to multiple interfaces without having to reconfigure the filter of every affected interface.

In general, the solution is to apply an effectively “chained” structure of multiple firewall filters to a single interface. You partition your filtering terms into multiple firewall filters that each perform a filtering task. You can then choose which filtering tasks you want to perform for a given interface and apply the filtering tasks to that interface. In this way, you only manage the configuration for a filtering task in a single firewall filter.

The Junos OS policy framework provides two options for managing the application of multiple separate firewall filters to individual router interfaces. One option is to apply multiple filters as a single input list or output list. The other option is to reference a firewall filter from within the term of another firewall filter.

A Solution: Apply Lists of Firewall Filters

The most straightforward way to avoid configuring duplicate filtering terms common to multiple firewall filters is to configure multiple firewall filters and then apply a customized *list* of filters to each interface. The Junos OS uses the filters—in the order in which they appear in the list—to evaluate packets that transit the interface. If you need to modify filtering terms shared across multiple interfaces, you only need to modify one firewall filter that contains those terms.

NOTE: In contrast with the alternative approach (configuring nested firewall filters) applying firewall filter lists combines multiple firewall filters at each interface application point.

Configuration of Multiple Filters for Filter Lists

Configuring firewall filters to be applied in unique lists for each router interface involves separating shared packet-filtering rules from interface-specific packet-filtering rules as follows:

- **Unique filters**—For each set of packet-filtering rules unique to a specific interface, configure a separate firewall filter that contains only the filtering terms for that interface.
- **Shared filters**—For each set of packet-filtering rules common across two or more interfaces, consider configuring a separate firewall filter that contains the shared filtering terms.

TIP: When planning for a large number firewall filters to be applied using filter lists, administrators often organize the shared filters by filtering criteria, by the services to which customers subscribe, or by the purposes of the interfaces.

Application of Filter Lists to a Router Interface

Applying a list of firewall filters to an interface is a matter of selecting the filters that meet the packet-filtering requirements of that interface. For each interface, you can include an **input-list** or **output-list** statement (or both) within the **filter** stanza to specify the relevant filters in the order in which they are to be used:

- Include any filters that contain common filtering terms relevant to the interface.
- Include the filter that contain only the filtering terms unique to the interface.

Interface-Specific Names for Filter Lists

Because a filter list is configured under an interface, the resulting concatenated filter is *interface-specific*.

NOTE: When a filter list is configured under an interface, the resulting concatenated filter is interface-specific, regardless whether the firewall filters in the filter list are configured as interface-specific or not. Furthermore, the instantiation of interface-specfic firewall filters not only create separate instances of any firewall filter counters, but also separate instances of any policer actions. Any policers applied through an action specified in the firewall filter configuration are applied separately to each interface in the interface group.

The system-generated name of an interface-specific filter consists of the full interface name followed by either '**-i**' for an input filter list or '**-o**' for an output filter list.

- **Input filter list name**—For example, if you use the **input-list** statement to apply a chain of filters to logical interface **ge-1/3/0.0**, the Junos OS uses the following name for the filter:

  ge-1/3/0.0-i

- **Output filter list name**—For example, if you use the **output-list** statement to apply a chain of filters to logical interface **fe-0/1/2.0**, the Junos OS uses the following name for the filter:

  fe-0/1/2.0-o

You can use the interface-specific name of a filter list when you enter a Junos OS operational mode command that specifies a firewall filter name.

How Filter Lists Evaluate Packets When the Matched Term Includes Terminating or Next Term Actions

The device evaluates a packet against the filters in a list sequentially, beginning with the first filter in the list until either a terminating action occurs or the packet is implicitly discarded.

Table 12 on page 57 describes how a firewall filter list evaluates a packet based on whether the matched term specifies a terminating action and the **next term** action. The **next term** action is neither a terminating action nor a nonterminating action but a *flow control* action.

Table 12: Firewall Filter List Behavior

Firewall Filter Actions Included in the Matched Term		Term Description	Packet-Filtering Behavior
Terminating	next term		
Yes	—	The matched term includes a terminating action (such as **discard**) but not the **next term** action	The device executes the terminating action. No subsequent terms in the filter and no subsequent filters in the list are used to evaluate the packet.
—	Yes	The matched term includes the **next term** action, but it does not include any terminating actions.	The device executes any nonterminating actions, then the device evaluates the packet against the next term in the filter or the next filter in the list.
—	—	The matched term includes neither the **next term** action nor any terminating actions.	The device executes any nonterminating actions, then the device implicitly accepts the packet. Because the **accept** action is a terminating action, no subsequent terms in the filter and no subsequent filters in the list are used to evaluate the packet.

For information about terminating actions, see "Firewall Filter Terminating Actions" on page 384.

NOTE: **You cannot configure the next term action with a terminating action in the same firewall filter term.**

How Filter Lists Evaluate Packets When the List Includes Protocol-Independent and IP Firewall Filters

On a single interface associated with a protocol-independent (**family any**) firewall filter and a protocol-specific (**family inet** or **family inet6**) firewall filter simultaneously, the protocol-independent firewall filter executes first.

The terminating action of the first filter determines whether the second filter also evaluates the packet:

- If the first filter terminates by executing the **accept** action, the second filter also evaluates the packet.
- If the first filter terminates without any terms matching the packet (an *implicit* **discard** action), the second filter also evaluates the packet.
- If the first filter terminates by executing an *explicit* **discard** action, the second filter does not evaluate the packet.

Related Documentation

- How Firewall Filters Evaluate Packets on page 18
- Guidelines for Applying Multiple Firewall Filters as a List on page 58
- Example: Applying Lists of Multiple Firewall Filters on page 197

Guidelines for Applying Multiple Firewall Filters as a List

This topic covers the following information:

Statement Hierarchy for Applying Lists of Multiple Firewall Filters

To apply a single filter to the input or output direction of a router (or switch) logical interface, you include the **input** ***filter-name*** or **output** ***filter-name*** statement under the **filter** stanza for a protocol family.

To apply a list of multiple filters to the input or output direction of a router (or switch) logical interface, include the **input-list [** ***filter-names*** **]** or **output-list [** ***filter-names*** **]** statement under the **filter** stanza for a protocol family:

```
interfaces {
   interface-name {
      unit logical-unit-number {
         family family-name {
            filter {
               ...filter-options...
               input-list [ filter-names ];
               output-list [ filter-names ];
            }
         }
      }
   }
}
```

You can include the interface configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems** ***logical-system-name*****]**

Filter Input Lists and Output Lists for Router or Switch Interfaces

When applying a list of firewall filters as a list, the following limitations apply:

- You can specify up to 16 firewall filters for a filter input list.
- You can specify up to 16 firewall filters for a filter output list.

Types of Filters Supported in Lists

Lists of multiple firewall filters applied to a router (or switch) interface support standard stateless firewall filters only. You cannot apply lists containing service filters or simple filters to a router (or switch) interface.

Restrictions on Applying Filter Lists for MPLS or Layer 2 CCC Traffic

When applying firewall filters that evaluate MPLS traffic (**family mpls**) or Layer 2 circuit cross-connection traffic (**family ccc**), you can use the **input-list [*filter-names*]** and **output-list [*filter-names*]** statements for all interfaces except the following:

- Management and internal Ethernet (**fxp**) interfaces
- Loopback (**lo0**) interfaces
- USB modem (**umd**) interfaces

Related Documentation

- Understanding Multiple Firewall Filters Applied as a List on page 54
- Example: Applying Lists of Multiple Firewall Filters on page 197

CHAPTER 6

Single Firewall Filter Attached to Multiple Interfaces

Interface-Specific Firewall Filter Instances Overview

This topic covers the following information:

Instantiation of Interface-Specific Firewall Filters

On T Series, M120, M320, and MX Series routers, you can enable the Junos OS to automatically create an interface-specific instance of a firewall filter for each interface to which you apply the filter. If you enable interface-specific instantiation of a firewall filter and then apply that filter to multiple interfaces, any **count** actions or **policer** actions configured in the filter terms act on the traffic stream entering or exiting each individual interface, regardless of the sum of traffic on the multiple interfaces.

You can enable this option per firewall filter by including the **interface-specific** statement in the filter configuration.

NOTE: **On T Series, M120, M320, and MX Series routers, interfaces are distributed among multiple packet-forwarding components.**

Interface-specific firewall filtering is not supported on M Series routers other than the M120 and M320 routers. If you apply a firewall filter to multiple interfaces on an M Series router other than the M120 or M320 routers, the filter acts on the sum of traffic entering or exiting those interfaces.

Interface-specific firewall filtering is supported for standard stateless firewall filters and for service filters. Interface-specific instances are not supported for simple filters.

Interface-Specific Names for Firewall Filter Instances

When the Junos OS creates a separate instance of a firewall filter for a logical interface, the instance is associate with an interface-specific name. The system-generated name of a firewall filter instance consists of the name of the configured filter followed by a hyphen ('-'), the full interface name, and either '**-i**' for an input filter instance or '**-o**' for an output filter instance.

- **Input filter instance name**—For example, if you apply the interface-specific firewall filter **filter_s_tcp** to the input at logical interface **at-1/1/1.0**, the Junos OS instantiates an interface-specific filter instance with the following system-generated name:

  filter_s_tcp-at-1/1/1.0-i

- **Output filter instance name**—For example, if you apply the interface-specific firewall filter **filter_s_tcp** to the output at logical interface **so-2/2/2.2**, the Junos OS instantiates an interface-specific filter instance with the following system-generated name:

  count_s_tcp-so-2/2/2.2-o

You can use the interface-specific name of a filter instance when you enter a Junos OS operational mode command that specifies a stateless firewall filter name.

TIP: When you configure a firewall filter with interface-specific instances enabled, we recommend you limit the filter name to *52 bytes* in length. This is because firewall filter names are restricted to *64 bytes* in length. If a system-generated filter instance name exceeds this maximum length, the policy framework software might reject the instance name.

Interface-Specific Firewall Filter Counters

Instantiation of interface-specific firewall filters causes the Packet Forwarding Engine to maintain any counters for the firewall filter separately for each interface. You specify interface-specific counters per firewall filter term by specifying the **count *counter-name*** non-terminating action.

The system-generated name of an interface-specific firewall filter counter consists of the name of the configured counter followed by a hyphen ('-'), the full interface name, and either '**-i**' for an input filter instance or '**-o**' for an output filter instance.

- **Interface-specific input filter counter name**—For example, suppose you configure the filter counter **count_tcp** for an interface-specific firewall filter. If the filter is applied to the input at logical interface **at-1/1/1.0**, the Junos OS creates the following system-generated counter name:

  count_tcp-at-1/1/1.0-i

- **Interface-specific output filter counter name**—For example, suppose you configure the filter counter **count_udp** for an interface-specific firewall filter. If the filter is applied to the output at logical interface **so-2/2/2.2**, the Junos OS creates the following system-generated counter name:

  count_udp-so-2/2/2.2-o

Interface-Specific Firewall Filter Policers

Instantiation of interface-specific firewall filters not only creates separate instances of any firewall filter counters but also creates separate instances of any policer actions. Any policers applied through an action specified in the firewall filter configuration are applied separately to each interface in the interface group. You specify interface-specific policers per firewall filter term by specifying the **policer *policer-name*** non-terminating action.

Related Documentation

- Statement Hierarchy for Configuring Interface-Specific Firewall Filters on page 269
- Statement Hierarchy for Applying Interface-Specific Firewall Filters on page 270
- Example: Configuring Interface-Specific Firewall Filter Counters on page 207

Filtering Packets Received on a Set of Interface Groups Overview

You can configure a firewall filter term that matches packets tagged for a specified *interface group* or set of interface groups. An interface group consists of one or more logical interfaces with the same group number. Packets received on an interface in an interface group are tagged as being part of that group.

For standard stateless firewall filters, you can filter packets received on an interface group for IPv4, IPv6, virtual private LAN service (VPLS), Layer 2 circuit cross-connection (CCC), and Layer 2 bridging traffic. For service filters, you can filter packets received on an interface group for either IPv4 or IPv6 traffic.

NOTE: You can also configure a firewall filter term that matches on packets tagged for a specified *interface set*. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.

Related Documentation

- Statement Hierarchy for Assigning Interfaces to Interface Groups on page 271
- Statement Hierarchy for Configuring a Filter to Match on a Set of Interface Groups on page 271
- Example: Filtering Packets Received on an Interface Group on page 211

Filtering Packets Received on an Interface Set Overview

You can configure a standard stateless firewall filter term that matches packets tagged for a specified *interface set*. An interface set groups two or more physical or logical

interfaces into a single interface-set name. You can filter packets received on an interface set for protocol-independent, IPv4, IPv6, MPLS, VPLS, or bridging traffic.

NOTE: You can also configure a standard stateless firewall filter term or a service filter term that matches on packets tagged for a specified *interface group*. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63.

Related Documentation

- Statement Hierarchy for Defining an Interface Set on page 273
- Statement Hierarchy for Configuring a Filter to Match on an Interface Set on page 273
- Example: Configuring a Rate-Limiting Filter Based on Destination Class on page 252
- Example: Filtering Packets Received on an Interface Set on page 116

CHAPTER 7

Filter-Based Tunneling Across IP Networks

Understanding Filter-Based Tunneling Across IPv4 Networks

This topic covers the following information:

Understanding Filter-Based Tunneling Across IPv4 Networks

Generic routing encapsulation (GRE) in its simplest form is the encapsulation of any network layer protocol over any other network layer protocol to connect disjoint networks that lack a native routing path between them. You can configure an IPv4 network to transport IPv4, IPv6, or MPLS transit traffic by using GRE tunneling protocol mechanisms initiated by two standard firewall filter actions. This feature is also supported in logical systems.

When you configure GRE tunneling with firewall filters, you do not need to create tunnel interfaces on Tunnel Services physical interface cards (PICs) or on MPC3E Modular Port Concentrators (MPCs). Instead, Packet Forwarding Engines provide tunnel services to Ethernet logical interfaces or aggregated Ethernet interfaces hosted on Modular Interface Cards (MICs) or MPCs in MX Series 3D Universal Edge Routers.

NOTE: **GRE is a connectionless and stateless Layer 3 encapsulation protocol, and it offers no mechanisms for reliability, flow control, or sequencing. Traffic flows through the tunnel provided that the tunnel destination is routable.**

Two MX Series routers installed as provider edge (PE) routers provide connectivity to customer edge (CE) routers on two disjoint networks. MIC or MPC interfaces on the PE routers perform GRE IPv4 encapsulation and de-encapsulation of payloads.

Ingress Firewall Filter on the Ingress PE Router

On the ingress PE router, you configure a tunnel definition that specifies a unidirectional GRE tunnel. On a MIC or MPC ingress logical interface, you attach an encapsulating firewall filter. The firewall filter action references a tunnel definition and initiates the encapsulation of matched packets. The encapsulation process attaches an IPv4 header and a GRE header to the payload packet and then forwards the resulting GRE packet to the filter-specified tunnel.

Ingress Firewall Filter on the Egress PE Router

On the egress PE router, you attach a de-encapsulating firewall filter to the input of all MIC or MPC logical interfaces that are advertised addresses for the router. The firewall filter initiates the de-encapsulation of GRE protocol packets. De-encapsulation removes the inner GRE header and then forwards the original payload packet to its original destination on the destination customer network. If the action specifies an optional routing instance, route lookup is performed using that secondary table instead of the primary table.

Characteristics of Filter-Based Tunneling Across IPv4 Networks

Filter-based tunnels across IPv4 networks are unidirectional. They transport transit packets only, and they do not require tunnel interfaces.

Unidirectional Tunneling

Filter-based tunneling across IPv4 networks is unidirectional. You construct a filter-based GRE tunnel by attaching standard firewall filters at the *input* of each tunnel endpoint (at both the ingress PE router and the egress PE router). At the input to the ingress PE router, you apply an encapsulating firewall filter. At the input to the egress PE router, you apply a de-encapsulating firewall filter.

If you want to configure bidirectional GRE tunneling, you can use the same pair of PE routers, but you must configure a second tunnel in the reverse direction.

Transit Traffic Payloads

A filter-based GRE IPv4 tunnel can transport unicast or multicast transit traffic payloads only. Filter-initiated encapsulation and de-encapsulation operations execute on Packet Forwarding Engines for Ethernet logical interfaces and aggregated Ethernet interfaces hosted on MICs or MPCs in MX Series routers. This design enables more efficient use of Packet Forwarding Engine bandwidth as compared to GRE tunneling using tunnel interfaces. One of the trade-offs for this optimization, however, is the inability to transport router control traffic.

Packet Forwarding Engines operate in the Junos OS *forwarding plane* to process packets by forwarding them between input and output interfaces using a locally stored forwarding table (a local copy of the information from the Routing Engine). Routing Engines, on the other hand, operate in the Junos OS *control plane* to handle system management, user access to the router, and processes for routing protocols, router interface control, and some chassis component control. The Junos OS architecture separates the functions of these planes to enable flexibility of platform support and scalability of platform

performance. Ingress control packets are directed to the control plane where the GRE encapsulation and de-encapsulation processes of the Packet Forwarding Engine are not available.

Although you can apply firewall filters to loopback addresses, GRE encapsulating and de-encapsulating firewall filter actions are not supported on router loopback interfaces.

Compact Configuration for Multiple GRE Tunnels

Firewall filters support a wide variety of match criteria and, by extension, the ability to terminate multiple GRE tunnels that match criteria specified in a single firewall filter definition. By creating multiple tunnels, each with its own set of match conditions, you can create tunnels that do not interfere with customer GRE packets or with one another and that re-inject packets to separate routing tables after de-encapsulation.

Tunneling with Firewall Filters and Tunneling with Tunnel Interfaces

Unlike tunneling with firewall filters, tunneling with tunnel interfaces supports router control traffic (in addition to transit traffic) and encryption. On the other hand, tunneling with firewall filters carries advantages in performance and scaling.

Tunnel Security

Filter-based tunneling across IPv4 networks is not encrypted. If you require secure tunneling, you must use IP Security (IPsec) encryption, which is not supported on MIC or MPC interfaces. However, Multiservices DPC (MS-DPC) interfaces on MX240, MX480, and MX960 routers support IPsec tools for configuring manual or dynamic security associations (SAs) for encryption of data traffic as well as traffic destined to or originating at the Routing Engine.

For information about Junos OS support for the IPsec security suite for the IPv4 and IPv6 network layers, see the *System Basics: Security Services ConfigurationAdministration Guide for Routing Devices*, the *Junos VPN Site Secure* guide, and *Enabling Service Packages*.

IPsec encryption is also supported on Adaptive Services PIC interfaces and Multiservices PIC interfaces on supported M Series Multiservice Edge Routers and T Series Core Routers.

Forwarding Performance

Filter-based tunneling across IPv4 networks enables more efficient use of Packet Forwarding Engine bandwidth as compared to GRE tunneling using tunnel interfaces. Encapsulation, de-encapsulation, and route lookup are packet header-processing activities that, for firewall filter-based tunneling, are performed on the Junos Trio chipset-based Packet Forwarding Engine. Consequently, the encapsulator never needs to send payload packets to a separate tunnel interface (which might reside on a PIC in a different slot than the interface that receives payload packets).

Forwarding Scalability

Forwarding GRE traffic with tunnel interfaces requires traffic to be sent to a slot that hosts the tunnel interfaces. When you use tunnel interfaces to forward GRE traffic, this requirement limits the amount of traffic that can be forwarded per GRE tunnel destination address.

As an example, suppose you want to send 100 Gbps of GRE traffic from Router A to Router B and you have only 10 Gbps interfaces. To ensure that your configuration does not encapsulate all the traffic on the same board going to the same 10 Gbps interface, you must distribute the traffic across multiple encapsulation points.

Related Documentation

- Interfaces That Support Filter-Based Tunneling Across IPv4 Networks on page 68
- Components of Filter-Based Tunneling Across IPv4 Networks on page 69
- Firewall Filter Terminating Actions on page 384
- **tunnel-end-point on page 313**
- Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling on page 217

Interfaces That Support Filter-Based Tunneling Across IPv4 Networks

You can attach IPv4 encapsulation and de-encapsulation firewall filters to the input of Ethernet logical interfaces or aggregated Ethernet interfaces hosted on Modular Interface Cards (MICs) or Modular Port Concentrators (MPCs) in MX Series routers.

- Interfaces on MX240, MX480, MX960, MX2010, and MX2020 Routers on page 68
- Interfaces on MX5, MX10, MX40, and MX80 Routers on page 69
- CLI Commit Check for Filter-Based Tunneling Across IPv4 Networks on page 69

Interfaces on MX240, MX480, MX960, MX2010, and MX2020 Routers

On MX240, MX480, MX960, MX2010, and MX2020 routers, firewall filter actions for IPv4 tunneling are supported on Ethernet logical interfaces or aggregated Ethernet interfaces configured on the following types of ports:

- Ports on MICs that insert into slots in MPCs, which have two Packet Forwarding Engines.
- Ports on a 16-port 10-Gigabit Ethernet MPC (MPC-3D-16XGE-SFPP), a specialized fixed-configuration MPC that has four Packet Forwarding Engines and contains no slots for MICs.

For these physical interfaces, Trio chipset-based Packet Forwarding Engine processes operate in *fabric mode* to provide forwarding and storage functions and lookup and processing functions between Ethernet interfaces and the routing fabric of the chassis.

For information about MPCs, see *MX Series MPC Overview* and *MPCs Supported by MX240, MX480, MX960, MX2010, and MX2020 Routers*. For information about MICs, see *MX Series MIC Overview* and *MICs Supported by MX Series Routers*.

Interfaces on MX5, MX10, MX40, and MX80 Routers

On the MX Series midrange family of routers (MX5, MX10, MX40, and MX80 routers), firewall filter actions for IPv4 tunneling are supported on Ethernet logical interfaces and aggregated Ethernet interfaces configured on ports on a built-in MIC or on MICs that install into dedicated slots in the router chassis.

- The MX80 router—available as a modular (MX80) or fixed (MX80-48T) chassis—has a built-in 4-port 10-Gigabit Ethernet MIC. The modular chassis has two dedicated slots for MICs. The fixed chassis has 48 built-in tri-rate (10/100/1000Base-T) RJ-45 ports in place of two front-pluggable MIC slots.
- On the MX40 router, only the first two of the four built-in 10-Gigabit Ethernet MIC ports are enabled. As with the modular MX80, the two front-pluggable MIC slots are enabled and support dual-wide MICs that span the two slots.
- The MX5 and MX10 routers are pre-populated with a front-pluggable 20-port Gigabit Ethernet MIC with SFP, and none of the four built-in 10-Gigabit Ethernet MIC ports is enabled. The MX10 supports MICs in both front-pluggable slots, but the MX5 supports MICs in the second slot only.

For more information, see *MX5, MX10, MX40, and MX80 Modular Interface Card Description*.

The MX Series midrange routers have no switching fabric, and the single Packet Forwarding Engine resides on the base board of the chassis and operates in *standalone mode*. In standalone mode, the Packet Forwarding Engine provides—in addition to forwarding and storage functions and lookup and processing functions—hierarchical queuing, congestion management, and granular statistical functions.

CLI Commit Check for Filter-Based Tunneling Across IPv4 Networks

If you commit a configuration that attaches an encapsulating or de-encapsulating firewall filter to an interface that does not support filter-based tunneling across IPv4 networks, a system event writes a syslog warning message that the interface does not support the filter.

Related Documentation

- Understanding Filter-Based Tunneling Across IPv4 Networks on page 65
- Components of Filter-Based Tunneling Across IPv4 Networks on page 69
- Firewall Filter Terminating Actions on page 384
- **tunnel-end-point on page 313**
- Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling on page 217

Components of Filter-Based Tunneling Across IPv4 Networks

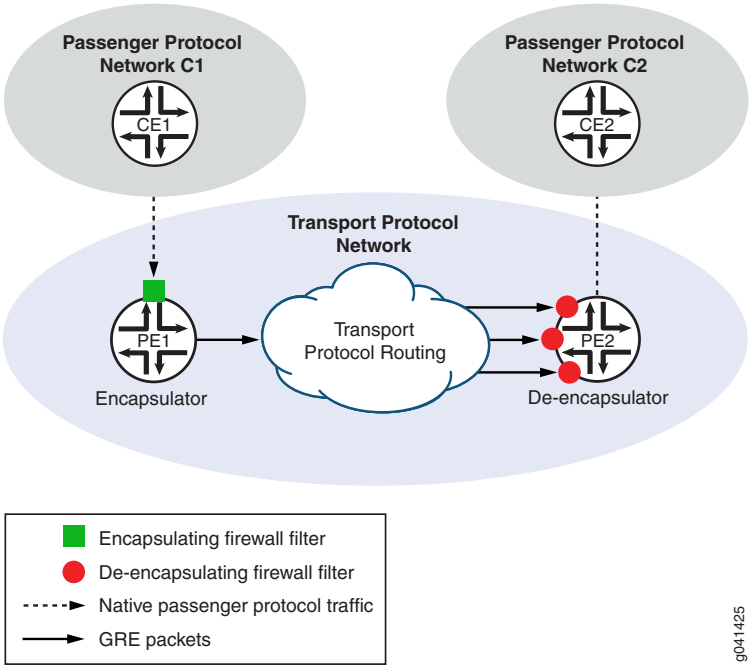
This topic covers the following information:

- Topology of Filter-Based Tunneling Across IPv4 Networks on page 70
- Terminology at the Network Layer Protocols Level on page 71

Topology of Filter-Based Tunneling Across IPv4 Networks

Figure 2 on page 70 shows the path of passenger protocol packets from customer network C1 as they are transported across a service provider IPv4 network to customer network C2.

Figure 2: Unidirectional Filter-Based Tunnel Across an IPv4 Network



In this example topology, C1 and C2 are disjoint networks that lack a native routing path between them. The IPv4 transport network is configured with a unidirectional generic routing encapsulation (GRE) tunnel from PE1 to PE2 using firewall filters and without requiring tunnel interfaces. The GRE tunnel from PE1 to PE2 provides a logical path from C1 to C2 across the IPv4 transport network.

Routing of GRE Packets Across the Tunnel

Traffic flows through the tunnel provided that PE2 is routable from PE1. Routing paths from PE1 to PE2 can be provided by static routes manually added to routing tables or by static or dynamic route-sharing protocols.

Routing of Passenger Protocol Packets from PE2 to C2

By default, PE2 forwards packets based on interface routes (direct routes) imported from the primary routing table. As an option, the de-encapsulating filter can specify that the Packet Forwarding Engine uses an alternate routing table to forward payload packets to the destination customer network. Specify the alternate routing table in a routing

instance installed with routes into C2, then use a routing information base (RIB) group definition to share the primary routes with the alternate routes. A RIB group specifies the sharing of routing information (including routes learned from peers, local routes resulting from the application of protocol policies to the learned routes, and the routes advertised to peers) of multiple routing tables.

Terminology at the Network Layer Protocols Level

In filter-based tunneling across an IPv4 network, the network-layer protocols are described in the following terms:

passenger protocol—The type of protocol (IPv4, IPv6, or MPLS) used by the networks that are connected by a GRE tunnel. Packets that are encapsulated and routed across the transport network are *payload packets*.

encapsulation protocol—The type of network layer protocol (GRE) used to encapsulate passenger protocol packets so that the resulting GRE packets can be carried over the transport protocol network as the packet payload.

transport protocol—The type of protocol (IPv4) used by the network that routes passenger protocol packets through a GRE tunnel. The transport protocol is also called the *delivery protocol*.

Terminology at the Ingress PE Router

In filter-based tunneling across an IPv4 network, an egress PE router is described in the following terms:

encapsulator—A PE router that receives packets from a passenger protocol source network, adds an encapsulation protocol (GRE) header and a transport protocol (IPv4) header to this payload, and forwards the resulting GRE packet to the GRE tunnel. This ingress node is also known as the *tunnel source*.

encapsulating interface—On the encapsulator, an Ethernet logical interface or an aggregated Ethernet interface configured on a customer-facing interface hosted on a MIC or an MPC. The encapsulating interface receives passenger protocol packets from a CE router. For more information, see "Interfaces That Support Filter-Based Tunneling Across IPv4 Networks" on page 68.

encapsulation filter—On the encapsulator, a firewall filter that you apply to the input of the encapsulating interface. The encapsulating filter action causes the Packet Forwarding Engine to use information in the specified tunnel template to encapsulate matched packets and forward the resulting GRE packets.

tunnel source interface—On the encapsulator, one or more core-facing egress interfaces to the tunnel.

tunnel template—On the encapsulator, a named CLI construct that defines the characteristics of a tunnel:

- Transport protocol family (IPv4).
- IP address or address range of tunnel-facing *egress* interfaces on the encapsulator.

- IP address or address range of tunnel-facing *ingress* interfaces on the de-encapsulator (the egress PE router).
- Encapsulation protocol (GRE).

Terminology at the Egress PE Router

In filter-based tunneling across IPv4 networks, an egress PE router is described in the following terms:

de-encapsulator—A PE router that receives GRE packets routed through a filter-based GRE tunnel, removes the transport protocol header and GRE header, and forwards the resulting payload protocol packets to the destination network CE router. The de-encapsulator node is also known as a *de-encapsulating tunnel endpoint* or the *tunnel destination*.

de-encapsulating interfaces—On the de-encapsulator, any Ethernet logical interface or aggregated Ethernet interface configured on any core-facing ingress interface that can receive GRE packets from a GRE tunnel. The underlying physical interface must be hosted on a MIC or an MPC. For more information, see "Interfaces That Support Filter-Based Tunneling Across IPv4 Networks" on page 68.

de-encapsulation filter—On the de-encapsulator, a firewall filter that causes the Packet Forwarding Engine to de-encapsulate matched GRE packets and then forward the original passenger protocol packets to destination network CE routers.

GRE packets transported through a single GRE tunnel can arrive at the de-encapsulator node on any of multiple ingress interfaces, depending on how routing is configured. Therefore, you must apply the de-encapsulation firewall filter to the input of every core-facing interface that is an advertised address for the de-encapsulator.

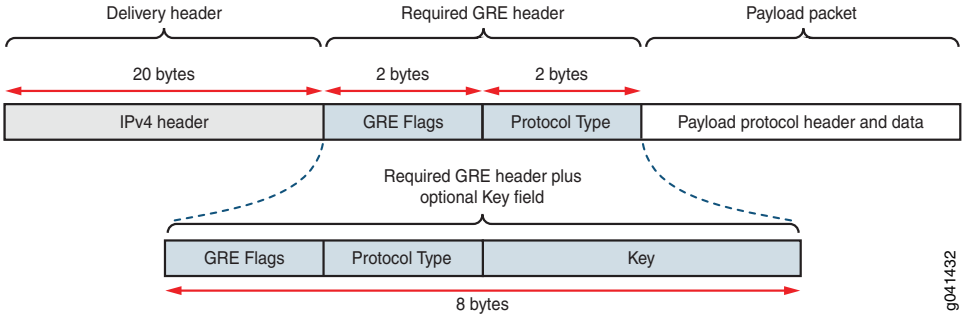
GRE Protocol Format for Filter-Based Tunneling Across IPv4 Networks

In filter-based tunneling across IPv4 networks, the encapsulating interface is an *RFC 1701-compliant transmitter* and the de-encapsulating interfaces are *RFC 1701-compliant receivers*. The packet encapsulation structure implemented in this feature uses a GRE header format that complies with informational RFC 1701, *Generic Routing Encapsulation (GRE)*, October 1994, and with standards track RFC 2784, *Generic Routing Encapsulation (GRE)*, March 2000.

Packet Encapsulation Structure

Filter-based tunneling encapsulates the original passenger protocol packet in an outer shell. For filter-based tunneling across IPv4 networks, the shell adds 24 bytes or 28 bytes of overhead, including 20 bytes of IPv4 header. Figure 3 on page 73 shows the structure of a passenger protocol packet (the GRE payload) with a GRE header and IPv4 header attached.

Figure 3: Encapsulation Structure for Filter-Based Tunneling Across an IPv4 Network

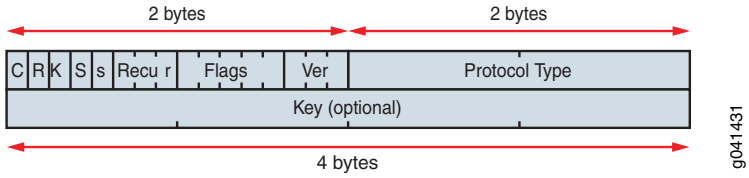


As specified in RFC 1701, five GRE flag bits indicate whether a particular GRE header includes any optional fields (Checksum, Offset, Key, Sequence Number, and Routing). Of the five optional fields, filter-based GRE IPv4 tunneling uses the Key field only.

GRE Header Format

Figure 4 on page 73 shows the format of the variable-size GRE header used for filter-based tunneling across IPv4 networks, with bit 0 the most significant bit and bit 15 the least significant bit.

Figure 4: GRE Header Format for Filter-Based Tunneling Across IPv4 Networks



The first two octets encode GRE flags, as described in Table 13 on page 73.

The 2-octet Protocol Type field contains the value 0x0800 to specify the EtherType value for the IPv4 protocol.

The 4-octet Key field is included only if the Key Present bit is set to 1. The Key field carries the key value of the tunnel defined on the encapsulator. If the GRE tunnel definition specifies a key, the Packet Forwarding Engine for the encapsulating endpoint sets the Key Present bit and adds the Key to the GRE header.

Table 13: GRE Flag Values for Filter-Based Tunneling Across IPv4 Networks

Bit Offset and Field Name		Transmitted Value for Filter-Based GRE Tunneling	
0	**C** = Checksum Present	**0**	Checksum field is not used.
1	**R** = Routing Present	**0**	Offset and Routing fields are not used.
2	**K** = Key Present	**0** or **1**	Transmitted as **0** for a keyless tunnel or **1** for a keyed tunnel.

Table 13: GRE Flag Values for Filter-Based Tunneling Across IPv4 Networks *(continued)*

Bit Offset and Field Name		Transmitted Value for Filter-Based GRE Tunneling	
3	**S** = Sequence Number Present	**0**	Sequence Number field is not used.
4	**s** = Strict Source Route	**0**	Not all routing information is Strict Source Routes.
5 - 7	**Recur** = Recursion Control information	**000**	No additional encapsulations are permitted.
8 - 12	**Flags** = Flag bits	**00000**	Reserved.
13 - 15	**Ver** = Version number	**000**	Reserved.

When the Packet Forwarding Engine performs encapsulation for a keyed GRE IPv4 tunnel, the process constructs the first two octets of the GRE header as 0x0000. When the Packet Forwarding Engine performs encapsulation for a non-keyed GRE IPv4 tunnel, the process constructs the first two octets of the GRE header as 0x2000.

Related Documentation

- Understanding Filter-Based Tunneling Across IPv4 Networks on page 65
- Interfaces That Support Filter-Based Tunneling Across IPv4 Networks on page 68
- Firewall Filter Terminating Actions on page 384
- **tunnel-end-point on page 313**
- Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling on page 217

CHAPTER 8

Firewall Filters for Forwarding, Fragments, and Policing

Filter-Based Forwarding Overview

Filters That Classify Packets or Direct Them to Routing Instances

For IPv4 or IPv6 traffic only, you can use stateless firewall filters in conjunction with forwarding classes and routing instances to control how packets travel in a network. This is called *filter-based forwarding* (FBF).

You can define a filtering term that matches incoming packets based on source address and then classifies matching packets to a specified forwarding class. This type of filtering can be configured to grant certain types of traffic preferential treatment or to improve load balancing. To configure a stateless firewall filter to classify packets to a forwarding class, configure a term with the *nonterminating action* **forwarding-class *class-name***.

You can also define a filtering term that directs matching packets to a specified routing instance. This type of filtering can be configured to route specific types of traffic through a firewall or other security device before the traffic continues on its path. To configure a stateless firewall filter to direct traffic to a routing instance, configure a term with the *terminating action* **routing-instance *routing-instance-name* <topology *topology-name*>** to specify the routing instance to which matching packets will be forwarded.

To forward traffic to the master routing instance, reference **routing-instance default** in the firewall configuration, as shown here:

```
[edit firewall]
family inet {
```

```
filter test {
  term 1 {
    then {
      routing-instance default;
    }
  }
}
```

NOTE: Do not reference routing-instance master. This does not work.

Input Filtering to Classify and Forward Packets Within the Router or Switch

You can configure filters to classify packets based on source address and specify the forwarding path the packets take within the router or switch by configuring a filter on the ingress interface.

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 or switch 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.

Output Filtering to Forward Packets to Another Routing Table

You can also forward packets based on output filters by configuring a filter on the egress interfaces. In the case of port mirroring, it is useful for port-mirrored packets to be distributed to multiple monitoring PICs and collection PICs based on patterns in packet headers. FBF on the port-mirroring egress interface must be configured.

Packets forwarded to the output filter have been through at least one route lookup when an FBF filter is configured on the egress interface. After the packet is classified at the egress interface by the FBF filter, it is redirected to another routing table for further route lookup.

Restrictions for Applying Filter-Based Forwarding

An interface configured with filter-based forwarding does not support source-class usage (SCU) filter matching, source-class and destination-class usage (SCU/DCU) accounting, or unicast reverse-path forwarding (RPF) check filters.

Related Documentation

- Example: Configuring Filter-Based Forwarding on the Source Address on page 233
- Example: Configuring Filter-Based Forwarding on Logical Systems on page 173

Firewall Filters That Handle Fragmented Packets Overview

You can create stateless firewall filters that handle fragmented packets destined for the Routing Engine. By applying these policies to the Routing Engine, you protect against the use of IP fragmentation as a means to disguise TCP packets from a firewall filter.

For example, consider an IP packet that is fragmented into the smallest allowable fragment size of 8 bytes (a 20-byte IP header plus an 8-byte payload). If this IP packet carries a TCP packet, the first fragment (fragment offset of 0) that arrives at the device contains only the TCP source and destination ports (first 4 bytes), and the sequence number (next 4 bytes). The TCP flags, which are contained in the next 8 bytes of the TCP header, arrive in the second fragment (fragment offset of 1).

See RFC 1858, *Security Considerations for IP Fragment Filtering*.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Handle Fragments on page 167

Stateless Firewall Filters That Reference Policers Overview

Policing, or rate limiting, is an important component of firewall filters that lets you limit the amount of traffic that passes into or out of an interface.

A firewall filter that references a policer can provide protection from denial-of-service (DOS) attacks. Traffic that exceeds the rate limits configured for the policer is either discarded or marked as lower priority than traffic that conforms to the configured rate limits. Packets can be marked for a lower priority by being set to a specific output queue, set to a specific packet loss priority (PLP) level, or both. When necessary, low-priority traffic can be discarded to prevent congestion.

A policer specifies two types of rate limits on traffic:

- Bandwidth limit—The average traffic rate permitted, specified as a number of bits per second.
- Maximum burst size—The packet size permitted for bursts of data that exceed the bandwidth limit.

Policing uses an algorithm to enforce a limit on average bandwidth while allowing bursts up to a specified maximum value. You can use policing to define specific classes of traffic on an interface and apply a set of rate limits to each class. After you name and configure a policer, it is stored as a template. You can then apply the policer in an interface configuration or, to rate-limit packet-filtered traffic only, in a firewall filter configuration.

For an IPv4 firewall filter term only, you can also specify a *prefix-specific action* as a nonterminating action that applies a policer to the matched packets. A prefix-specific action applies additional matching criteria on the filter-matched packets based on specified address prefix bits and then associates the matched packets with a counter and policer instance for that filter term or for all terms in the firewall filter.

To apply a policer or a prefix action to packet-filtered traffic, you can use the following firewall filter nonterminating actions:

- **policer** *policer-name*
- **three-color-policer (single-rate | two-rate)** *policer-name*
- **prefix-action** *action-name*

NOTE: The packet lengths that a policer considers depends on the address family of the firewall filter. See *Understanding the Frame Length for Policing Packets*.

Related Documentation

- Firewall Filter Nonterminating Actions on page 377
- *Traffic Policing Overview*
- *Prefix-Specific Counting and Policing Overview*

CHAPTER 9

Understanding Service Filters

Service Filter Overview

This topic covers the following information:

Services

The Adaptive Services Physical Interface Cards (PICs), Multiservices PICs, and Multiservices Dense Port Concentrators (DPCs) provide *adaptive services interfaces*. Adaptive services interfaces enable you to coordinate a special range of services on a single PIC or DPC by configuring a set of services and applications.

NOTE: **Service filters are not supported on T4000 routers.**

Service Rules

A *service set* is an optional definition you can apply to the traffic at an adaptive services interface. A service set enables you to configure combinations of directional rules and default settings that control the behavior of each service in the service set.

Service Rule Refinement

When you apply a service set to the traffic at an adaptive services interface, you can optionally use *service filters* to refine the target of the set of services and also to process traffic. Service filters enable you to manipulate traffic by performing packet filtering to a defined set of services on an adaptive services interface before the traffic is delivered

to its destination. You can apply a service filter to traffic before packets are accepted for input or output service processing or after packets return from input service processing.

Service Filter Counters

Like standard firewall filters, service filters support counting of matched packets. When you display counters for a service filter, however, the syntax for specifying the filter name includes the name of the *service set* to which the service filter is applied.

- To enable counting of the packets matched by a service filter term, specify the **count *counter-name*** nonterminating action in that term.
- To display counters for service filters, use the **show firewall filter *filter-name* <counter *counter-name*>** operational mode command, and specify the ***filter-name*** as follows:

  __service-*service-set-name***:***service-filter-name*

For example, suppose you configure a service filter named **out_filter** with a counter named **out_counter** and apply that service filter to a logical interface to direct certain packets for processing by the output services associated with the service set **nat_set**. In this scenario, the syntax for using the **show firewall** operational mode command to display the counter is as follows:

```
[edit]
user@host> show firewall filter __service-nat_set:out_filter counter out_counter
```

Related Documentation

- Stateless Firewall Filter Types on page 6
- How Service Filters Evaluate Packets on page 80
- Guidelines for Configuring Service Filters on page 82
- Guidelines for Applying Service Filters on page 84
- Example: Configuring and Applying Service Filters on page 257
- *Adaptive Services Overview*
- *Configuring Service Sets to be Applied to Services Interfaces*
- *Configuring Service Rules*

How Service Filters Evaluate Packets

This topic covers the following information:

Service Filters That Contain a Single Term

For a service filter that consists of a single term, the policy framework software evaluates a packet as follows:

- If the packet matches all the conditions, the actions are taken.
- If the packet matches all the conditions and no actions are specified, the packet is accepted.
- If the packet does not match all the conditions, it is discarded.

Service Filters That Contain Multiple Terms

For a service filter that consists of multiple terms, the policy framework software evaluates a packet against the terms in the filter sequentially, beginning with the first term in the filter, until either the packet matches all the conditions in one of the terms or there are no more terms in the filter.

- If the packet matches all the conditions in a term, the actions in that term are performed and evaluation of the packet ends at that term. Any subsequent terms in the filter are not used.
- If the packet does not match all the conditions in the term, evaluation of the packet proceeds to the next term in the filter.

Service Filter Terms That Do Not Contain Any Match Conditions

For service filters with a single term and for filters with multiple terms, if a term does not contain any match conditions, the actions are taken on any packet evaluated.

Service Filter Terms That Do Not Contain Any Actions

If a term does not contain any actions, and if the packet matches the conditions in the term, the packet is accepted.

Service Filter Default Action

Each service filter has an *implicit* **skip** action at the end of the filter, which is equivalent to including the following example term **explicit_skip** as the final term in the service filter:

```
term explicit_skip {
  then skip;
}
```

By default, if a packet matches none of the terms in a service filter, the packet bypasses service processing.

Related Documentation

- Service Filter Overview on page 79
- Guidelines for Configuring Service Filters on page 82
- Guidelines for Applying Service Filters on page 84
- Example: Configuring and Applying Service Filters on page 257

Guidelines for Configuring Service Filters

This topic covers the following information:

Statement Hierarchy for Configuring Service Filters

To configure a service filter, include the **service-filter** ***service-filter-name*** statement at the **[edit firewall family (inet | inet6)]** hierarchy level:

```
[edit]
firewall {
    family (inet | inet6) {
        service-filter service-filter-name {
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    actions;
                }
            }
        }
    }
}
```

Individual statements supported under the **service-filter** ***service-filter-name*** statement are described separately in this topic and are illustrated in the example of configuring and applying a service filter.

Service Filter Protocol Families

You can configure service filters to filter IPv4 traffic (**family inet**) and IPv6 traffic (**family inet6**) only. No other protocol families are supported for service filters.

Service Filter Names

Under the **family inet** or **family inet6** statement, you can include **service-filter** ***service-filter-name*** statements to create and name service filters. The filter name can contain letters, numbers, and hyphens (-) and be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

Service Filter Terms

Under the **service-filter** ***service-filter-name*** statement, you can include **term** ***term-name*** statements to create and name filter terms.

- You must configure at least one term in a firewall filter.
- You must specify a unique name for each term within a firewall filter. The term name can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).
- The order in which you specify terms within a firewall filter configuration is important. Firewall filter terms are evaluated in the order in which they are configured. By default, new terms are always added to the end of the existing filter. You can use the **insert** configuration mode command to reorder the terms of a firewall filter.

Service Filter Match Conditions

Service filter terms support only a subset of the IPv4 and IPv6 match conditions that are supported for standard stateless firewall filters.

If you specify an IPv6 address in a match condition (the **address**, **destination-address**, or **source-address** match conditions), use the syntax for text representations described in RFC 2373, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see *IPv6 Overview* and *Supported IPv6 Standards*.

Service Filter Terminating Actions

When configuring a service filter term, you must specify one of the following filter-terminating actions:

- **service**
- **skip**

NOTE: These actions are unique to service filters.

Service filter terms support only a subset of the IPv4 and IPv6 nonterminating actions that are supported for standard stateless firewall filters:

- **count** ***counter-name***
- **log**
- **port-mirror**
- **sample**

Service filters do not support the **next** action.

Related Documentation

- Service Filter Overview on page 79
- How Service Filters Evaluate Packets on page 80

Guidelines for Applying Service Filters

This topic covers the following information:

Restrictions for Adaptive Services Interfaces

The following restrictions apply to adaptive services interfaces and service filters.

Adaptive Services Interfaces

You can apply a service filter to IPv4 or IPv6 traffic associated with a service set at an *adaptive services interface* only. Adaptive services interfaces are supported for the following hardware only:

- Adaptive Services (AS) PICs on M Series and T Series routers
- Multiservices (MS) PICs on M Series and T Series routers
- Multiservices (MS) DPCs on MX Series routers (and EX Series switches)

System Logging to a Remote Host from M Series Routers

Logging of adaptive services interfaces messages to an external server by means of the **fxp0** or **em0** port is not supported on M Series routers. The architecture does not support system logging traffic out of a management interface. Instead, access to an external server is supported on a Packet Forwarding Engine interface.

Statement Hierarchy for Applying Service Filters

You can enable packet filtering of IPv4 or IPv6 traffic before a packet is accepted for input or output service processing. To do this, apply a service filter to the adaptive services interface input or output in conjunction with an interface service set.

You can also enable packet filtering of IPv4 or IPv6 traffic that is returning to the Packet Forwarding Engine after input service processing completes. To do this, apply a post-service filter to the adaptive services interface input.

The following configuration shows the hierarchy levels at which you can apply the service filters to adaptive services interfaces:

```
[edit]
interfaces {
  interface-name {
    unit unit-number {
      family (inet | inet6) {
        service {
          input {
            service-set service-set-name service-filter service-filter-name;
            post-service-filter service-filter-name;
          }
          output {
            service-set service-set-name service-filter service-filter-name;
          }
        }
      }
    }
  }
}
```

Associating Service Rules with Adaptive Services Interfaces

To define and group the service rules be applied to an adaptive services interface, you define an *interface service set* by including the **service-set *service-set-name*** statement at the **[edit services]** hierarchy level.

To apply an interface service set to the input and output of an adaptive services interface, you include the **service-set *service-set-name*** at the following hierarchy levels:

- **[edit interfaces *interface-name* unit *unit-number* input]**
- **[edit interfaces *interface-name* unit *unit-number* output]**

If you apply a service set to one direction of an adaptive services interface but do not apply a service set to the other direction, an error occurs when you commit the configuration.

The adaptive services PIC performs different actions depending on whether the packet is sent to the PIC for input service or for output service. For example, you can configure a single service set to perform Network Address Translation (NAT) in one direction and destination NAT (dNAT) in the other direction.

Filtering Traffic Before Accepting Packets for Service Processing

To filter IPv4 or IPv6 traffic before accepting packets for input or output service processing, include the **service-set *service-set-name* service-filter *service-filter-name*** at one of the following interfaces:

- **[edit interfaces *interface-name* unit *unit-number* family (inet | inet6) service input]**
- **[edit interfaces *interface-name* unit *unit-number* family (inet | inet6) service output]**

For the ***service-set-name***, specify a service set configured at the **[edit services service-set]** hierarchy level.

The service set retains the input interface information even after services are applied, so that functions such as filter-class forwarding and destination class usage (DCU) that depend on input interface information continue to work.

The following requirements apply to filtering inbound or outbound traffic before accepting packets for service processing:

- You configure the same service set on the input and output sides of the interface.
- If you include the **service-set** statement without an optional **service-filter** definition, the Junos OS assumes the match condition is true and selects the service set for processing automatically.
- The service filter is applied only if a service set is configured and selected.

You can include more than one service set definition on each side of an interface. The following guidelines apply:

- If you include multiple service sets, the router (or switch) software evaluates them in the order in which they appear in the configuration. The system executes the first service set for which it finds a match in the service filter and ignores the subsequent definitions.
- A maximum of six service sets can be applied to an interface.
- When you apply multiple service sets to an interface, you must also configure and apply a service filter to the interface.

Postservice Filtering of Returning Service Traffic

As an option to filtering of IPv4 or IPv6 input service traffic, you can apply a service filter to IPv4 or IPv6 traffic that is returning to the services interface after the service set is executed. To apply a service filter in this manner, include the **post-service-filter** ***service-filter-name*** statement at the **[edit interfaces** ***interface-name*** **unit** ***unit-number*** **family (inet | inet6) service input]** hierarchy level.

Related Documentation

- Service Filter Overview on page 79
- How Service Filters Evaluate Packets on page 80
- Guidelines for Configuring Service Filters on page 82
- Example: Configuring and Applying Service Filters on page 257
- *Adaptive Services Overview*
- *Configuring Service Sets to be Applied to Services Interfaces*
- *Configuring Service Rules*

CHAPTER 10

Understanding Simple Filters

Simple Filter Overview

Simple filters are supported on Gigabit Ethernet intelligent queuing 2 (IQ2) and Enhanced Queuing Dense Port Concentrator (DPC) interfaces only.

Simple filters are recommended for metropolitan Ethernet applications.

Related Documentation

- How Simple Filters Evaluate Packets on page 87
- Guidelines for Configuring Simple Filters on page 89
- Guidelines for Applying Simple Filters on page 92
- Example: Configuring and Applying a Simple Filter on page 263

How Simple Filters Evaluate Packets

This topic covers the following information:

Simple Filters That Contain a Single Term

For a simple filter that consists of a single term, the policy framework software evaluates a packet as follows:

- If the packet matches all the conditions, the actions are taken.

- If the packet matches all the conditions and no actions are specified, the packet is accepted.
- If the packet does not match all the conditions, it is discarded.

Simple Filters That Contain Multiple Terms

For a simple filter that consists of multiple terms, the policy framework software evaluates a packet against the terms in the filter sequentially, beginning with the first term in the filter, until either the packet matches all the conditions in one of the terms or there are no more terms in the filter.

- If the packet matches all the conditions in a term, the actions in that term are performed and evaluation of the packet ends at that term. Any subsequent terms in the filter are not used.
- If the packet does not match all the conditions in the term, evaluation of the packet proceeds to the next term in the filter.

Simple Filter Terms That Do Not Contain Any Match Conditions

For simple filters with a single term and for filters with multiple terms, if a term does not contain any match conditions, the actions are taken on any packet evaluated.

Simple Filter Terms That Do Not Contain Any Actions

If a simple filter term does not contain any actions, and if the packet matches the conditions in the term, the packet is accepted.

Simple Filter Default Action

Each simple filter has an *implicit* **discard** action at the end of the filter, which is equivalent to including the following example term **explicit_discard** as the final term in the simple filter:

```
term explicit_discard {
  then discard;
}
```

By default, if a packet matches none of the terms in a simple filter, the packet is discarded.

Related Documentation

- Simple Filter Overview on page 87
- Guidelines for Configuring Simple Filters on page 89
- Guidelines for Applying Simple Filters on page 92
- Example: Configuring and Applying a Simple Filter on page 263

Guidelines for Configuring Simple Filters

This topic covers the following information:

Statement Hierarchy for Configuring Simple Filters

To configure a simple filter, include the **simple-filter** ***simple-filter-name*** statement at the **[edit firewall family inet]** hierarchy level.

```
[edit]
firewall {
    family inet {
        simple-filter simple-filter-name {
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    actions;
                }
            }
        }
    }
}
```

Individual statements supported under the **simple-filter** ***simple-filter-name*** statement are described separately in this topic and are illustrated in the example of configuring and applying a simple filter.

Simple Filter Protocol Families

You can configure simple filters to filter IPv4 traffic (**family inet**) only. No other protocol family is supported for simple filters.

Simple Filter Names

Under the **family inet** statement, you can include **simple-filter** ***simple-filter-name*** statements to create and name simple filters. The filter name can contain letters, numbers, and hyphens (-) and be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).

Simple Filter Terms

Under the **simple-filter *simple-filter-name*** statement, you can include **term *term-name*** statements to create and name filter terms.

- You must configure at least one term in a firewall filter.
- You must specify a unique name for each term within a firewall filter. The term name can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ ”).
- The order in which you specify terms within a firewall filter configuration is important. Firewall filter terms are evaluated in the order in which they are configured. By default, new terms are always added to the end of the existing filter. You can use the **insert** configuration mode command to reorder the terms of a firewall filter.

Simple filters do *not* support the **next term** action.

Simple Filter Match Conditions

Simple filter terms support only a subset of the IPv4 match conditions that are supported for standard stateless firewall filters.

Unlike standard stateless firewall filters, the following restrictions apply to simple filters:

- On MX Series routers with the Enhanced Queuing DPC and on EX Series switches, simple filters do *not* support the **forwarding- class** match condition.
- Simple filters support only one **source-address** and one **destination-address** prefix for each filter term. If you configure multiple prefixes, only the last one is used.
- Simple filters do *not* support multiple source addresses and destination addresses in a single term. If you configure multiple addresses, only the last one is used.
- Simple filters do *not* support negated match conditions, such as the **protocol-except** match condition or the **exception** keyword.
- Simple filters support a range of values for **source-port** and **destination-port** match conditions only. For example, you can configure **source-port 400-500** or **destination-port 600-700**.
- Simple filters do *not* support noncontiguous mask values.

Table 14 on page 90 lists the simple filter match conditions.

Table 14: Simple Filter Match Conditions

Match Condition	Description
destination-address *destination-address*	Match IP destination address.

Table 14: Simple Filter Match Conditions *(continued)*

Match Condition	Description
destination-port *number*	TCP or UDP destination port field. If you configure this match condition, we recommend that you also configure the **protocol** match statement to determine which protocol is being used on the port. In place of the numeric value, you can specify one of the following text aliases (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), or **xdmcp** (177).
forwarding-class *class*	Match the forwarding class of the packet. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**. For information about forwarding classes and router-internal output queues, see *Overview of Forwarding Classes*.
protocol *number*	IP protocol field. In place of the numeric value, you can specify one of the following text aliases (the field values are also listed): **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **ospf** (89), **pim** (103), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).
source-address *ip-source-address*	Match the IP source address.
source-port *number*	Match the UDP or TCP source port field. If you configure this match condition, we recommend that you also configure the **protocol** match statement to determine which protocol is being used on the port. In place of the numeric field, you can specify one of the text aliases listed for **destination-port**.

Simple Filter Terminating Actions

Simple filters do *not* support explicitly configurable terminating actions, such as **accept**, **reject**, and **discard**. Terms configured in a simple filter always accept packets.

Simple filters do *not* support the **next** action.

Simple Filter Nonterminating Actions

Simple filters support only the following nonterminating actions:

- **forwarding-class (*forwarding-class* | assured-forwarding |best-effort | expedited-forwarding | network-control)**

NOTE: On the MX Series routers and EX Series switches with the Enhanced Queuing DPC, the forwarding class is not supported as a **from** match condition.

- **loss-priority (high | low | medium-high | medium-low)**

Simple filters do not support actions that perform other functions on a packet (such as incrementing a counter, logging information about the packet header, sampling the packet data, or sending information to a remote host using the system log functionality).

Related Documentation

- Simple Filter Overview on page 87
- How Simple Filters Evaluate Packets on page 87
- Guidelines for Applying Simple Filters on page 92
- Example: Configuring and Applying a Simple Filter on page 263

Guidelines for Applying Simple Filters

This topic covers the following information:

- Statement Hierarchy for Applying Simple Filters on page 92
- Restrictions for Applying Simple Filters on page 92

Statement Hierarchy for Applying Simple Filters

You can apply a simple filter to the IPv4 ingress traffic at a logical interface by including the **simple-filter input** *simple-filter-name* statement at the **[edit interfaces** *interface-name* **unit** *unit-number* **family inet]** hierarchy level.

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

Restrictions for Applying Simple Filters

You can apply a simple filter to the ingress IPv4 traffic at a logical interface configured on the following hardware only:

- Gigabit Ethernet intelligent queuing (IQ2) PICs installed on M120, M320, or T Series routers.

- Enhanced Queuing Dense Port Concentrators (EQ DPCs) installed on MX Series routers and EX Series switches.

The following additional restrictions pertain to applying simple filters:

- Simple filters are not supported on Modular Port Concentrator (MPC) interfaces, including Enhanced Queuing MPC interfaces.
- Simple filters are not supported for interfaces in an aggregated-Ethernet bundle.
- You can apply simple filters to **family inet** traffic only. No other protocol family is supported.
- You can apply simple filters to ingress traffic only. Egress traffic is not supported.
- You can apply only a single simple filter to a supported logical interface. Input lists are not supported.

Related Documentation

- Simple Filter Overview on page 87
- How Simple Filters Evaluate Packets on page 87
- Guidelines for Configuring Simple Filters on page 89
- Example: Configuring and Applying a Simple Filter on page 263

PART 2

Configuration

CHAPTER 11

Firewall Filters Applied to Routing Engine Traffic

Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List

This example shows how to configure a standard stateless firewall filter that limits certain TCP and Internet Control Message Protocol (ICMP) traffic destined for the Routing Engine by specifying a list of prefix sources that contain allowed BGP peers.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except BGP peers that have a specified prefix.

A source prefix list, **plist_bgp179**, is created that specifies the list of source prefixes that contain allowed BGP peers.

The stateless firewall filter **filter_bgp179** matches all packets from the source prefix list **plist_bgp179** to the destination port number 179.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

```
set policy-options prefix-list plist_bgp179 apply-path "protocols bgp group <*> neighbor
  <*>"
set firewall family inet filter filter_bgp179 term 1 from source-address 0.0.0.0/0
set firewall family inet filter filter_bgp179 term 1 from source-prefix-list plist_bgp179 except
set firewall family inet filter filter_bgp179 term 1 from destination-port bgp
set firewall family inet filter filter_bgp179 term 1 then reject
set firewall family inet filter filter_bgp179 term 2 then accept
set interfaces lo0 unit 0 family inet filter input filter_bgp179
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Configure the Filter

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the filter:

1. Expand the prefix list **bgp179** to include all prefixes pointed to by the BGP peer group defined by **protocols bgp group <*> neighbor <*>**.

   ```
   [edit policy-options prefix-list plist_bgp179]
   user@host# set apply-path "protocols bgp group <*> neighbor <*>"
   ```

2. Define the filter term that rejects TCP connection attempts to port 179 from all requesters except the specified BGP peers.

   ```
   [edit firewall family inet filter filter_bgp179]
   user@host# set term term1 from source-address 0.0.0.0/0
   user@host# set term term1 from source-prefix-list bgp179 except
   user@host# set term term1 from destination-port bgp
   user@host# set term term1 then reject
   ```

3. Define the other filter term to accept all packets.

   ```
   [edit firewall family inet filter filter_bgp179]
   ```

```
user@host# set term term2 then accept
```

4. Apply the firewall filter to the loopback interface.

```
[edit interfaces lo0 unit 0 family inet]
user@host# set filter input filter_bgp179
user@host# set address 127.0.0.1/32
```

Results

From configuration mode, confirm your configuration by entering the **show firewall**, **show interfaces**, and **show policy-options** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet {
    filter filter_bgp179 {
        term 1 {
            from {
                source-address {
                    0.0.0.0/0;
                }
                source-prefix-list {
                    plist_bgp179 except;
                }
                destination-port bgp;
            }
            then {
                reject;
            }
        }
        term 2 {
            then {
                accept;
            }
        }
    }
}
```

```
user@host# show interfaces
lo0 {
    unit 0 {
        family inet {
            filter {
                input filter_bgp179;
            }
            address 127.0.0.1/32;
        }
    }
}
```

```
user@host# show policy-options
prefix-list plist_bgp179 {
    apply-path "protocols bgp group <*> neighbor <*>";
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Repeat the procedure, where appropriate, for every BGP-enabled device in the network, using the appropriate interface names and addresses for each BGP-enabled device.

Verification

Confirm that the configuration is working properly.

Displaying the Firewall Filter Applied to the Loopback Interface

Purpose Verify that the firewall filter **filter_bgp179** is applied to the IPv4 input traffic at logical interface **lo0.0**.

Action Use the **show interfaces statistics** operational mode command for logical interface **lo0.0**, and include the **detail** option. Under the **Protocol inet** section of the command output section, the **Input Filters** field displays the name of the stateless firewall filter applied to the logical interface in the input direction:

```
[edit]
user@host> show interfaces statistics lo0.0 detail
  Logical interface lo0.0 (Index 321) (SNMP ifIndex 16) (Generation 130)
    Flags: SNMP-Traps Encapsulation: Unspecified
    Traffic statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Transit statistics:
     Input  bytes  :                   0                   0 bps
     Output bytes  :                   0                   0 bps
     Input  packets:                   0                   0 pps
     Output packets:                   0                   0 pps
    Protocol inet, MTU: Unlimited, Generation: 145, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter_bgp179
      Addresses, Flags: Primary
        Destination: Unspecified, Local: 127.0.0.1, Broadcast: Unspecified,
Generation: 138
```

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Firewall Filter Match Conditions Based on Address Fields on page 328
- Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods on page 128
- Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags on page 113
- prefix-list on page 306

Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources

This example shows how to create a stateless firewall filter that protects the Routing Engine from traffic originating from untrusted sources.

Requirements

No special configuration beyond device initialization is required before configuring stateless firewall filters.

Overview

In this example, you create a stateless firewall filter called protect-RE that discards all traffic destined for the Routing Engine except SSH and BGP protocol packets from specified trusted sources. This example includes the following firewall filter terms:

- **ssh-term**—Accepts TCP packets with a source address of **192.168.122.0/24** and a destination port that specifies SSH.
- **bgp-term**—Accepts TCP packets with a source address of **10.2.1.0/24** and a destination port that specifies BGP.
- **discard-rest-term**—For all packets that are not accepted by **ssh-term** or **bgp-term**, creates a firewall filter log and system logging records, then discards all packets.

NOTE: You can move terms within the firewall filter using the **insert** command. See *insert* in the *CLI User Guide*.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter protect-RE term ssh-term from source-address
  192.168.122.0/24
set firewall family inet filter protect-RE term ssh-term from protocol tcp
set firewall family inet filter protect-RE term ssh-term from destination-port ssh
set firewall family inet filter protect-RE term ssh-term then accept
set firewall family inet filter protect-RE term bgp-term from source-address 10.2.1.0/24
set firewall family inet filter protect-RE term bgp-term from protocol tcp
set firewall family inet filter protect-RE term bgp-term from destination-port bgp
set firewall family inet filter protect-RE term bgp-term then accept
set firewall family inet filter protect-RE term discard-rest-term then log
```

```
set firewall family inet filter protect-RE term discard-rest-term then syslog
set firewall family inet filter protect-RE term discard-rest-term then discard
set interfaces lo0 unit 0 family inet filter input protect-RE
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the stateless firewall filter:

1. Create the stateless firewall filter.

   ```
   [edit]
   user@host# edit firewall family inet filter protect-RE
   ```

2. Create the first filter term.

   ```
   [edit firewall family inet filter protect-RE]
   user@host# edit term ssh-term
   ```

3. Define the protocol, destination port, and source address match conditions for the term.

   ```
   [edit firewall family inet filter protect-RE term ssh-term]
   user@host# set from protocol tcp destination-port ssh source-address
     192.168.122.0/24
   ```

4. Define the actions for the term.

   ```
   [edit firewall family inet filter protect-RE term ssh-term]
   user@host# set then accept
   ```

5. Create the second filter term.

   ```
   [edit firewall family inet filter protect-RE]
   user@host# edit term bgp-term
   ```

6. Define the protocol, destination port, and source address match conditions for the term.

   ```
   [edit firewall family inet filter protect-RE term bgp-term]
   user@host# set from protocol tcp destination-port bgp source-address 10.2.1.0/24
   ```

7. Define the action for the term.

   ```
   [edit firewall family inet filter protect-RE term bgp-term]
   user@host# set then accept
   ```

8. Create the third filter term.

   ```
   [edit firewall family inet filter protect-RE]
   user@host# edit term discard-rest-term
   ```

9. Define the action for the term.

   ```
   [edit firewall family inet filter protect-RE term discard-rest]
   user@host# set then log syslog discard
   ```

10. Apply the filter to the input side of the Routing Engine interface.

    ```
    [edit]
    ```

```
user@host# set interfaces lo0 unit 0 family inet filter input protect-RE
```

Results Confirm your configuration by entering the **show firewall** command and the **show interfaces lo0** command from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet {
    filter protect-RE {
        term ssh-term {
            from {
                source-address {
                    192.168.122.0/24;
                }
                protocol tcp;
                destination-port ssh;
            }
            then accept;
        }
        term bgp-term {
            from {
                source-address {
                    10.2.1.0/24;
                }
                protocol tcp;
                destination-port bgp;
            }
            then accept;
        }
        term discard-rest-term {
            then {
                log;
                syslog;
                discard;
            }
        }
    }
}
```

```
user@host# show interfaces lo0
unit 0 {
    family inet {
        filter {
            input protect-RE;
        }
        address 127.0.0.1/32;
    }
}
```

If you are done configuring the device, enter **commit** from configuration mode.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, perform these tasks:

- Displaying Stateless Firewall Filter Configurations on page 104
- Verifying a Services, Protocols, and Trusted Sources Firewall Filter on page 104
- Displaying Stateless Firewall Filter Logs on page 105

Displaying Stateless Firewall Filter Configurations

Purpose Verify the configuration of the firewall filter.

Action From configuration mode, enter the **show firewall** command and the **show interfaces lo0** command.

Meaning Verify that the output shows the intended configuration of the firewall filter. In addition, verify that the terms are listed in the order in which you want the packets to be tested. You can move terms within a firewall filter by using the **insert** CLI command.

Verifying a Services, Protocols, and Trusted Sources Firewall Filter

Purpose Verify that the actions of the firewall filter terms are taken.

Action Send packets to the device that match the terms. In addition, verify that the filter actions are *not* taken for packets that do not match.

- Use the **ssh *host-name*** command from a host at an IP address that matches **192.168.122.0/24** to verify that you can log in to the device using only SSH from a host with this address prefix.
- Use the **show route summary** command to verify that the routing table on the device does not contain any entries with a protocol other than **Direct**, **Local**, **BGP**, or **Static**.

Sample Output

```
% ssh 192.168.249.71
%ssh host
user@host's password:
--- JUNOS 6.4-20040518.0 (JSERIES) #0: 2004-05-18 09:27:50 UTC

user@host>

user@host> show route summary
Router ID: 192.168.249.71

inet.0: 34 destinations, 34 routes (33 active, 0 holddown, 1 hidden)
              Direct:     10 routes,      9 active
               Local:      9 routes,      9 active
                 BGP:     10 routes,     10 active
              Static:      5 routes,      5 active
...
```

Meaning Verify the following information:

- You can successfully log in to the device using SSH.
- The **show route summary** command does not display a protocol other than **Direct**, **Local**, **BGP**, or **Static**.

Displaying Stateless Firewall Filter Logs

Purpose Verify that packets are being logged. If you included the **log** or **syslog** action in a term, verify that packets matching the term are recorded in the firewall log or your system logging facility.

Action From operational mode, enter the **show firewall log** command.

Sample Output

```
user@host> show firewall log
Log :
Time      Filter    Action Interface     Protocol Src Addr      Dest Addr
15:11:02  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
15:11:01  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
15:11:01  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
15:11:01  pfe       D      ge-0/0/0.0    TCP      172.17.28.19  192.168.70.71
...
```

Meaning Each record of the output contains information about the logged packet. Verify the following information:

- Under **Time**, the time of day the packet was filtered is shown.
- The **Filter** output is always **pfe**.
- Under **Action**, the configured action of the term matches the action taken on the packet—**A** (accept), **D** (discard), **R** (reject).
- Under **Interface**, the inbound (ingress) interface on which the packet arrived is appropriate for the filter.
- Under **Protocol**, the protocol in the IP header of the packet is appropriate for the filter.
- Under **Src Addr**, the source address in the IP header of the packet is appropriate for the filter.
- Under **Dest Addr**, the destination address in the IP header of the packet is appropriate for the filter.

Related Documentation

- *show route summary*
- *show firewall*
- *show firewall log*
- *show interfaces (Loopback)*

Example: Configuring a Filter to Block Telnet and SSH Access

Requirements

You must have access to a remote host that has network connectivity with this router or switch.

Overview

In this example, you create an IPv4 stateless firewall filter that logs and rejects Telnet or SSH access packets unless the packet is destined for or originates from the 192.168.1.0/24 subnet.

- To match packets destined for or originating from the **address 192.168.1.0/24** subnet, you use the **address 192.168.1.0/24** IPv4 match condition.
- To match packets destined for or originating from a TCP port, Telnet port, or SSH port, you use the **protocol tcp**, **port telnet**, and **telnet ssh** IPv4 match conditions.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter local_acl term terminal_access from address 192.168.1.0/24
set firewall family inet filter local_acl term terminal_access from protocol tcp
set firewall family inet filter local_acl term terminal_access from port ssh
set firewall family inet filter local_acl term terminal_access from port telnet
set firewall family inet filter local_acl term terminal_access then accept
set firewall family inet filter local_acl term terminal_access_denied from protocol tcp
set firewall family inet filter local_acl term terminal_access_denied from port ssh
set firewall family inet filter local_acl term terminal_access_denied from port telnet
set firewall family inet filter local_acl term terminal_access_denied then log
set firewall family inet filter local_acl term terminal_access_denied then reject
set firewall family inet filter local_acl term default-term then accept
set interfaces lo0 unit 0 family inet filter input local_acl
```

```
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter that selectively blocks Telnet and SSH access:

1. Create the stateless firewall filter **local_acl**.

```
[edit]
user@myhost# edit firewall family inet filter local_acl
```

2. Define the filter term **terminal_access**.

```
[edit firewall family inet filter local_acl]
user@myhost# set term terminal_access from address 192.168.1.0/24
user@myhost# set term terminal_access from protocol tcp
user@myhost# set term terminal_access from port ssh
user@myhost# set term terminal_access from port telnet
user@myhost# set term terminal_access then accept
```

3. Define the filter term **terminal_access_denied**.

```
[edit firewall family inet filter local_acl]
user@myhost# set term terminal_access_denied from protocol tcp
user@myhost# set term terminal_access_denied from port ssh
user@myhost# set term terminal_access_denied from port telnet
user@myhost# set term terminal_access_denied then log
user@myhost# set term terminal_access_denied then reject
user@myhost# set term default-term then accept
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

- To apply the firewall filter to the loopback interface:

```
[edit]
user@myhost# set interfaces lo0 unit 0 family inet filter input local_acl
user@myhost# set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@myhost# show firewall
family inet {
   filter local_acl {
      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;
        }
      }
    }
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@myhost# show interfaces
lo0 {
  unit 0 {
    family inet {
      filter {
        input local_acl;
      }
      address 127.0.0.1/32;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@myhost# commit
```

Verification

Confirm that the configuration is working properly.

- Verifying Accepted Packets on page 108
- Verifying Logged and Rejected Packets on page 109

Verifying Accepted Packets

Purpose Verify that the actions of the firewall filter terms are taken.

Action

1. Clear the firewall log on your router or switch.

```
user@myhost> clear firewall log
```

2. From a host at an IP address *within* the 192.168.1.0/24 subnet, use the **ssh *hostname*** command to verify that you can log in to the device using only SSH. This packet should be accepted, and the packet header information for this packet should not be logged in the firewall filter log buffer in the Packet Forwarding Engine.

```
user@host-A> ssh myhost
user@myhosts's password:
--- JUNOS 11.1-20101102.0 built 2010-11-02 04:48:46 UTC

% cli
user@myhost>
```

3. From a host at an IP address *within* the 192.168.1.0/24 subnet, use the **telnet *hostname*** command to verify that you can log in to your router or switch using only Telnet. This packet should be accepted, and the packet header information for this packet should not be logged in the firewall filter log buffer in the Packet Forwarding Engine.

```
user@host-A> telnet myhost
Trying 192.168.249.71...
Connected to myhost-fxp0.acme.net.
Escape character is '^]'.

host (ttyp0)

login: user
Password:

--- JUNOS 11.1-20101102.0 built 2010-11-02 04:48:46 UTC

% cli
user@myhost>
```

4. Use the **show firewall log** command to verify that the routing table on the device does not contain any entries with a source address in the 192.168.1.0/24 subnet.

```
user@myhost> show firewall log
```

Verifying Logged and Rejected Packets

Purpose Verify that the actions of the firewall filter terms are taken.

Action

1. Clear the firewall log on your router or switch.

   ```
   user@myhost> clear firewall log
   ```

2. From a host at an IP address *outside of* the 192.168.1.0/24 subnet, use the **ssh *hostname*** command to verify that you cannot log in to the device using only SSH. This packet should be rejected, and the packet header information for this packet should be logged in the firewall filter log buffer in the Packet Forwarding Engine.

   ```
   user@host-B ssh myhost
   ssh: connect to host sugar port 22: Connection refused
   --- JUNOS 11.1-20101102.0 built 2010-11-02 04:48:46 UTC
   %
   ```

3. From a host at an IP address *outside of* the 192.168.1.0/24 subnet, use the **telnet *hostname*** command to verify that you can log in to the device using only Telnet. This packet should be rejected, and the packet header information for this packet should be logged in the firewall filter log buffer in the PFE.

   ```
   user@host-B> telnet myhost
   Trying 192.168.249.71...
   telnet: connect to address 192.168.187.3: Connection refused
   telnet: Unable to connect to remote host
   %
   ```

4. Use the **show firewall log** command to verify that the routing table on the device does not contain any entries with a source address in the 192.168.1.0/24 subnet.

   ```
   user@myhost> show firewall log

   Time     Filter    Action Interface Protocol Src Addr       Dest Addr
   18:41:25 local_acl R      fxp0.0    TCP      192.168.187.5  192.168.187.1
   18:41:25 local_acl R      fxp0.0    TCP      192.168.187.5  192.168.187.1
   18:41:25 local_acl R      fxp0.0    TCP      192.168.187.5  192.168.187.1
   ...
   18:43:06 local_acl R      fxp0.0    TCP      192.168.187.5  192.168.187.1
   18:43:06 local_acl R      fxp0.0    TCP      192.168.187.5  192.168.187.1
   18:43:06 local_acl R      fxp0.0    TCP      192.168.187.5  192.168.187.1
   ...
   ```

Related Documentation

- Logging of Packet Headers Evaluated by a Firewall Filter Term on page 48
- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources on page 101
- Example: Configuring a Filter to Block TFTP Access on page 110
- Example: Configuring a Filter to Accept OSPF Packets from a Prefix on page 164
- Example: Configuring a Filter to Accept DHCP Packets Based on Address on page 162

Example: Configuring a Filter to Block TFTP Access

- Requirements on page 111
- Overview on page 111

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

By default, to decrease vulnerability to denial-of-service (DoS) attacks, the Junos OS 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 of these addresses and overrides the default RPF-check filter by accepting these packets. In this example, you block Trivial File Transfer Protocol (TFTP) access, logging any attempts to establish TFTP connections.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter tftp_access_control term one from protocol udp
set firewall family inet filter tftp_access_control term one from port tftp
set firewall family inet filter tftp_access_control term one then log
set firewall family inet filter tftp_access_control term one then discard
set interfaces lo0 unit 0 family inet filter input tftp_access_control
set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter that selectively blocks TFTP access:

1. Create the stateless firewall filter **tftp_access_control**.

```
[edit]
user@host# edit firewall family inet filter tftp_access_control
```

2. Specify a match on packets received on UDP port 69.

```
[edit firewall family inet filter tftp_access_control]
user@host# set term one from protocol udp
user@host# set term one from port tftp
```

3. Specify that matched packets be logged to the buffer on the Packet Forwarding Engine and then discarded.

```
[edit firewall family inet filter tftp_access_control]
user@host# set term one then log
user@host# set term one then discard
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the firewall filter to the loopback interface:

-
```
[edit]
user@host# set interfaces lo0 unit 0 family inet filter input tftp_access_control
user@host# set interfaces lo0 unit 0 family inet address 127.0.0.1/32
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter tftp_access_control {
        term one {
            from {
                protocol udp;
                port tftp;
            }
            then {
                log;
                discard;
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
    unit 0 {
        family inet {
            filter {
```

```
            input tftp_access_control;
          }
          address 127.0.0.1/32;
        }
      }
    }
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

Confirm that the configuration is operating properly:

- Verifying Logged and Discarded Packets on page 113

Verifying Logged and Discarded Packets

Purpose Verify that the actions of the firewall filter terms are taken.

Action To

1. Clear the firewall log on your router or switch.

```
user@myhost> clear firewall log
```

2. From another host, send a packet to UDP port **69** on this router or switch.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources on page 101
- Example: Configuring a Filter to Block Telnet and SSH Access on page 106
- Example: Configuring a Filter to Accept OSPF Packets from a Prefix on page 164
- Example: Configuring a Filter to Accept DHCP Packets Based on Address on page 162

Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags

This example shows how to configure a standard stateless firewall filter to accept packets from a trusted source.

- Requirements on page 114
- Overview on page 114
- Configuration on page 114
- Verification on page 116

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a filter that accepts packets with specific IPv6 TCP flags.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet6 filter tcp_filter term 1 from next-header tcp
set firewall family inet6 filter tcp_filter term 1 from tcp-flags syn
set firewall family inet6 filter tcp_filter term 1 then count tcp_syn_pkt
set firewall family inet6 filter tcp_filter term 1 then log
set firewall family inet6 filter tcp_filter term 1 then accept
set interfaces lo0 unit 0 family inet6 filter input tcp_filter
set interfaces lo0 unit 0 family inet6 address ::10.34.1.0/120
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the firewall filter

1. Create the IPv6 stateless firewall filter **tcp_filter**.

   ```
   [edit]
   user@host# edit firewall family inet6 filter tcp_filter
   ```

2. Specify that a packet matches if it is the initial packet in a TCP session and the next header after the IPv6 header is type TCP.

   ```
   [edit firewall family inet6 filter tcp_filter]
   user@host# set term 1 from next-header tcp
   user@host# set term 1 from tcp-flags syn
   ```

3. Specify that matched packets are counted, logged to the buffer on the Packet Forwarding Engine, and accepted.

   ```
   [edit firewall family inet6 filter tcp_filter]
   user@host# set term 1 then count tcp_syn_pkt
   user@host# set term 1 then log
   user@host# set term 1 then accept
   ```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the firewall filter to the loopback interface:

- [edit]
  user@host# **set interfaces lo0 unit 0 family inet6 filter input tcp_filter**
  user@host# **set interfaces lo0 unit 0 family inet6 address ::10.34.1.0/120**

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

   ```
   [edit]
   user@host# show firewall
   family inet6 {
     filter tcp_filter {
       term 1 {
         from {
           next-header tcp;
           tcp-flags syn;
         }
         then {
           count tcp_syn_pkt;
           log;
           accept;
         }
       }
     }
   }
   ```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

   ```
   [edit]
   user@host# show interfaces
   lo0 {
     unit 0 {
       family inet6 {
         filter {
           input tcp_filter;
         }
         address ::10.34.1.0/120;
       }
     }
   }
   ```

3. When you are done configuring the device, commit your candidate configuration.

   ```
   [edit]
   user@host# commit
   ```

Verification

To confirm that the configuration is working properly, enter the **show firewall** operational mode command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods on page 128
- Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers on page 122

Example: Filtering Packets Received on an Interface Set

This example shows how to configure a standard stateless firewall filter to match packets tagged for a particular interface set.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you apply a stateless firewall filter to the input of the router or switch loopback interface. The firewall filter includes a term that matches packets tagged for a particular interface set.

Topology

You create the firewall filter **L2_filter** to apply rate limits to the protocol-independent traffic received on the following interfaces:

- **fe-0/0/0.0**
- **fe-1/0/0.0**
- **fe-1/1/0.0**

NOTE: **The interface type in this topic is just an example. The fe- interface type is not supported by EX Series switches.**

First, for protocol-independent traffic received on **fe-0/0/0.0**, the firewall filter term **t1** applies policer **p1**.

For protocol-independent traffic received on any other Fast Ethernet interfaces, firewall filter term **t2** applies policer **p2**. To define an interface set that consists of all Fast Ethernet interfaces, you include the **interface-set** ***interface-set-name interface-name*** statement at the **[edit firewall]** hierarchy level. To define a packet-matching criteria based on the interface on which a packet arrives to a specified interface set, you configure a term that uses the **interface-set** firewall filter match condition.

Finally, for any other protocol-independent traffic, firewall filter term **t3** applies policer **p3**.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

- Configuring the Interfaces for Which the Stateless Firewall Filter Terms Take Rate-Limiting Actions on page 118
- Configuring the Stateless Firewall Filter That Rate-Limits Protocol-Independent Traffic Based on the Interfaces on Which Packets Arrive on page 119
- Applying the Stateless Firewall Filter to the Routing Engine Input Interface on page 121

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set interfaces fe-0/0/0 unit 0 family inet address 10.1.1.1/30
set interfaces fe-1/0/0 unit 0 family inet address 10.2.2.1/30
set interfaces fe-1/1/0 unit 0 family inet address 10.4.4.1/30
set firewall policer p1 if-exceeding bandwidth-limit 5m
set firewall policer p1 if-exceeding burst-size-limit 10m
set firewall policer p1 then discard
set firewall policer p2 if-exceeding bandwidth-limit 40m
set firewall policer p2 if-exceeding burst-size-limit 100m
set firewall policer p2 then discard
set firewall policer p3 if-exceeding bandwidth-limit 600m
set firewall policer p3 if-exceeding burst-size-limit 1g
set firewall policer p3 then discard
set firewall interface-set ifset fe-*
set firewall family any filter L2_filter term t1 from interface fe-0/0/0.0
set firewall family any filter L2_filter term t1 then count c1
set firewall family any filter L2_filter term t1 then policer p1
set firewall family any filter L2_filter term t2 from interface-set ifset
set firewall family any filter L2_filter term t2 then count c2
set firewall family any filter L2_filter term t2 then policer p2
set firewall family any filter L2_filter term t3 then count c3
set firewall family any filter L2_filter term t3 then policer p3
set interfaces lo0 unit 0 family inet address 1.1.1.157/30
set interfaces lo0 unit 0 filter input L2_filter
```

Configuring the Interfaces for Which the Stateless Firewall Filter Terms Take Rate-Limiting Actions

Step-by-Step Procedure

To configure the interfaces for which the stateless firewall filter terms take rate-limiting actions:

1. Configure the logical interface whose input traffic will be matched by the first term of the firewall filter.

   ```
   [edit]
   user@host# set interfaces fe-0/0/0 unit 0 family inet address 10.1.1.1/30
   ```

2. Configure the logical interfaces whose input traffic will be matched by the second term of the firewall filter.

   ```
   [edit ]
   user@host# set interfaces fe-1/0/0 unit 0 family inet address 10.2.2.1/30
   user@host# set interfaces fe-1/1/0 unit 0 family inet address 10.4.4.1/30
   ```

3. If you are done configuring the device, commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

Results

Confirm the configuration of the router (or switch) transit interfaces by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
fe-0/0/0 {
  unit 0 {
    family inet {
      address 10.1.1.1/30;
    }
  }
}
fe-1/0/0 {
  unit 0 {
    family inet {
      address 10.2.2.1/30;
    }
  }
}
fe-1/1/0 {
  unit 0 {
    family inet {
      address 10.4.4.1/30;
    }
  }
}
```

Configuring the Stateless Firewall Filter That Rate-Limits Protocol-Independent Traffic Based on the Interfaces on Which Packets Arrive

Step-by-Step Procedure

To configure the standard stateless firewall **L2_filter** that uses policers (**p1**, **p2**, and **p3**) to rate-limit protocol-independent traffic based on the interfaces on which the packets arrive:

1. Configure the firewall statements.

   ```
   [edit]
   user@host# edit firewall
   ```

2. Configure the policer **p1** to discard traffic that exceeds a traffic rate of **5m** bps or a burst size of **10m** bytes.

   ```
   [edit firewall]
   user@host# set policer p1 if-exceeding bandwidth-limit 5m
   user@host# set policer p1 if-exceeding burst-size-limit 10m
   user@host# set policer p1 then discard
   ```

3. Configure the policer **p2** to discard traffic that exceeds a traffic rate of **40m** bps or a burst size of **100m** bytes .

   ```
   [edit firewall]
   user@host# set policer p2 if-exceeding bandwidth-limit 40m
   user@host# set policer p2 if-exceeding burst-size-limit 100m
   user@host# set policer p2 then discard
   ```

4. Configure the policer **p3** to discard traffic that exceeds a traffic rate of **600m** bps or a burst size of **1g** bytes.

   ```
   [edit firewall]
   user@host# set policer p3 if-exceeding bandwidth-limit 600m
   user@host# set policer p3 if-exceeding burst-size-limit 1g
   user@host# set policer p3 then discard
   ```

5. Define the interface set **ifset** to be the group of all Fast Ethernet interfaces on the router.

   ```
   [edit firewall]
   user@host# set interface-set ifset fe-*
   ```

6. Create the stateless firewall filter **L2_filter**.

   ```
   [edit firewall]
   user@host# edit family any filter L2_filter
   ```

7. Configure filter term **t1** to match IPv4, IPv6, or MPLS packets received on interface **fe-0/0/0.0** and use policer **p1** to rate-limit that traffic.

   ```
   [edit firewall family any filter L2_filter]
   user@host# set term t1 from interface fe-0/0/0.0
   user@host# set term t1 then count c1
   user@host# set term t1 then policer p1
   ```

8. Configure filter term **t2** to match packets received on interface-set **ifset** and use policer **p2** to rate-limit that traffic.

   ```
   [edit firewall family any filter L2_filter]
   ```

```
user@host# set term t2 from interface-set ifset
user@host# set term t2 then count c2
user@host# set term t2 then policer p2
```

9. Configure filter term **t3** to use policer **p3** to rate-limit all other traffic.

```
[edit firewall family any filter L2_filter]
user@host# set term t3 then count c3
user@host# set term t3 then policer p3
```

10. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Results Confirm the configuration of the stateless firewall filter and the policers referenced as firewall filter actions by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family any {
    filter L2_filter {
        term t1 {
            from {
                interface fe-0/0/0.0;
            }
            then {
                policer p1;
                count c1;
            }
        }
        term t2 {
            from {
                interface-set ifset;
            }
            then {
                policer p2;
                count c2;
            }
        }
        term t3 {
            then {
                policer p3;
                count c3;
            }
        }
    }
}
policer p1 {
    if-exceeding {
        bandwidth-limit 5m;
        burst-size-limit 10m;
    }
    then discard;
```



```
}
policer p2 {
  if-exceeding {
    bandwidth-limit 40m;
    burst-size-limit 100m;
  }
  then discard;
}
policer p3 {
  if-exceeding {
    bandwidth-limit 600m;
    burst-size-limit 1g;
  }
  then discard;
}
interface-set ifset {
  fe-*;
}
```

Applying the Stateless Firewall Filter to the Routing Engine Input Interface

Step-by-Step Procedure

To apply the stateless firewall filter to the Routing Engine input interface:

1. Apply the stateless firewall filter to the Routing Engine interface in the input direction.

```
[edit]
user@host# set interfaces lo0 unit 0 family inet address 1.1.1.157/30
user@host# set interfaces lo0 unit 0 filter input L2_filter
```

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Results

Confirm the application of the firewall filter to the Routing Engine input interface by entering the **show interfaces** command again. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
user@host# show interfaces
fe-0/0/0 {
  ...
}
fe-1/0/0 {
  ...
}
fe-1/1/0 {
  ...
}
lo0 {
  unit 0 {
    filter {
      input L2_filter;
    }
    family inet {
      address 1.1.1.157/30;
```

```
    }
  }
}
```

Verification

To confirm that the configuration is working properly, use the **show firewall filter L2_filter** operational mode command to monitor traffic statistics about the firewall filter and three counters.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Filtering Packets Received on an Interface Set Overview on page 63
- Statement Hierarchy for Defining an Interface Set on page 273
- Statement Hierarchy for Configuring a Filter to Match on an Interface Set on page 273

Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers

This example shows how to configure a standard stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except from specified BGP peers.

Requirements

No special configuration beyond device initialization is required before you configure this example.

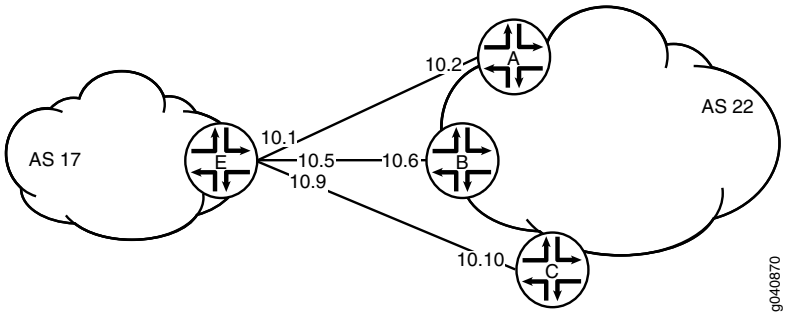
Overview

In this example, you create a stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except the specified BGP peers.

The stateless firewall filter **filter_bgp179** matches all packets from the directly connected interfaces on Device A and Device B to the destination port number 179.

Figure 5 on page 123 shows the topology used in this example. Device C attempts to make a TCP connection to Device E. Device E blocks the connection attempt. This example shows the configuration on Device E.

Figure 5: Typical Network with BGP Peer Sessions



Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Device C

```
set interfaces ge-1/2/0 unit 10 description to-E
set interfaces ge-1/2/0 unit 10 family inet address 10.10.10.10/30
set protocols bgp group external-peers type external
set protocols bgp group external-peers peer-as 17
set protocols bgp group external-peers neighbor 10.10.10.9
set routing-options autonomous-system 22
```

Device E

```
set interfaces ge-1/2/0 unit 0 description to-A
set interfaces ge-1/2/0 unit 0 family inet address 10.10.10.1/30
set interfaces ge-1/2/1 unit 5 description to-B
set interfaces ge-1/2/1 unit 5 family inet address 10.10.10.5/30
set interfaces ge-1/0/0 unit 9 description to-C
set interfaces ge-1/0/0 unit 9 family inet address 10.10.10.9/30
set interfaces lo0 unit 2 family inet filter input filter_bgp179
set interfaces lo0 unit 2 family inet address 192.168.0.1/32
set protocols bgp group external-peers type external
set protocols bgp group external-peers peer-as 22
set protocols bgp group external-peers neighbor 10.10.10.2
set protocols bgp group external-peers neighbor 10.10.10.6
set protocols bgp group external-peers neighbor 10.10.10.10
set routing-options autonomous-system 17
set firewall family inet filter filter_bgp179 term 1 from source-address 10.10.10.2/32
set firewall family inet filter filter_bgp179 term 1 from source-address 10.10.10.6/32
set firewall family inet filter filter_bgp179 term 1 from destination-port bgp
set firewall family inet filter filter_bgp179 term 1 then accept
set firewall family inet filter filter_bgp179 term 2 then reject
```

Configuring Device E

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure Device E with a stateless firewall filter that blocks all TCP connection attempts to port 179 from all requestors except specified BGP peers:

1. Configure the interfaces.

```
user@E# set interfaces ge-1/2/0 unit 0 description to-A
user@E# set interfaces ge-1/2/0 unit 0 family inet address 10.10.10.1/30

user@E# set interfaces ge-1/2/1 unit 5 description to-B
user@E# set interfaces ge-1/2/1 unit 5 family inet address 10.10.10.5/30

user@E# set interfaces ge-1/0/0 unit 9 description to-C
user@E# set interfaces ge-1/0/0 unit 9 family inet address 10.10.10.9/30
```

2. Configure BGP.

```
[edit protocols bgp group external-peers]
user@E# set type external
user@E# set peer-as 22
user@E# set neighbor 10.10.10.2
user@E# set neighbor 10.10.10.6
user@E# set neighbor 10.10.10.10
```

3. Configure the autonomous system number.

```
[edit routing-options]
user@E# set autonomous-system 17
```

4. Define the filter term that accepts TCP connection attempts to port 179 from the specified BGP peers.

```
[edit firewall family inet filter filter_bgp179]
user@E# set term 1 from source-address 10.10.10.2/32
user@E# set term 1 from source-address 10.10.10.6/32
user@E# set term 1 from destination-port bgp
user@E# set term 1 then accept
```

5. Define the other filter term to reject packets from other sources.

```
[edit firewall family inet filter filter_bgp179]
user@E# set term 2 then reject
```

6. Apply the firewall filter to the loopback interface.

```
[edit interfaces lo0 unit 2 family inet]
user@E# set filter input filter_bgp179
user@E# set address 192.168.0.1/32
```

Results

From configuration mode, confirm your configuration by entering the **show firewall**, **show interfaces**, **show protocols**, and **show routing-options** commands. If the output does not

display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@E# show firewall
family inet {
    filter filter_bgp179 {
        term 1 {
            from {
                source-address {
                    10.10.10.2/32;
                    10.10.10.6/32;
                }
                destination-port bgp;
            }
            then accept;
        }
        term 2 {
            then {
                reject;
            }
        }
    }
}
user@E# show interfaces
lo0 {
    unit 2 {
        family inet {
            filter {
                input filter_bgp179;
            }
            address 192.168.0.1/32;
        }
    }
}
ge-1/2/0 {
    unit 0 {
        description to-A;
        family inet {
            address 10.10.10.1/30;
        }
    }
}
ge-1/2/1 {
    unit 5 {
        description to-B;
        family inet {
            address 10.10.10.5/30;
        }
    }
}
ge-1/0/0 {
    unit 9 {
        description to-C;
        family inet {
            address 10.10.10.9/30;
```

```
    }
  }
}

user@E# show protocols
bgp {
  group external-peers {
    type external;
    peer-as 22;
    neighbor 10.10.10.2;
    neighbor 10.10.10.6;
    neighbor 10.10.10.10;
  }
}

user@E# show routing-options
autonomous-system 17;
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

Confirm that the configuration is working properly.

- Verifying That the Filter Is Configured on page 126
- Verifying the TCP Connections on page 126
- Monitoring Traffic on the Interfaces on page 127

Verifying That the Filter Is Configured

Purpose Make sure that the filter is listed in output of the **show firewall filter** command.

Action

```
user@E> show firewall filter filter_bgp179
Filter: filter_bgp179
```

Verifying the TCP Connections

Purpose Verify the TCP connections.

Action From operational mode, run the **show system connections extensive** command on Device C and Device E.

The output on Device C shows the attempt to establish a TCP connection. The output on Device E shows that connections are established with Device A and Device B only.

```
user@C> show system connections extensive | match 10.10.10

tcp4       0      0  10.10.10.9.51872     10.10.10.10.179        SYN_SENT

user@E> show system connections extensive | match 10.10.10

tcp4       0      0  10.10.10.5.179       10.10.10.6.62096       ESTABLISHED
tcp4       0      0  10.10.10.6.62096     10.10.10.5.179         ESTABLISHED
```

```
tcp4       0      0  10.10.10.1.179         10.10.10.2.61506      ESTABLISHED
tcp4       0      0  10.10.10.2.61506       10.10.10.1.179        ESTABLISHED
```

Monitoring Traffic on the Interfaces

Purpose Use the **monitor traffic** command to compare the traffic on an interface that establishes a TCP connection with the traffic on an interface that does not establish a TCP connection.

Action From operational mode, run the **monitor traffic** command on the Device E interface to Device B and on the Device E interface to Device C. The following sample output verifies that in the first example, acknowledgment (**ack**) messages are received. In the second example, **ack** messages are not received.

```
user@E> monitor traffic size 1500 interface ge-1/2/1.5
19:02:49.700912 Out IP 10.10.10.5.bgp > 10.10.10.6.62096: P
3330573561:3330573580(19) ack 915601686 win 16384 <nop,nop,timestamp 1869518816
1869504850>: BGP, length: 19
19:02:49.801244  In IP 10.10.10.6.62096 > 10.10.10.5.bgp: . ack 19 win 16384
<nop,nop,timestamp 1869518916 1869518816>
19:03:03.323018  In IP 10.10.10.6.62096 > 10.10.10.5.bgp: P 1:20(19) ack 19 win
16384 <nop,nop,timestamp 1869532439 1869518816>: BGP, length: 19
19:03:03.422418 Out IP 10.10.10.5.bgp > 10.10.10.6.62096: . ack 20 win 16384
<nop,nop,timestamp 1869532539 1869532439>
19:03:17.220162 Out IP 10.10.10.5.bgp > 10.10.10.6.62096: P 19:38(19) ack 20 win
 16384 <nop,nop,timestamp 1869546338 1869532439>: BGP, length: 19
19:03:17.320501  In IP 10.10.10.6.62096 > 10.10.10.5.bgp: . ack 38 win 16384
<nop,nop,timestamp 1869546438 1869546338>

user@E> monitor traffic size 1500 interface ge-1/0/0.9

18:54:20.175471 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0)
 win 16384 <mss 1460,nop,wscale 0,nop,nop,timestamp 1869009240 0,sackOK,eol>
18:54:23.174422 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0)
 win 16384 <mss 1460,nop,wscale 0,nop,nop,timestamp 1869012240 0,sackOK,eol>
18:54:26.374118 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0)
 win 16384 <mss 1460,nop,wscale 0,nop,nop,timestamp 1869015440 0,sackOK,eol>
18:54:29.573799 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0)
 win 16384 <mss 1460,sackOK,eol>
18:54:32.773493 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0)
 win 16384 <mss 1460,sackOK,eol>
18:54:35.973185 Out IP 10.10.10.9.61335 > 10.10.10.10.bgp: S 573929123:573929123(0)
 win 16384 <mss 1460,sackOK,eol>
```

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods on page 128
- Example: Configuring a Filter to Accept Packets Based on IPv6 TCP Flags on page 113

Example: Configuring a Stateless Firewall Filter to Protect Against TCP and ICMP Floods

This example shows how to create a stateless firewall filter that protects against TCP and ICMP denial-of-service attacks.

Requirements

No special configuration beyond device initialization is required before configuring stateless firewall filters.

Overview

In this example, you create a stateless firewall filter called **protect-RE** that polices TCP and ICMP packets. This example includes the following policers:

- **tcp-connection-policer**—Limits the traffic rate of the TCP packets to 500,000 bps and the burst size to 15,000 bytes. Packets that exceed the traffic rate are discarded.
- **icmp-policer**—Limits the traffic rate of the ICMP packets to 1,000,000 bps and the burst size to 15,000 bytes. Packets that exceed the traffic rate are discarded.

When specifying limits, the bandwidth limit can be from 32,000 bps to 32,000,000,000 bps and the burst-size limit can be from 1,500 bytes through 100,000,000 bytes. Use the following abbreviations when specifying limits: k (1,000), m (1,000,000), and g (1,000,000,000).

Each policer is incorporated into the action of a filter term. This example includes the following terms:

- **tcp-connection-term**—Polices certain TCP packets with a source address of 192.168.0.0/24 or 10.0.0.0/24. These addresses are defined in the **trusted-addresses** prefix list.

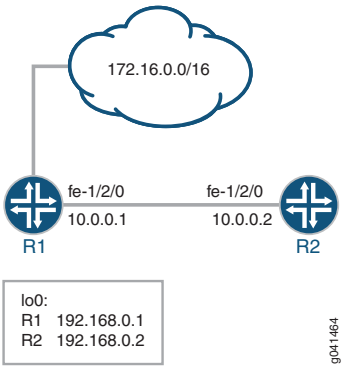
  Filtered packets include **tcp-established** packets The **tcp-established** match condition is an alias for the bit-field match condition **tcp-flags "(ack | rst)"**, which indicates an established TCP session, but not the first packet of a TCP connection.
- **icmp-term**—Polices ICMP packets. All ICMP packets are counted in the **icmp-counter** counter.

NOTE: You can move terms within the firewall filter by using the **insert** command. See *insert* in the *CLI User Guide*.

You can apply a stateless firewall to the input or output sides, or both, of an interface. To filter packets transiting the device, apply the firewall filter to any non-Routing Engine interface. To filter packets originating from, or destined for, the Routing Engine, apply the firewall filter to the loopback (lo0) interface.

Figure 6 on page 129 shows the sample network.

Figure 6: Firewall Filter to Protect Against TCP and ICMP Floods



Because this firewall filter limits Routing Engine traffic to TCP packets, routing protocols that use other transport protocols for Layer 4 cannot successfully establish sessions when this filter is active. To demonstrate, this example sets up OSPF between Device R1 and Device R2.

"CLI Quick Configuration" on page 129 shows the configuration for all of the devices in Figure 6 on page 129.

The section "Step-by-Step Procedure" on page 130 describes the steps on Device R2.

Configuration

CLI Quick Configuration

To quickly configure the stateless firewall filter, copy the following commands to a text file, remove any line breaks, and then paste the commands into the CLI.

Device R1

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.1/30
set interfaces lo0 unit 0 family inet address 192.168.0.1/32 primary
set interfaces lo0 unit 0 family inet address 172.16.0.1/32
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext peer-as 200
set protocols bgp group ext neighbor 10.0.0.2
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 192.168.0.1
set routing-options autonomous-system 100
```

Device R2

```
set interfaces fe-1/2/0 unit 0 family inet address 10.0.0.2/30
set interfaces lo0 unit 0 family inet filter input protect-RE
set interfaces lo0 unit 0 family inet address 192.168.0.2/32 primary
set interfaces lo0 unit 0 family inet address 172.16.0.2/32
```

```
set protocols bgp group ext type external
set protocols bgp group ext export send-direct
set protocols bgp group ext neighbor 10.0.0.1 peer-as 100
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface fe-1/2/0.0
set policy-options prefix-list trusted-addresses 10.0.0.0/24
set policy-options prefix-list trusted-addresses 192.168.0.0/24
set policy-options policy-statement send-direct term 1 from protocol direct
set policy-options policy-statement send-direct term 1 then accept
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 200
set firewall family inet filter protect-RE term tcp-connection-term from source-prefix-list
    trusted-addresses
set firewall family inet filter protect-RE term tcp-connection-term from protocol tcp
set firewall family inet filter protect-RE term tcp-connection-term from tcp-established
set firewall family inet filter protect-RE term tcp-connection-term then policer
    tcp-connection-policer
set firewall family inet filter protect-RE term tcp-connection-term then accept
set firewall family inet filter protect-RE term icmp-term from source-prefix-list
    trusted-addresses
set firewall family inet filter protect-RE term icmp-term from protocol icmp
set firewall family inet filter protect-RE term icmp-term then policer icmp-policer
set firewall family inet filter protect-RE term icmp-term then count icmp-counter
set firewall family inet filter protect-RE term icmp-term then accept
set firewall policer tcp-connection-policer filter-specific
set firewall policer tcp-connection-policer if-exceeding bandwidth-limit 1m
set firewall policer tcp-connection-policer if-exceeding burst-size-limit 15k
set firewall policer tcp-connection-policer then discard
set firewall policer icmp-policer filter-specific
set firewall policer icmp-policer if-exceeding bandwidth-limit 1m
set firewall policer icmp-policer if-exceeding burst-size-limit 15k
set firewall policer icmp-policer then discard
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure stateless firewall filter policers:

1. Configure the device interfaces.

```
[edit interfaces fe-1/2/0 unit 0 family inet ]
user@R2# set address 10.0.0.2/30

[edit interfaces lo0 unit 0 family inet]
user@R2# set address 192.168.0.2/32 primary
user@R2# set address 172.16.0.2/32
```

2. Configure the BGP peering session.

```
[edit protocols bgp group ext]
user@R2# set type external
user@R2# set export send-direct
user@R2# set neighbor 10.0.0.1 peer-as 100
```

3. Configure the autonomous system (AS) number and router ID.


```
[edit routing-options]
user@R2# set autonomous-system 200
user@R2# set router-id 192.168.0.2
```

4. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@R2# set interface lo0.0 passive
user@R2# set interface fe-1/2/0.0
```

5. Define the list of trusted addresses.

```
[edit policy-options prefix-list trusted-addresses]
user@R2# set 10.0.0.0/24
user@R2# set 192.168.0.0/24
```

6. Configure a policy to advertise direct routes.

```
[edit policy-options policy-statement send-direct term 1]
user@R2# set from protocol direct
user@R2# set then accept
```

7. Configure the TCP policer.

```
[edit firewall policer tcp-connection-policer]
user@R2# set filter-specific
user@R2# set if-exceeding bandwidth-limit 1m
user@R2# set if-exceeding burst-size-limit 15k
user@R2# set then discard
```

8. Create the ICMP policer.

```
[edit firewall policer icmp-policer]
user@R2# set filter-specific
user@R2# set if-exceeding bandwidth-limit 1m
user@R2# set if-exceeding burst-size-limit 15k
user@R2# set then discard
```

9. Configure the TCP filter rules.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# set from source-prefix-list trusted-addresses
user@R2# set from protocol tcp
user@R2# set from tcp-established
user@R2# set then policer tcp-connection-policer
user@R2# set then accept
```

10. Configure the ICMP filter rules.

```
[edit firewall family inet filter protect-RE term icmp-term]
user@R2# set from source-prefix-list trusted-addresses
user@R2# set from protocol icmp
user@R2# set then policer icmp-policer
user@R2# set then count icmp-counter
user@R2# set then accept
```

11. Apply the filter to the loopback interface.

```
[edit interfaces lo0 unit 0]
user@R2# set family inet filter input protect-RE
```

Results Confirm your configuration by entering the **show interfaces**, **show protocols**, **show policy-options**, **show routing-options**, and **show firewall** commands from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@R2# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            address 10.0.0.2/30;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            filter {
                input protect-RE;
            }
            address 192.168.0.2/32 {
                primary;
            }
            address 172.16.0.2/32;
        }
    }
}
```

```
user@R2# show protocols
bgp {
    group ext {
        type external;
        export send-direct;
        neighbor 10.0.0.1 {
            peer-as 100;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface fe-1/2/0.0;
    }
}
```

```
user@R2# show policy-options
prefix-list trusted-addresses {
    10.0.0.0/24;
    192.168.0.0/24;
}
policy-statement send-direct {
    term 1 {
        from protocol direct;
        then accept;
    }
```

```
}

user@R2# show routing-options
router-id 192.168.0.2;
autonomous-system 200;

user@R2# show firewall
family inet {
   filter protect-RE {
      term tcp-connection-term {
         from {
            source-prefix-list {
               trusted-addresses;
            }
            protocol tcp;
            tcp-established;
         }
         then {
            policer tcp-connection-policer;
            accept;
         }
      }
      term icmp-term {
         from {
            source-prefix-list {
               trusted-addresses;
            }
            protocol icmp;
         }
         then {
            policer icmp-policer;
            count icmp-counter;
            accept;
         }
      }
   }
}
policer tcp-connection-policer {
   filter-specific;
   if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 15k;
   }
   then discard;
}
policer icmp-policer {
   filter-specific;
   if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 15k;
   }
   then discard;
}
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

Confirm that the configuration is working properly.

NOTE: To verify the TCP policer, you can use a packet generation tool. This task is not shown here.

Displaying Stateless Firewall Filter That Are in Effect

Purpose Verify the configuration of the firewall filter.

Action From operational mode, enter the **show firewall** command.

```
user@R2> show firewall
Filter: protect-RE
Counters:
Name                                                Bytes              Packets
icmp-counter                                            0                    0
Policers:
Name                                                Bytes              Packets
icmp-policer                                                                0
tcp-connection-policer                                                      0
```

Meaning The output shows the filter, the counter, and the policers that are in effect on Device R2.

Using telnet to Verify the tcp-established Condition in the TCP Firewall Filter

Purpose Make sure that telnet traffic works as expected.

Action Verify that the device can establish only TCP sessions with hosts that meet the **from tcp-established** condition..

1. From Device R2, make sure that the BGP session with Device R1 is established.

   ```
   user@R2> show bgp summary | match down
   Groups: 1 Peers: 1 Down peers: 0
   ```

2. From Device R2, telnet to Device R1.

   ```
   user@R2> telnet 192.168.0.1
   Trying 192.168.0.1...
   Connected to R1.acme.net.
   Escape character is '^]'.

   R1 (ttyp4)
   ```

```
login:
```

3. From Device R1, telnet to Device R2.

```
user@R1> telnet 192.168.0.2
Trying 192.168.0.2...
telnet: connect to address 192.168.0.2: Operation timed out
telnet: Unable to connect to remote host
```

4. On Device R2, deactivate the **from tcp-established** match condition.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# deactivate from tcp-established
user@R2# commit
```

5. From Device R1, try again to telnet to Device R2.

```
user@R1> telnet 192.168.0.1
Trying 192.168.0.2...
Connected to R2.acme.net.
Escape character is '^]'.

R2 (ttyp4)

login:
```

Meaning Verify the following information:

- As expected , the BGP session is established. The **from tcp-established** match condition is not expected to block BGP session establishment.
- From Device R2, you can telnet to Device R1. Device R1 has no firewall filter configured, so this is the expected behavior.
- From Device R1, you cannot telnet to Device R2. Telnet uses TCP as the transport protocol, so this result might be surprising. The cause for the lack of telnet connectivity is the **from tcp-established** match condition. This match condition limits the type of TCP traffic that is accepted of Device R2. After this match condition is deactivated, the telnet session is successful.

Using telnet to Verify the Trusted Prefixes Condition in the TCP Firewall Filter

Purpose Make sure that telnet traffic works as expected.

Action Verify that the device can establish only telnet sessions with a host at an IP address that matches one of the trusted source addresses. For example, log in to the device with the **telnet** command from another host with one of the trusted address prefixes. Also, verify that telnet sessions with untrusted source addresses are blocked.

1. From Device R1, telnet to Device R2 from an untrusted source address.

```
user@R1> telnet 172.16.0.2 source 172.16.0.1
Trying 172.16.0.2...
^C
```

2. From Device R2, add 172.16/16 to the list of trusted prefixes.

```
[edit policy-options prefix-list trusted-addresses]
user@R2# set 172.16.0.0/16
user@R2# commit
```

3. From Device R1, try again to telnet to Device R2.

```
user@R1> telnet 172.16.0.2 source 172.16.0.1
Trying 172.16.0.2...
Connected to R2.acme.net.
Escape character is '^]'.

R2 (ttyp4)

login:
```

Meaning Verify the following information:

- From Device R1, you cannot telnet to Device R2 with an unstrusted source address. After the 172.16/16 prefix is added to the list of trusted prefixes, the telnet request from source address 172.16.0.1 is accepted.
- OSPF session establishment is blocked. OSPF does not use TCP as its transport protocol. After the **from protocol tcp** match condition is deactivated, OSPF session establishment is not blocked.

Using OSPF to Verify the TCP Firewall Filter

Purpose Make sure that OSPF traffic works as expected.

Action Verify that the device cannot establish OSPF connectivity.

1. From Device R1, check the OSPF sessions.

```
user@R1> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.2         fe-1/2/0.0             Init      192.168.0.2      128    34
```

2. From Device R2, check the OSPF sessions.

```
user@R2> show ospf neighbor
```

3. From Device R2, remove the **from protocol tcp** match condition.

```
[edit firewall family inet filter protect-RE term tcp-connection-term]
user@R2# deactivate from protocol
user@R2# commit
```

4. From Device R1, recheck the OSPF sessions.

```
user@R1> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.2         fe-1/2/0.0             Full      192.168.0.2      128    36
```

5. From Device R2, recheck the OSPF sessions.

```
user@R2> show ospf neighbor
Address          Interface              State     ID               Pri  Dead
10.0.0.1         fe-1/2/0.0             Full      192.168.0.1      128    39
```

Meaning Verify the following information:

- OSPF session establishment is blocked. OSPF does not use TCP as its transport protocol. After the **from protocol tcp** match condition is deactivated, OSPF session establishment is successful.

Verifying the ICMP Firewall Filter

Purpose Verify that ICMP packets are being policed and counted. Also make sure that ping requests are discarded when the requests originate from an untrusted source address.

Action

1. Undo the configuration changes made in previous verification steps.

   Reactivate the TCP firewall settings, and delete the 172.16/16 trusted source address.

   ```
   [edit firewall family inet filter protect-RE term tcp-connection-term]
   user@R2# activate from protocol
   user@R2# activate from tcp-established

   [edit policy-options prefix-list trusted-addresses]
   user@R2# delete 172.16.0.0/16

   user@R2# commit
   ```

2. From Device R1, ping the loopback interface on Device R2.

   ```
   user@R1> ping 192.168.0.2 rapid count 600 size 2000
   PING 192.168.0.2 (192.168.0.2): 2000 data bytes
   ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
   --- 192.168.0.2 ping statistics ---
   600 packets transmitted, 536 packets received, 10% packet loss
   pinground-trip min/avg/max/stddev = 2.976/3.405/42.380/2.293 ms
   ```

3. From Device R2, check the firewall statistics.

   ```
   user@R2> show firewall

   Filter: protect-RE
   Counters:
   Name                                                Bytes              Packets
   icmp-counter                                      1180804                 1135
   Policers:
   Name                                                Bytes              Packets
   icmp-policer                                                              66
   tcp-connection-policer                                                     0
   ```

4. From an untrusted source address on Device R1, send a ping request to Device R2's loopback interface.

   ```
   user@R1> ping 172.16.0.2 source 172.16.0.1

   PING 172.16.0.2 (172.16.0.2): 56 data bytes
   ^C
   --- 172.16.0.2 ping statistics ---
   14 packets transmitted, 0 packets received, 100% packet loss
   ```

Meaning Verify the following information:

- The ping output shows that 10% packet loss is occurring.
- The ICMP packet counter is incrementing, and the icmp-policer is incrementing.
- Device R2 does not send ICMP responses to the **ping 172.16.0.2 source 172.16.0.1** command.

Related Documentation

- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources on page 101
- *Two-Color Policer Configuration Overview*

CHAPTER 12

Firewall Filters Applied to Transit Traffic

Example: Configuring a Filter to Match on IPv6 Flags

This example shows how to configure a filter to match on IPv6 TCP flags.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you configure a filter to match on IPv6 TCP flags. You can use this example to configure IPv6 TCP flags in M Series, MX Series, and T Series routing devices.

Configuration

Step-by-Step Procedure

To configure a filter to match on IPv6 TCP flags:

1. Include the family statement at the firewall hierarchy level, specifying **inet6** as the protocol family.

   ```
   [edit]
   user@host# edit firewall family inet6
   ```

2. Create the stateless firewall filter.

   ```
   [edit firewall family inet6]
   user@host# edit filter tcpfilt
   ```

3. Define the first term for the filter.

   ```
   [edit firewall family inet6 filter tcpfilt]
   user@host# edit term 1
   ```

4. Define the source address match conditions for the term.

   ```
   [edit firewall family inet6 filter tcpfilt term 1]
   user@host# set from next-header tcp tcp-flags syn
   ```

5. Define the actions for the term.

   ```
   [edit firewall family inet6 filter tcpfilt term 1]
   user@host# set then count tcp_syn_pkt log accept
   ```

6. If you are done configuring the device, commit the configuration.

   ```
   [edit firewall family inet6 filter tcpfilt term 1]
   user@host top

   [edit]
   user@host# commit
   ```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter tcpfilt** command.

Example: Configuring a Filter to Match on Port and Protocol Fields

This example shows how to configure a standard stateless firewall filter to match on destination port and protocol fields.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you configure a stateless firewall filter that accepts all IPv4 packets except for TCP and UDP packets. TCP and UDP packets are accepted if destined for the SSH port or the Telnet port. All other packets are rejected.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

- Configure the Stateless Firewall Filter on page 141
- Apply the Stateless Firewall Filter to a Logical Interface on page 142
- Confirm and Commit Your Candidate Configuration on page 142

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level:

```
set firewall family inet filter filter1 term term1 from protocol-except tcp
set firewall family inet filter filter1 term term1 from protocol-except udp
set firewall family inet filter filter1 term term1 then accept
set firewall family inet filter filter1 term term2 from address 192.168.0.0/16
set firewall family inet filter filter1 term term2 then reject
set firewall family inet filter filter1 term term3 from destination-port ssh
set firewall family inet filter filter1 term term3 from destination-port telnet
set firewall family inet filter filter1 term term3 then accept
set firewall family inet filter filter1 term term4 then reject
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input filter1
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **filter1**:

1. Create the IPv4 stateless firewall filter.

   ```
   [edit]
   user@host# edit firewall family inet filter filter1
   ```

2. Configure a term to accept all traffic except for TCP and UDP packets.

   ```
   [edit firewall family inet filter filter1]
   user@host# set term term1 from protocol-except tcp
   user@host# set term term1 from protocol-except udp
   user@host# set term term1 then accept
   ```

3. Configure a term to reject packets to or from the **192.168/16** prefix.

```
[edit firewall family inet filter filter1]
user@host# set term term2 from address 192.168.0.0/16
user@host# set term term2 then reject
```

4. Configure a term to accept packets destined for either the SSH port or the Telnet port.

```
[edit firewall family inet filter filter1]
user@host# set term term3 from destination-port ssh
user@host# set term term3 from destination-port telnet
user@host# set term term3 then accept
```

5. Configure the last term to reject all packets.

```
[edit firewall family inet filter filter1]
user@host# set term term4 then reject
```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input filter1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter filter1 {
        term term1 {
            from {
                protocol-except [tcp udp];
            }
            then {
                accept;
            }
```

```
    }
    term term2 {
      from {
        address 192.168/16;
      }
      then {
        reject;
      }
    }
    term term3 {
      from {
        destination-port [ssh telnet];
      }
      then {
        accept;
      }
    }
    term term4 {
      then {
        reject;
      }
    }
  }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input filter1;
      }
      address 10.1.2.3/30;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter filter1** operational mode command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Match on IPv6 Flags on page 139

Example: Configuring a Filter to Count Accepted and Rejected Packets

This example shows how to configure a firewall filter to count packets.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to reject all addresses except 192.168.5.0/24.

Topology

In the first term, the match condition **address 192.168.5.0/24 except** causes this address to be considered a mismatch, and this address is passed to the next term in the filter. The match condition **address 0.0.0.0/0** matches all other packets, and these are counted, logged, and rejected.

In the second term, all packets that passed though the first term (that is, packets whose address matches **192.168.5.0/24**) are counted, logged, and accepted.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter fire1 term 1 from address 192.168.5.0/24 except
set firewall family inet filter fire1 term 1 from address 0.0.0.0/0
set firewall family inet filter fire1 term 1 then count reject_pref1_1
set firewall family inet filter fire1 term 1 then log
set firewall family inet filter fire1 term 1 then reject
```

```
set firewall family inet filter fire1 term 2 then count reject_pref1_2
set firewall family inet filter fire1 term 2 then log
set firewall family inet filter fire1 term 2 then accept
set interfaces ge-0/0/1 unit 0 family inet filter input fire1
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **fire1**:

1. Create the stateless firewall filter **fire1**.

   ```
   [edit]
   user@host# edit firewall family inet filter fire1
   ```

2. Configure the first term to reject all addresses except those to or from the **192.168.5.0/24** prefix and then count, log, and reject all other packets.

   ```
   [edit firewall family inet filter fire1]
   user@host# set term 1 from address 192.168.5.0/24 except
   user@host# set term 1 from address 0.0.0.0/0
   user@host# set term 1 then count reject_pref1_1
   user@host# set term 1 then log
   user@host# set term 1 then reject
   ```

3. Configure the next term to count, log, and accept packets in the **192.168.5.0/24** prefix.

   ```
   [edit firewall family inet filter fire1]
   user@host# set term 2 then count reject_pref1_2
   user@host# set term 2 then log
   user@host# set term 2 then accept
   ```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

   ```
   [edit]
   user@host# edit interfaces ge-0/0/1 unit 0 family inet
   ```

2. Configure the interface address for the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set address 10.1.2.3/30
   ```

3. Apply the stateless firewall filter to the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set filter input fire1
   ```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
   filter fire1 {
      term 1 {
         from {
            address {
               192.168.5.0/24 except;
               0.0.0.0/0;
            }
         }
         then {
            count reject_pref1_1;
            log;
            reject;
         }
      }
      term 2 {
         then {
            count reject_pref1_2;
            log;
            accept;
         }
      }
   }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
   unit 0 {
      family inet {
         filter {
            input fire1;
         }
         address 10.1.2.3/30;
      }
   }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter fire1** operational mode command. You can also display the log and individual counters separately by using the following forms of the command:

- **show firewall counter reject_pref1_1**
- **show firewall counter reject_pref1_2**
- **show firewall log**

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Count IP Options Packets on page 150
- Example: Configuring a Filter to Count and Discard IP Options Packets on page 147

Example: Configuring a Filter to Count and Discard IP Options Packets

This example shows how to configure a standard stateless firewall to count packets.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Because the filter term matches on *any* IP option value, the filter term can use the **count** nonterminating action without the **discard** terminating action or (alternatively) without requiring an interface on a 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, or 60-Gigabit Ethernet Enhanced Queuing MPC on an MX Series router.

Overview

In this example, you use a standard stateless firewall filter to count and discard packets that include any IP option value but accept all other packets.

The IP option header field is an optional field in IPv4 headers only. The **ip-options** and **ip-options-except** match conditions are supported for standard stateless firewall filters and service filters only.

NOTE: On M and T series routers, firewall filters cannot count ip-options packets on a per option type and per interface basis. A limited work around is to use the **show pfe statistics ip options** command to see ip-options statistics on a per Packet Forwarding Engine (PFE) basis. See *show pfe statistics ip* for sample output.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

- Configure the Stateless Firewall Filter on page 148
- Apply the Stateless Firewall Filter to a Logical Interface on page 149
- Confirm and Commit Your Candidate Configuration on page 149

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter block_ip_options term 10 from ip-options any
set firewall family inet filter block_ip_options term 10 then count option_any
set firewall family inet filter block_ip_options term 10 then discard
set firewall family inet filter block_ip_options term 999 then accept
set interfaces ge-0/0/1 unit 0 family inet filter input block_ip_options
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter:

1. Create the stateless firewall filter **block_ip_options**.

   ```
   [edit]
   user@host# edit firewall family inet filter block_ip_options
   ```

2. Configure the first term to count and discard packets that include any IP options header fields.

   ```
   [edit firewall family inet filter block_ip_options]
   user@host# set term 10 from ip-options any
   user@host# set term 10 then count option_any
   user@host# set term 10 then discard
   ```

3. Configure the other term to accept all other packets.

   ```
   [edit firewall family inet filter block_ip_options]
   user@host# set term 999 then accept
   ```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input block_ip_options
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter block_ip_options {
        term 10 {
            from {
                ip-options any;
            }
            then {
                count option_any;
                discard;
            }
        }
        term 999 {
            then accept;
        }
    }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input block_ip_options;
```

```
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter block_ip_options** operational mode command. To display the count of discarded packets separately, enter the **show firewall count option_any** form of the command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Count Accepted and Rejected Packets on page 144
- Example: Configuring a Filter to Count IP Options Packets on page 150

Example: Configuring a Filter to Count IP Options Packets

This example shows how use a stateless firewall filter to count individual IP options packets:

Requirements

This example uses an interface on a 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, or 60-Gigabit Ethernet Enhanced Queuing MPC on an MX Series router. This interface enables you to apply an IPv4 firewall filter (standard or service filter) that can use the **count**, **log**, and **syslog** nonterminating actions on packets that match a *specific* **ip-option** value without having to also use the **discard** terminating action.

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to count IP options packets but not block any traffic. Also, the filter logs packets that have loose or strict source routing.

The IP option header field is an optional field in IPv4 headers only. The **ip-options** and **ip-options-except** match conditions are supported for standard stateless firewall filters and service filters only.

NOTE: On M and T series routers, firewall filters cannot count **ip-options** packets on a per option type and per interface basis. A limited work around is to use the **show pfe statistics ip options** command to see **ip-options** statistics on a per Packet Forwarding Engine (PFE) basis. See *show pfe statistics ip* for sample output.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

- Configure the Stateless Firewall Filter on page 152
- Apply the Stateless Firewall Filter to a Logical Interface on page 153
- Confirm and Commit Your Candidate Configuration on page 153

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter ip_options_filter term match_strict_source from ip-options
    strict-source-route
set firewall family inet filter ip_options_filter term match_strict_source then count
    strict_source_route
set firewall family inet filter ip_options_filter term match_strict_source then log
set firewall family inet filter ip_options_filter term match_strict_source then accept
set firewall family inet filter ip_options_filter term match_loose_source from ip-options
    loose-source-route
set firewall family inet filter ip_options_filter term match_loose_source then count
    loose_source_route
set firewall family inet filter ip_options_filter term match_loose_source then log
set firewall family inet filter ip_options_filter term match_loose_source then accept
set firewall family inet filter ip_options_filter term match_record from ip-options
    record-route
set firewall family inet filter ip_options_filter term match_record then count record_route
set firewall family inet filter ip_options_filter term match_record then accept
set firewall family inet filter ip_options_filter term match_timestamp from ip-options
    timestamp
set firewall family inet filter ip_options_filter term match_timestamp then count timestamp
set firewall family inet filter ip_options_filter term match_timestamp then accept
set firewall family inet filter ip_options_filter term match_router_alert from ip-options
    router-alert
set firewall family inet filter ip_options_filter term match_router_alert then count
    router_alert
set firewall family inet filter ip_options_filter term match_router_alert then accept
set firewall family inet filter ip_options_filter term match_all then accept
```

```
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input ip_options_filter
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **ip_option_filter**:

1. Create the stateless firewall filter **ip_option_filter**.

   ```
   [edit]
   user@host# edit firewall family inet filter ip_options_filter
   ```

2. Configure the first term to count, log, and accept packets with the **strict_source_route** IP optional header field.

   ```
   [edit firewall family inet filter ip_option_filter]
   user@host# set term match_strict_source from ip-options strict_source_route
   user@host# set term match_strict_source then count strict_source_route
   user@host# set term match_strict_source then log
   user@host# set term match_strict_source then accept
   ```

3. Configure the next term to count, log, and accept packets with the **loose-source-route** IP optional header field.

   ```
   [edit firewall family inet filter ip_option_filter]
   user@host# set term match_loose_source from ip-options loose-source-route
   user@host# set term match_loose_source then count loose_source_route
   user@host# set term match_loose_source then log
   user@host# set term match_loose_source then accept
   ```

4. Configure the next term to count and accept packets with the **record-route** IP optional header field.

   ```
   [edit firewall family inet filter ip_option_filter]
   user@host# set term match_record from ip-options record-route
   user@host# set term match_record then count record_route
   user@host# set term match_record then accept
   ```

5. Configure the next term to count and accept packets with the **timestamp** IP optional header field.

   ```
   [edit firewall family inet filter ip_option_filter]
   user@host# set term match_timestamp from ip-options timestamp
   user@host# set term match_timestamp then count timestamp
   user@host# set term match_timestamp then accept
   ```

6. Configure the next term to count and accept packets with the **router-alert** IP optional header field.

   ```
   [edit firewall family inet filter ip_option_filter]
   user@host# set term match_router_alert from ip-options router-alert
   user@host# set term match_router_alert then count router_alert
   user@host# set term match_router_alert then accept
   ```

7. Create the last term to accept any packet without incrementing any counters.

   ```
   [edit firewall family inet filter ip_option_filter]
   user@host# set term match_all then accept
   ```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input ip_options_filter
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter ip_options_filter {
        term match_strict_source {
            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 match_record {
            from {
                ip-options record-route;
            }
            then {
```

```
            count record_route;
            accept;
          }
        }
        term match_timestamp {
          from {
            ip-options timestamp;
          }
          then {
            count timestamp;
            accept;
          }
        }
        term match_router_alert {
          from {
            ip-options router-alert;
          }
          then {
            count router_alert;
            accept;
          }
        }
        term match_all {
          then accept;
        }
      }
    }
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input ip_option_filter;
      }
      address 10.1.2.3/30;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter ip_option_filter** operational mode command. You can also display the log and individual counters separately by using the following forms of the command:

- **show firewall counter strict_source_route**
- **show firewall counter loose_source_route**
- **show firewall counter record_route**
- **show firewall counter timestamp**
- **show firewall counter router_alert**
- **show firewall log**

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Count Accepted and Rejected Packets on page 144
- Example: Configuring a Filter to Count and Discard IP Options Packets on page 147

Example: Configuring a Filter to Count and Sample Accepted Packets

This example shows how to configure a standard stateless firewall filter to count and sample accepted packets.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you begin, configure traffic sampling by including the **sampling** statement at the **[edit forwarding-options]** hierarchy level.

Overview

In this example, you use a standard stateless firewall filter to count and sample all packets received on a logical interface.

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.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter sam term all then count count_sam
set firewall family inet filter sam term all then sample
set interfaces at-2/0/0 unit 301 family inet address 10.1.2.3/30
set interfaces at-2/0/0 unit 301 family inet filter input sam
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **sam**:

1. Create the stateless firewall filter **sam**.

   ```
   [edit]
   user@host# edit firewall family inet filter sam
   ```

2. Configure the term to count and sample all packets.

   ```
   [edit firewall family inet filter sam]
   user@host# set term all then count count_sam
   user@host# set term all then sample
   ```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

   ```
   [edit]
   user@host# edit interfaces ge-0/0/1 unit 0 family inet
   ```

2. Configure the interface address for the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set address 10.1.2.3/30
   ```

3. Apply the stateless firewall filter to the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set filter input sam
   ```

NOTE: The Junos OS does not sample packets originating from the router or switch. If you configure a 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.

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
   filter sam {
      term all {
         then {
            count count_sam;
            sample; # default action is accept
         }
      }
   }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
interfaces {
   at-2/0/0 {
      unit 301 {
         family inet {
            filter {
               input sam;
            }
            address 10.1.2.3/30;
         }
      }
   }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

Confirm that the configuration is working properly.

Displaying the Packet Counter

Purpose Verify that the firewall filter is evaluating packets.

Action

```
user@host> show firewall filter sam
Filter:
Counters:
Name                Bytes               Packets
sam
sam-1               98                  8028
```

Displaying the Firewall Filter Log Output

Purpose Display the packet header information for all packets evaluated by the firewall filter.

Action

```
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
```

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

- **Pro**—Packet's protocol name or number.
- **Source address**—Source IP address in the packet.
- **Destination address**—Destination IP address in the packet.

Displaying the Sampling Output

Purpose Verify that the sampling output contains appropriate data.

Action

```
  wtmp.0.gz            Size: 15017, Last changed: Dec 19 13:15:54 wtmp.1.gz
       Size: 493, Last changed: Nov 19 13:47:29
  wtmp.2.gz            Size: 57, Last changed: Oct 20 15:24:34
  |                    Pipe through a command

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
```

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Set the DSCP Bit to Zero on page 249

Example: Configuring a Filter to Match on Two Unrelated Criteria

This example shows how to configure a standard stateless firewall filter to match on two unrelated criteria.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a standard stateless firewall filter to match IPv4 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.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

- Configuring the IPv4 Firewall Filter on page 160
- Applying the IPv4 Firewall Filter to a Logical Interface on page 161

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter ospf_or_131 term protocol_match from protocol ospf
set firewall family inet filter ospf_or_131 term address-match from source-address
  10.108.0.0/16
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input ospf_or_131
```

Configuring the IPv4 Firewall Filter

Step-by-Step Procedure

To configure the IPv4 firewall filter:

1. Enable configuration of the IPv4 firewall filter.

   ```
   [edit]
   user@host# edit firewall family inet filter ospf_or_131
   ```

2. Configure the first term to accept OSPF packets.

   ```
   [edit firewall family inet filter ospf_or_131]
   user@host# set term protocol_match from protocol ospf
   ```

   Packets that match the condition are accepted by default. Because another term follows this term, packets that do not match this condition are evaluated by the next term.

3. Configure the second term to accept packets from any IPv4 address in a particular prefix.

   ```
   [edit firewall family inet filter ospf_or_131]
   user@host# set term address_match from source-address 10.108.0.0/16
   ```

   Packets that match this condition are accepted by default. Because this is the last term in the filter, packets that do not match this condition are discarded by default.

Results Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
  filter ospf_or_131 {
    term protocol_match {
      from {
        protocol ospf;
      }
    }
    term address_match {
      from {
        source-address {
          10.108.0.0/16;
        }
      }
    }
  }
}
```

Applying the IPv4 Firewall Filter to a Logical Interface

Step-by-Step Procedure To apply the stateless firewall filter to a logical interface:

1. Enable configuration of a logical interface.

   ```
   [edit]
   user@host# edit interfaces ge-0/0/1 unit 0 family inet
   ```

2. Configure an IP address for the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set address 10.1.2.3/30
   ```

3. Apply the IPv4 firewall filter to the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set filter input ospf_or_131
   ```

Results Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input ospf_or_131;
      }
      address 10.1.2.3/30;
    }
  }
```

```
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

To confirm that the configuration is working properly, enter the **show firewall filter ospf_or_131** operational mode command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Match on IPv6 Flags on page 139
- Example: Configuring a Filter to Match on Port and Protocol Fields on page 140

Example: Configuring a Filter to Accept DHCP Packets Based on Address

This example shows how to configure a standard stateless firewall filter to accept packets from a trusted source.

Requirements

This example is supported only on MX Series routers and EX Series switches.

Overview

In this example, you create 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**.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter rpf_dhcp term dhcp_term from source-address 0.0.0.0/32
set firewall family inet filter rpf_dhcp term dhcp_term from destination-address
  255.255.255.255/32
set firewall family inet filter rpf_dhcp term dhcp_term then accept
```

```
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input sam
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter:

1. Create the stateless firewall filter **rpf_dhcp**.

   ```
   [edit]
   user@host# edit firewall family inet filter rpf_dhcp
   ```

2. Configure the term to match 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]
   user@host# set term dhcp_term from source-address 0.0.0.0/32
   user@host# set term dhcp_term from destination-address 255.255.255.255/32
   ```

3. Configure the term to accept packets that match the specified conditions.

   ```
   [edit firewall family inet filter rpf_dhcp]
   set term dhcp_term then accept
   ```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the filter to the input at the loopback interface:

1. Apply the **rpf_dhcp** filter if packets are not arriving on an expected path.

   ```
   [edit]
   user@host# set interfaces lo0 unit 0 family inet rpf-check fail-filter rpf_dhcp
   ```

2. Configure an address for the loopback interface.

   ```
   [edit]
   user@host# set interfaces lo0 unit 0 family inet address 127.0.0.1/32
   ```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

   ```
   [edit]
   user@host# show firewall
   family inet {
     filter rpf_dhcp {
       term dhcp_term {
         from {
           source-address {
             0.0.0.0/32;
           }
           destination-address {
             255.255.255.255/32;
   ```

```
            }
          }
          then accept;
        }
      }
    }
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
    unit 0 {
        family inet {
            filter {
                rpf-check {
                    fail-filter rpf_dhcp;
                    mode loose;
                }
            }
            address 127.0.0.1/32;
        }
    }
}
```

3. When you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall** operational mode command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources on page 101
- Example: Configuring a Filter to Block Telnet and SSH Access on page 106
- Example: Configuring a Filter to Block TFTP Access on page 110
- Example: Configuring a Filter to Accept OSPF Packets from a Prefix on page 164

Example: Configuring a Filter to Accept OSPF Packets from a Prefix

This example shows how to configure a standard stateless firewall filter to accept packets from a trusted source.

- Requirements on page 165
- Overview on page 165

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a filter that accepts only OSPF packets from an address in the prefix 10.108.0.0/16, discarding all other packets with an **administratively-prohibited** ICMP message

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter ospf_filter term term1 from source-address 10.108.0.0/16
set firewall family inet filter ospf_filter term term1 from protocol ospf
set firewall family inet filter ospf_filter term term1 then accept
set firewall family inet filter ospf_filter term default-term then reject
  administratively-prohibited
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input ospf_filter
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **ospf_filter**:

1. Create the filter.

   ```
   [edit]
   user@host# edit firewall family inet filter ospf_filter
   ```

2. Configure the term that accepts packets.

   ```
   [edit firewall family inet filter ospf_filter]
   user@host# set term term1 from source-address 10.108.0.0/16
   user@host# set term term1 from protocol ospf
   user@host# set term term1 then accept
   ```

3. Configure the term that rejects all other packets.

```
[edit firewall family inet filter ospf_filter]
user@host# set term default_term then reject administratively-prohibited
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

To apply the firewall filter to the loopback interface:

1. Configure the interface.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the logical interface IP address.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the filter to the input.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input ospf_filter
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter ospf_filter {
        term term1 {
            from {
                source-address {
                    10.108.0.0/16;
                }
                protocol ospf;
            }
            then {
                accept;
            }
        }
        term default_term {
            then {
                reject administratively-prohibited; # default reject action
            }
        }
    }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
lo0 {
  unit 0 {
    family inet {
      filter {
        input ospf_filter;
      }
      address 10.1.2.3/30;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter ospf_filter** operational mode command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Stateless Firewall Filter to Accept Traffic from Trusted Sources on page 101
- Example: Configuring a Filter to Block Telnet and SSH Access on page 106
- Example: Configuring a Filter to Block TFTP Access on page 110
- Example: Configuring a Filter to Accept DHCP Packets Based on Address on page 162

Example: Configuring a Stateless Firewall Filter to Handle Fragments

This example shows how to create a stateless firewall filter that handles packet fragments.

Requirements

No special configuration beyond device initialization is required before configuring stateless firewall filters.

Overview

In this example, you create a stateless firewall filter called **fragment-RE** that accepts fragmented packets originating from 10.2.1.0/24 and destined for the BGP port. This example includes the following firewall filter terms:

- **not-from-prefix-term**-–Discards packets that are not from 10.2.1.0/24 to ensure that subsequent terms in the firewall filter are matched against packets from 10.2.1.0/24 only.
- **small-offset-term**—Discards small (1–5) offset packets to ensure that subsequent terms in the firewall filter can be matched against all the headers in the packet. In addition, the term adds a record to the system logging destinations for the firewall facility.
- **not-fragmented-term**—Accepts unfragmented TCP packets with a destination port that specifies the BGP protocol. A packet is considered unfragmented if the MF flag is not set and the fragment offset equals 0.
- **first-fragment-term**—Accepts the first fragment of a fragmented TCP packet with a destination port that specifies the BGP protocol.
- **fragment-term**—Accepts all fragments that were not discarded by **small-offset-term**. (packet fragments 6–8191). However, only those fragments that are part of a packet containing a first fragment accepted by **first-fragment-term** are reassembled by the destination device.

Packet fragments offset can be from 1 through 8191.

NOTE: You can move terms within the firewall filter by using the **insert** command. For more information, see "*insert*" in the *CLI User Guide*.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter fragment-RE term not-from-prefix-term from source-address
    0.0.0.0/0
set firewall family inet filter fragment-RE term not-from-prefix-term from source-address
    10.2.1.0/24 except
set firewall family inet filter fragment-RE term not-from-prefix-term then discard
set firewall family inet filter fragment-RE term small-offset-term from fragment-offset
    1-5
set firewall family inet filter fragment-RE term small-offset-term then syslog
set firewall family inet filter fragment-RE term small-offset-term then discard
set firewall family inet filter fragment-RE term not-fragmented-term from fragment-offset
    0
set firewall family inet filter fragment-RE term not-fragmented-term from fragment-flags
    "!more-fragments"
```

```
set firewall family inet filter fragment-RE term not-fragmented-term from protocol tcp
set firewall family inet filter fragment-RE term not-fragmented-term from destination-port
  bgp
set firewall family inet filter fragment-RE term not-fragmented-term then accept
set firewall family inet filter fragment-RE term first-fragment-term from first-fragment
set firewall family inet filter fragment-RE term first-fragment-term from protocol tcp
set firewall family inet filter fragment-RE term first-fragment-term from destination-port
  bgp
set firewall family inet filter fragment-RE term first-fragment-term then accept
set firewall family inet filter fragment-RE term fragment-term from fragment-offset
  6-8191
set firewall family inet filter fragment-RE term fragment-term then accept
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the stateless firewall filter:

1. Define the stateless firewall filter.

   ```
   [edit]
   user@host# edit firewall family inet filter fragment-RE
   ```

2. Configure the first term for the filter.

   ```
   [edit firewall family inet filter fragment-RE ]
   user@host# set term not-from-prefix-term from source-address 0.0.0.0/0
   user@host# set term not-from-prefix-term from source-address 10.2.1.0/24 except
   user@host# set term not-from-prefix-term then discard
   ```

3. Define the second term for the filter.

   ```
   [edit firewall family inet filter fragment-RE]
   user@host# edit term small-offset-term
   ```

4. Define the match conditions for the term.

   ```
   [edit firewall family inet filter fragment-RE term small-offset-term]
   user@host# set from fragment-offset 1-5
   ```

5. Define the action for the term.

   ```
   [edit firewall family inet filter fragment-RE term small-offset-term]
   user@host# set then syslog discard
   ```

6. Define the third term for the filter.

   ```
   [edit]
   user@host# edit firewall family inet filter fragment-RE term not-fragmented-term
   ```

7. Define the match conditions for the term.

   ```
   [edit firewall family inet filter fragment-RE term not-fragmented-term]
   user@host# set from fragment-flags "!more-fragments" fragment-offset 0 protocol
     tcp destination-port bgp
   ```

8. Define the action for the term.

   ```
   [edit firewall family inet filter fragment-RE term not-fragmented-term]
   ```

```
user@host# set then accept
```

9. Define the fourth term for the filter.

```
[edit]
user@host# edit firewall family inet filter fragment-RE term first-fragment-term
```

10. Define the match conditions for the term.

```
[edit firewall family inet filter fragment-RE term first-fragment-term]
user@host# set from first-fragment protocol tcp destination-port bgp
```

11. Define the action for the term.

```
[edit firewall family inet filter fragment-RE term first-fragment-term]
user@host# set then accept
```

12. Define the last term for the filter.

```
[edit]
user@host# edit firewall family inet filter fragment-RE term fragment-term
```

13. Define the match conditions for the term.

```
[edit firewall family inet filter fragment-RE term fragment-term]
user@host# set from fragment-offset 6–8191
```

14. Define the action for the term.

```
[edit firewall family inet filter fragment-RE term fragment-term]
user@host# set then accept
```

Results Confirm your configuration by entering the **show firewall** command from configuration mode. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show firewall
family inet {
   filter fragment-RE {
      term not-from-prefix-term {
         from {
            source-address {
               0.0.0.0/0;
               10.2.1.0/24 except;
            }
         }
         then discard;
      }
      term small-offset-term {
         from {
            fragment-offset 1-5;
         }
         then {
            syslog;
            discard;
         }
      }
      term not-fragmented-term {
```



```
        from {
          fragment-offset 0;
          fragment-flags "!more-fragments";
          protocol tcp;
          destination-port bgp;
        }
        then accept;
      }
      term first-fragment-term {
        from {
          first-fragment;
          protocol tcp;
          destination-port bgp;
        }
        then accept;
      }
      term fragment-term {
        from {
          fragment-offset 6-8191;
        }
        then accept;
      }
    }
  }
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

To confirm that the configuration is working properly, perform these tasks:

- Displaying Stateless Firewall Filter Configurations on page 171
- Verifying a Firewall Filter that Handles Fragments on page 171

Displaying Stateless Firewall Filter Configurations

Purpose Verify the configuration of the firewall filter. You can analyze the flow of the filter terms by displaying the entire configuration.

Action From configuration mode, enter the **show firewall** command.

Meaning Verify that the output shows the intended configuration of the firewall filter. In addition, verify that the terms are listed in the order in which you want the packets to be tested. You can move terms within a firewall filter by using the **insert** CLI command.

Verifying a Firewall Filter that Handles Fragments

Purpose Verify that the actions of the firewall filter terms are taken.

Action Send packets to the device that match the terms.

Meaning Verify that packets from 10.2.1.0/24 with small fragment offsets are recorded in the device's system logging destinations for the firewall facility.

Related Documentation

- *show route summary*

CHAPTER 13

Firewall Filters in Logical Systems

Example: Configuring Filter-Based Forwarding on Logical Systems

This example shows how to configure filter-based forwarding within a logical system. The filter classifies packets to determine their forwarding path within the ingress routing device.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

Filter-based forwarding is supported for IP version 4 (IPv4) and IP version 6 (IPv6).

Use filter-based forwarding for service provider selection when customers have Internet connectivity provided by different ISPs yet share a common access layer. When a shared media (such as a cable modem) is used, a mechanism on the common access layer looks at Layer 2 or Layer 3 addresses and distinguishes between customers. You can use filter-based forwarding when the common access layer is implemented using a combination of Layer 2 switches and a single router.

With filter-based forwarding, all packets received on an interface are considered. Each packet passes through a filter that has match conditions. If the match conditions are met for a filter and you have created a routing instance, filter-based forwarding is applied to a packet. The packet is forwarded based on the next hop specified in the routing instance. For static routes, the next hop can be a specific LSP.

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

To configure filter-based forwarding, perform the following tasks:

- Create a match filter on an ingress router or switch. To specify a match filter, include the **filter *filter-name*** statement at the **[edit firewall]** hierarchy level. A packet that passes through the filter is compared against a set of rules to classify it and to determine its membership in a set. Once classified, the packet is forwarded to a routing table specified in the accept action in the filter description language. The routing table then forwards the packet to the next hop that corresponds to the destination address entry in the table.
- Create routing instances that specify the routing table(s) to which a packet is forwarded, and the destination to which the packet is forwarded at the **[edit routing-instances]** or **[edit logical-systems *logical-system-name* routing-instances]** hierarchy level. For example:

```
[edit]
routing-instances {
   routing-table-name1 {
      instance-type forwarding;
      routing-options {
         static {
            route 0.0.0.0/0 nexthop 10.0.0.1;
         }
      }
   }
   routing-table-name2 {
      instance-type forwarding;
      routing-options {
         static {
            route 0.0.0.0/0 nexthop 10.0.0.2;
         }
      }
   }
}
```

- Create a routing table group that adds interface routes to the forwarding routing instances used in filter-based forwarding (FBF), as well as to the default routing instance **inet.0**. This part of the configuration resolves the routes installed in the routing instances to directly connected next hops on that interface. Create the routing table group at the **[edit routing-options]** or **[edit logical-systems *logical-system-name* routing-options]** hierarchy level.

NOTE: Specify inet.0 as one of the routing instances that the interface routes are imported into. If the default instance inet.0 is not specified, interface routes are not imported into the default routing instance.

This example shows a packet filter that directs customer traffic to a next-hop router in the domains, SP 1 or SP 2, based on the packet's source address.

If the packet has a source address assigned to an SP 1 customer, destination-based forwarding occurs using the sp1-route-table.inet.0 routing table. If the packet has a source address assigned to an SP 2 customer, destination-based forwarding occurs using the sp2-route-table.inet.0 routing table. If a packet does not match either of these conditions, the filter accepts the packet, and destination-based forwarding occurs using the standard inet.0 routing table.

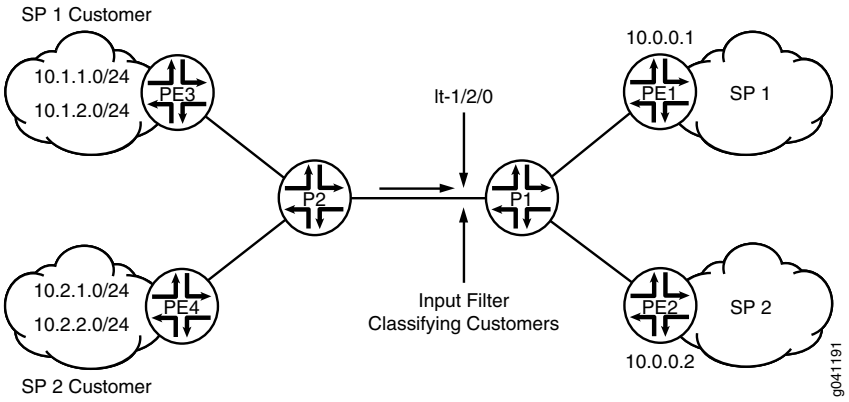
One way to make filter-based forwarding work within a logical system is to configure the firewall filter on the logical system that receives the packets. Another way is to configure the firewall filter on the main router or switch and then reference the logical system in the firewall filter. This example uses the second approach. The specific routing instances are configured within the logical system. Because each routing instance has its own routing table, you have to reference the routing instances in the firewall filter, as well. The syntax looks as follows:

[edit firewall filter *filter-name* term *term-name*]
user@host# **set then logical-system *logical-system-name* routing-instance *routing-instance-name***

Figure 7 on page 175 shows the topology used in this example.

On Logical System P1, an input filter classifies packets received from Logical System PE3 and Logical System PE4. The packets are routed based on the source addresses. Packets with source addresses in the 10.1.1.0/24 and 10.1.2.0/24 networks are routed to Logical System PE1. Packets with source addresses in the 10.2.1.0/24 and 10.2.2.0/24 networks are routed to Logical System PE2.

Figure 7: Logical Systems with Filter-Based Forwarding



To establish connectivity, OSPF is configured on all of the interfaces. For demonstration purposes, loopback interface addresses are configured on the routing devices to represent networks in the clouds.

The "CLI Quick Configuration" on page 176 section shows the entire configuration for all of the devices in the topology. The "Configuring the Routing Instances on the Logical System P1" on page 178 and "Configuring the Firewall Filter on the Main Router" on page 177

sections shows the step-by-step configuration of the ingress routing device, Logical System P1.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall filter classify-customers term sp1-customers from source-address 10.1.1.0/24
set firewall filter classify-customers term sp1-customers from source-address 10.1.2.0/24
set firewall filter classify-customers term sp1-customers then log
set firewall filter classify-customers term sp1-customers then logical-system P1
  routing-instance sp1-route-table
set firewall filter classify-customers term sp2-customers from source-address 10.2.1.0/24
set firewall filter classify-customers term sp2-customers from source-address 10.2.2.0/24
set firewall filter classify-customers term sp2-customers then log
set firewall filter classify-customers term sp2-customers then logical-system P1
  routing-instance sp2-route-table
set firewall filter classify-customers term default then accept
set logical-systems P1 interfaces lt-1/2/0 unit 10 encapsulation ethernet
set logical-systems P1 interfaces lt-1/2/0 unit 10 peer-unit 9
set logical-systems P1 interfaces lt-1/2/0 unit 10 family inet filter input classify-customers
set logical-systems P1 interfaces lt-1/2/0 unit 10 family inet address 172.16.0.10/30
set logical-systems P1 interfaces lt-1/2/0 unit 13 encapsulation ethernet
set logical-systems P1 interfaces lt-1/2/0 unit 13 peer-unit 14
set logical-systems P1 interfaces lt-1/2/0 unit 13 family inet address 172.16.0.13/30
set logical-systems P1 interfaces lt-1/2/0 unit 17 encapsulation ethernet
set logical-systems P1 interfaces lt-1/2/0 unit 17 peer-unit 18
set logical-systems P1 interfaces lt-1/2/0 unit 17 family inet address 172.16.0.17/30
set logical-systems P1 protocols ospf rib-group fbf-group
set logical-systems P1 protocols ospf area 0.0.0.0 interface all
set logical-systems P1 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems P1 routing-instances sp1-route-table instance-type forwarding
set logical-systems P1 routing-instances sp1-route-table routing-options static route
  0.0.0.0/0 next-hop 172.16.0.13
set logical-systems P1 routing-instances sp2-route-table instance-type forwarding
set logical-systems P1 routing-instances sp2-route-table routing-options static route
  0.0.0.0/0 next-hop 172.16.0.17
set logical-systems P1 routing-options rib-groups fbf-group import-rib inet.0
set logical-systems P1 routing-options rib-groups fbf-group import-rib
  sp1-route-table.inet.0
set logical-systems P1 routing-options rib-groups fbf-group import-rib
  sp2-route-table.inet.0
set logical-systems P2 interfaces lt-1/2/0 unit 2 encapsulation ethernet
set logical-systems P2 interfaces lt-1/2/0 unit 2 peer-unit 1
set logical-systems P2 interfaces lt-1/2/0 unit 2 family inet address 172.16.0.2/30
set logical-systems P2 interfaces lt-1/2/0 unit 6 encapsulation ethernet
set logical-systems P2 interfaces lt-1/2/0 unit 6 peer-unit 5
set logical-systems P2 interfaces lt-1/2/0 unit 6 family inet address 172.16.0.6/30
set logical-systems P2 interfaces lt-1/2/0 unit 9 encapsulation ethernet
set logical-systems P2 interfaces lt-1/2/0 unit 9 peer-unit 10
set logical-systems P2 interfaces lt-1/2/0 unit 9 family inet address 172.16.0.9/30
set logical-systems P2 protocols ospf area 0.0.0.0 interface all
```

```
set logical-systems P2 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE1 interfaces lt-1/2/0 unit 14 encapsulation ethernet
set logical-systems PE1 interfaces lt-1/2/0 unit 14 peer-unit 13
set logical-systems PE1 interfaces lt-1/2/0 unit 14 family inet address 172.16.0.14/30
set logical-systems PE1 interfaces lo0 unit 3 family inet address 1.1.1.1/32
set logical-systems PE1 protocols ospf area 0.0.0.0 interface all
set logical-systems PE1 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE2 interfaces lt-1/2/0 unit 18 encapsulation ethernet
set logical-systems PE2 interfaces lt-1/2/0 unit 18 peer-unit 17
set logical-systems PE2 interfaces lt-1/2/0 unit 18 family inet address 172.16.0.18/30
set logical-systems PE2 interfaces lo0 unit 4 family inet address 2.2.2.2/32
set logical-systems PE2 protocols ospf area 0.0.0.0 interface all
set logical-systems PE2 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE3 interfaces lt-1/2/0 unit 1 encapsulation ethernet
set logical-systems PE3 interfaces lt-1/2/0 unit 1 peer-unit 2
set logical-systems PE3 interfaces lt-1/2/0 unit 1 family inet address 172.16.0.1/30
set logical-systems PE3 interfaces lo0 unit 1 family inet address 10.1.1.1/32
set logical-systems PE3 interfaces lo0 unit 1 family inet address 10.1.2.1/32
set logical-systems PE3 protocols ospf area 0.0.0.0 interface all
set logical-systems PE3 protocols ospf area 0.0.0.0 interface fxp0.0 disable
set logical-systems PE4 interfaces lt-1/2/0 unit 5 encapsulation ethernet
set logical-systems PE4 interfaces lt-1/2/0 unit 5 peer-unit 6
set logical-systems PE4 interfaces lt-1/2/0 unit 5 family inet address 172.16.0.5/30
set logical-systems PE4 interfaces lo0 unit 2 family inet address 10.2.1.1/32
set logical-systems PE4 interfaces lo0 unit 2 family inet address 10.2.2.1/32
set logical-systems PE4 protocols ospf area 0.0.0.0 interface all
set logical-systems PE4 protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Configuring the Firewall Filter on the Main Router

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the firewall filter on the main router or switch:

1. Configure the source addresses for SP1 customers.

   ```
   [edit firewall filter classify-customers term sp1-customers]
   user@host# set from source-address 10.1.1.0/24
   user@host# set from source-address 10.1.2.0/24
   ```

2. Configure the actions that are taken when packets are received with the specified source addresses.

   To track the action of the firewall filter, a log action is configured. The sp1-route-table.inet.0 routing table on Logical System P1 routes the packets.

   ```
   [edit firewall filter classify-customers term sp1-customers]
   user@host# set then log
   user@host# set then logical-system P1 routing-instance sp1-route-table
   ```

3. Configure the source addresses for SP2 customers.

   ```
   [edit firewall filter classify-customers term sp2-customers]
   user@host# set from source-address 10.2.1.0/24
   user@host# set from source-address 10.2.2.0/24
   ```

4. Configure the actions that are taken when packets are received with the specified source addresses.

   To track the action of the firewall filter, a log action is configured. The sp2-route-table.inet.0 routing table on Logical System P1 routes the packet.

   ```
   [edit firewall filter classify-customers term sp2-customers]
   user@host# set then log
   user@host# set then logical-system P1 routing-instance sp2-route-table
   ```

5. Configure the action to take when packets are received from any other source address.

   All of these packets are simply accepted and routed using the default IPv4 unicast routing table, inet.0.

   ```
   [edit firewall filter classify-customers term default]
   user@host# set then accept
   ```

Configuring the Routing Instances on the Logical System P1

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the routing instances on a logical system:

1. Configure the interfaces on the logical system.

   ```
   [edit logical-systems P1 interfaces lt-1/2/0]
   user@host# set unit 10 encapsulation ethernet
   user@host# set unit 10 peer-unit 9
   user@host# set unit 10 family inet address 172.16.0.10/30

   user@host# set unit 13 encapsulation ethernet
   user@host# set unit 13 peer-unit 14
   user@host# set unit 13 family inet address 172.16.0.13/30

   user@host# set unit 17 encapsulation ethernet
   user@host# set unit 17 peer-unit 18
   user@host# set unit 17 family inet address 172.16.0.17/30
   ```

2. Assign the **classify-customers** firewall filter to router interface lt-1/2/0.10 as an input packet filter.

   ```
   [edit logical-systems P1 interfaces lt-1/2/0]
   user@host# set unit 10 family inet filter input classify-customers
   ```

3. Configure connectivity, using either a routing protocol or static routing.

   As a best practice, disable routing on the management interface.

   ```
   [edit logical-systems P1 protocols ospf area 0.0.0.0]
   user@host# set interface all
   user@host# set interface fxp0.0 disable
   ```

4. Create the routing instances.

   These routing instances are referenced in the **classify-customers** firewall filter.

   The forwarding instance type provides support for filter-based forwarding, where interfaces are not associated with instances. All interfaces belong to the default instance, in this case Logical System P1.

   ```
   [edit logical-systems P1 routing-instances]
   user@host# set sp1-route-table instance-type forwarding


   user@host# set sp2-route-table instance-type forwarding
   ```

5. Resolve the routes installed in the routing instances to directly connected next hops.

   ```
   [edit logical-systems P1 routing-instances]
   user@host# set sp1-route-table routing-options static route 0.0.0.0/0 next-hop
     172.16.0.13


   user@host# set sp2-route-table routing-options static route 0.0.0.0/0 next-hop
     172.16.0.17
   ```

6. Group together the routing tables to form a routing table group.

   The first routing table, inet.0, is the primary routing table, and the additional routing tables are the secondary routing tables.

   The primary routing table determines the address family of the routing table group, in this case IPv4.

   ```
   [edit logical-systems P1 routing-options]
   user@host# set rib-groups fbf-group import-rib inet.0
   user@host# set rib-groups fbf-group import-rib sp1-route-table.inet.0
   user@host# set rib-groups fbf-group import-rib sp2-route-table.inet.0
   ```

7. Apply the routing table group to OSPF.

   This causes the OSPF routes to be installed into all the routing tables in the group.

   ```
   [edit logical-systems P1 protocols ospf]
   user@host# set rib-group fbf-group
   ```

8. If you are done configuring the device, commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

Results

Confirm your configuration by issuing the **show firewall** and **show logical-systems P1** commands.

```
user@host# show firewall
filter classify-customers {
   term sp1-customers {
      from {
         source-address {
            10.1.1.0/24;
```

```
            10.1.2.0/24;
          }
        }
        then {
          log;
          logical-system P1 routing-instance sp1-route-table;
        }
      }
      term sp2-customers {
        from {
          source-address {
            10.2.1.0/24;
            10.2.2.0/24;
          }
        }
        then {
          log;
          logical-system P1 routing-instance sp2-route-table;
        }
      }
      term default {
        then accept;
      }
}
user@host# show logical-systems P1
interfaces {
    lt-1/2/0 {
        unit 10 {
            encapsulation ethernet;
            peer-unit 9;
            family inet {
                filter {
                    input classify-customers;
                }
                address 172.16.0.10/30;
            }
        }
        unit 13 {
            encapsulation ethernet;
            peer-unit 14;
            family inet {
                address 172.16.0.13/30;
            }
        }
        unit 17 {
            encapsulation ethernet;
            peer-unit 18;
            family inet {
                address 172.16.0.17/30;
            }
        }
    }
}
protocols {
    ospf {
```

```
        rib-group fbf-group;
        area 0.0.0.0 {
          interface all;
          interface fxp0.0 {
            disable;
          }
        }
      }
    }
    routing-instances {
      sp1-route-table {
        instance-type forwarding;
        routing-options {
          static {
            route 0.0.0.0/0 next-hop 172.16.0.13;
          }
        }
      }
      sp2-route-table {
        instance-type forwarding;
        routing-options {
          static {
            route 0.0.0.0/0 next-hop 172.16.0.17;
          }
        }
      }
    }
    routing-options {
      rib-groups {
        fbf-group {
          import-rib [ inet.0 sp1-route-table.inet.0 sp2-route-table.inet.0 ];
        }
      }
    }
```

Verification

Confirm that the configuration is working properly.

Pinging with Specified Source Addresses

Purpose Send some ICMP packets across the network to test the firewall filter.

Action

1. Log in to Logical System PE3.

   ```
   user@host> set cli logical-system PE3
   Logical system: PE3
   ```

2. Run the **ping** command, pinging the lo0.3 interface on Logical System PE1.

   The address configured on this interface is 1.1.1.1.

   Specify the source address 10.1.2.1, which is the address configured on the lo0.1 interface on Logical System PE3.

   ```
   user@host:PE3> ping 1.1.1.1 source 10.1.2.1
   ```

```
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=62 time=1.444 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=62 time=2.094 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.444/1.769/2.094/0.325 ms
```

3. Log in to Logical System PE4.

```
user@host:PE3> set cli logical-system PE4
Logical system: PE4
```

4. Run the **ping** command, pinging the lo0.4 interface on Logical System PE2.

   The address configured on this interface is 2.2.2.2.

   Specify the source address 10.2.1.1, which is the address configured on the lo0.2 interface on Logical System PE4.

```
user@host:PE4> ping 2.2.2.2 source 10.2.1.1
PING 2.2.2.2 (2.2.2.2): 56 data bytes
64 bytes from 2.2.2.2: icmp_seq=0 ttl=62 time=1.473 ms
64 bytes from 2.2.2.2: icmp_seq=1 ttl=62 time=1.407 ms
^C
--- 2.2.2.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.407/1.440/1.473/0.033 ms
```

Meaning Sending these pings activates the firewall filter actions.

Verifying the Firewall Filter

Purpose Make sure the firewall filter actions take effect.

Action

1. Log in to Logical System P1.

```
user@host> set cli logical-system P1
Logical system: P1
```

2. Run the **show firewall log** command on Logical System P1.

```
user@host:P1> show firewall log
Log :
Time      Filter    Action Interface     Protocol        Src Addr
Dest Addr
13:52:20  pfe       A      lt-1/2/0.10   ICMP            10.2.1.1
2.2.2.2
13:52:19  pfe       A      lt-1/2/0.10   ICMP            10.2.1.1
2.2.2.2
13:51:53  pfe       A      lt-1/2/0.10   ICMP            10.1.2.1
1.1.1.1
13:51:52  pfe       A      lt-1/2/0.10   ICMP            10.1.2.1
1.1.1.1
```

Related Documentation

- Example: Configuring Filter-Based Forwarding on the Source Address on page 233
- *Example: Configuring Multitopology Routing Based on Applications*
- *Copying and Redirecting Traffic with Port Mirroring and Filter-Based Forwarding*

- *Using Filter-Based Forwarding to Export Monitored Traffic to Multiple Destinations*
- Filter-Based Forwarding Overview on page 75

Example: Configuring a Stateless Firewall Filter to Protect a Logical System Against ICMP Floods

This example shows how to configure a stateless firewall filter that protects against ICMP denial-of-service attacks on a logical system.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

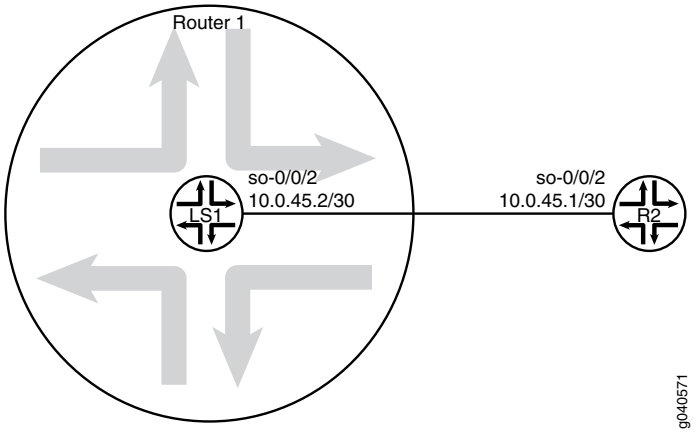
This example shows a stateless firewall filter called protect-RE that polices ICMP packets. The **icmp-policer** limits the traffic rate of the ICMP packets to 1,000,000 bps and the burst size to 15,000 bytes. Packets that exceed the traffic rate are discarded.

The policer is incorporated into the action of a filter term called **icmp-term**.

In this example, a ping is sent from a directly connected physical router to the interface configured on the logical system. The logical system accepts the ICMP packets if they are received at a rate of up to 1 Mbps (bandwidth-limit). The logical system drops all ICMP packets when this rate is exceeded. The **burst-size-limit** statement accepts traffic bursts up to 15 Kbps. If bursts exceed this limit, all packets are dropped. When the flow rate subsides, ICMP packets are again accepted.

Figure 8 on page 184 shows the topology used in this example.

Figure 8: Logical System with a Stateless Firewall



Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

```
set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet policer input icmp-policer
set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet address 10.0.45.2/30
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term from protocol
    icmp
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then policer
    icmp-policer
set logical-systems LS1 firewall family inet filter protect-RE term icmp-term then accept
set logical-systems LS1 firewall policer icmp-policer if-exceeding bandwidth-limit 1m
set logical-systems LS1 firewall policer icmp-policer if-exceeding burst-size-limit 15k
set logical-systems LS1 firewall policer icmp-policer then discard
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure an ICMP firewall filter on a logical system:

1. Configure the interface on the logical system.

   ```
   [edit]
   user@host# set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet address
       10.0.45.2/30
   ```

2. Explicitly enable ICMP packets to be received on the interface.

   ```
   [edit]
   user@host# set logical-systems LS1 firewall family inet filter protect-RE term
       icmp-term from protocol icmp
   user@host# set logical-systems LS1 firewall family inet filter protect-RE term
       icmp-term then accept
   ```

3. Create the policer.

```
[edit]
user@host# set logical-systems LS1 firewall policer icmp-policer if-exceeding
  bandwidth-limit 1m
user@host# set logical-systems LS1 firewall policer icmp-policer if-exceeding
  burst-size-limit 15k
user@host# set logical-systems LS1 firewall policer icmp-policer then discard
```

4. Apply the policer to a filter term.

```
[edit]
user@host# set logical-systems LS1 firewall family inet filter protect-RE term
  icmp-term then policer icmp-policer
```

5. Apply the policer to the logical system interface.

```
[edit]
user@host# set logical-systems LS1 interfaces so-0/0/2 unit 0 family inet policer
  input icmp-policer
```

6. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Results

Confirm your configuration by issuing the **show logical-systems LS1** command.

```
user@host# show logical-systems LS1
interfaces {
   so-0/0/2 {
      unit 0 {
         family inet {
            policer {
               input icmp-policer;
            }
            address 10.0.45.2/30;
         }
      }
   }
}
firewall {
   family inet {
      filter protect-RE {
         term icmp-term {
            from {
               protocol icmp;
            }
            then {
               policer icmp-policer;
               accept;
            }
         }
      }
   }
```

```
    policer icmp-policer {
      if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
      }
      then discard;
    }
  }
```

Verification

Confirm that the configuration is working properly.

Verifying That Ping Works Unless the Limits Are Exceeded

Purpose Make sure that the logical system interface is protected against ICMP-based DoS attacks.

Action Log in to a system that has connectivity to the logical system and run the **ping** command.

```
user@R2> ping 10.0.45.2
PING 10.0.45.2 (10.0.45.2): 56 data bytes
64 bytes from 10.0.45.2: icmp_seq=0 ttl=64 time=1.316 ms
64 bytes from 10.0.45.2: icmp_seq=1 ttl=64 time=1.277 ms
64 bytes from 10.0.45.2: icmp_seq=2 ttl=64 time=1.269 ms

user@R2> ping 10.0.45.2 size 20000
PING 10.0.45.2 (10.0.45.2): 20000 data bytes
^C
--- 10.0.45.2 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

Meaning When you send a normal ping, the packet is accepted. When you send a ping packet that exceeds the filter limit, the packet is discarded.

Related Documentation

- *Example: Creating an Interface on a Logical System*

CHAPTER 14

Firewall Filter Accounting and Logging

Example: Configuring Statistics Collection for a Firewall Filter

This example shows how to configure and apply a firewall filter that collects data according to parameters specified in an associated accounting profile.

Requirements

Firewall filter accounting profiles are supported for all traffic types except **family any**.

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create a firewall filter accounting profile and apply it to a firewall filter. The accounting profile specifies how frequently to collect packet and byte count statistics and the name of the file to which the statistics are written. The profile also specifies that statistics are to be collected for three firewall filter counters.

Topology

The firewall filter accounting profile **filter_acctg_profile** specifies that statistics are collected every 60 minutes, and the statistics are written to the file **/var/log/ff_accounting_file**. Statistics are collected for counters named **counter1**, **counter2**, and **counter3**.

The IPv4 firewall filter named **my_firewall_filter** increments a counter for each of three filter terms. The filter is applied to logical interface **ge-0/0/1.0**.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set accounting-options filter-profile filter_acctg_profile file ff_accounting_file
set accounting-options filter-profile filter_acctg_profile interval 60
set accounting-options filter-profile filter_acctg_profile counters counter1
set accounting-options filter-profile filter_acctg_profile counters counter2
set accounting-options filter-profile filter_acctg_profile counters counter3
set firewall family inet filter my_firewall_filter accounting-profile filter_acctg_profile
set firewall family inet filter my_firewall_filter term term1 from protocol ospf
set firewall family inet filter my_firewall_filter term term1 then count counter1
set firewall family inet filter my_firewall_filter term term1 then discard
set firewall family inet filter my_firewall_filter term term2 from source-address
  10.108.0.0/16
set firewall family inet filter my_firewall_filter term term2 then count counter2
set firewall family inet filter my_firewall_filter term term2 then discard
set firewall family inet filter my_firewall_filter term accept-all then count counter3
set firewall family inet filter my_firewall_filter term accept-all then accept
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input my_firewall_filter
```

Configure an Accounting Profile

Step-by-Step Procedure

To configure an accounting profile:

1. Create the accounting profile **filter_acctg_profile**.

   ```
   [edit]
   user@host# edit accounting-options filter-profile filter_acctg_profile
   ```

2. Configure the accounting profile to filter and collect packet and byte count statistics every 60 minutes and write them to the **/var/log/ff_accounting_file** file.

   ```
   [edit accounting-options filter-profile filter_acctg_profile]
   user@host# set file ff_accounting_file
   user@host# set interval 60
   ```

3. Configure the accounting profile to collect filter profile statistics (packet and byte counts) for three counters.

```
[edit accounting-options filter-profile filter_acctg_profile]
user@host# set counters counter1
user@host# set counters counter2
user@host# set counters counter3
```

Configure a Firewall Filter That References the Accounting Profile

Step-by-Step Procedure

To configure a firewall filter that references the accounting profile:

1. Create the firewall filter **my_firewall_filter**.

```
[edit]
user@host# edit firewall family inet filter my_firewall_filter
```

2. Apply the filter-accounting profile **filter_acctg_profile** to the firewall filter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set accounting-profile filter_acctg_profile
```

3. Configure the first filter term and counter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set term term1 from protocol ospf
user@host# set term term1 then count counter1
user@host# set term term1 then discard
```

4. Configure the second filter term and counter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set term term2 from source-address 10.108.0.0/16
user@host# set term term2 then count counter2
user@host# set term term2 then discard
```

5. Configure the third filter term and counter.

```
[edit firewall family inet filter my_firewall_filter]
user@host# set term accept-all then count counter3
user@host# set term accept-all then accept
```

Apply the Firewall Filter to an Interface

Step-by-Step Procedure

To apply the firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the firewall filter.

```
[edit]
user@host# edit interfaces ge-0/0/1 unit 0 family inet
```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input my_firewall_filter
```

Confirm Your Candidate Configuration

Step-by-Step Procedure

To confirm your candidate configuration:

1. Confirm the configuration of the accounting profile by entering the **show accounting-options** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show accounting-options
filter-profile filter_acctg_profile {
    file ff_accounting_file;
    interval 60;
    counters {
        counter1;
        counter2;
        counter3;
    }
}
```

2. Confirm the configuration of the firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter my_firewall_filter {
        accounting-profile filter_acctg_profile;
        term term1 {
            from {
                protocol ospf;
            }
            then {
                count counter1;
                discard;
            }
        }
        term term2 {
            from {
                source-address {
                    10.108.0.0/16;
                }
            }
            then {
                count counter2;
                discard;
            }
        }
        term accept-all {
```

```
            then {
              count counter3;
              accept;
            }
          }
        }
      }
```

3. Confirm the configuration of the interfaces by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input my_firewall_filter;
      }
      address 10.1.2.3/30;
    }
  }
}
```

Clear the Counters and Commit Your Candidate Configuration

Step-by-Step Procedure

To clear the counters and commit your candidate configuration:

1. From operational command mode, use the **clear firewall all** command to clear the statistics for all firewall filters.

   To clear only the counters incremented in this example, include the name of the firewall filter.

```
[edit]
user@host> clear firewall filter my_firewall_filter
```

2. Commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To verify that the filter is applied to the logical interface, run the **show interfaces** command with the **detail** or **extensive** output level.

To verify that the three counters are collected separately, run the **show firewall filter my_firewall_filter** command.

```
user@host> show firewall filter my_firewall_filter

Filter: my_firewall_filter
Counters:
Name                                                Bytes              Packets
counter1                                                0                    0
```

```
counter2                                          0                    0
counter3                                          0                    0
```

Related Documentation

- Accounting for Firewall Filters Overview on page 45
- Statement Hierarchy for Configuring Firewall Filter Accounting Profiles on page 279
- Statement Hierarchy for Applying Firewall Filter Accounting Profiles on page 280

Example: Configuring Logging for a Firewall Filter Term

This example shows how to configure a firewall filter to log packet headers.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a firewall filter that logs and counts ICMP packets that have **192.168.207.222** as either their source or destination.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set system syslog file messages_firewall_any firewall any
set system syslog file messages_firewall_any archive no-world-readable
set firewall family inet filter icmp_syslog term icmp_match from address
    192.168.207.222/32
set firewall family inet filter icmp_syslog term icmp_match from protocol icmp
set firewall family inet filter icmp_syslog term icmp_match then count packets
```

```
set firewall family inet filter icmp_syslog term icmp_match then syslog
set firewall family inet filter icmp_syslog term icmp_match then accept
set firewall family inet filter icmp_syslog term default_term then accept
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input icmp_syslog
```

Configure the Syslog Messages File for the Firewall Facility

Step-by-Step Procedure

To configure a syslog messages file for the **firewall** facility:

1. Configure a messages file for all syslog messages generated for the **firewall** facility.

   ```
   user@host# set system syslog file messages_firewall_any firewall any
   ```

2. Restrict permission to the archived **firewall** facility syslog files to the root user and users who have the Junos OS maintenance permission.

   ```
   user@host# set system syslog file messages_firewall_any archive no-world-readable
   ```

Configure the Firewall Filter

Step-by-Step Procedure

To configure the firewall filter **icmp_syslog** that logs and counts ICMP packets that have **192.168.207.222** as either their source or destination:

1. Create the firewall filter **icmp_syslog**.

   ```
   [edit]
   user@host# edit firewall family inet filter icmp_syslog
   ```

2. Configure matching on the ICMP protocol and an address.

   ```
   [edit firewall family inet filter icmp_syslog]
   user@host# set term icmp_match from address 192.168.207.222/32
   user@host# set term icmp_match from protocol icmp
   ```

3. Count, log,, and accept matching packets.

   ```
   [edit firewall family inet filter icmp_syslog]
   user@host# set term icmp_match then count packets
   user@host# set term icmp_match then syslog
   user@host# set term icmp_match then accept
   ```

4. Accept all other packets.

   ```
   [edit firewall family inet filter icmp_syslog]
   user@host# set term default_term then accept
   ```

Apply the Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the firewall filter to a logical interface:

1. Configure the logical interface to which you will apply the firewall filter.

   ```
   [edit]
   user@host# edit interfaces ge-0/0/1 unit 0 family inet
   ```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input icmp_syslog
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the syslog message file for the **firewall** facility by entering the **show system** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show system
syslog {
  file messages_firewall_any {
    firewall any;
    archive no-world-readable;
  }
}
```

2. Confirm the configuration of the firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show 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_term {
      then accept;
    }
  }
}
```

3. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input icmp_syslog;
            }
            address 10.1.2.3/30;
        }
    }
}
```

4. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show log filter** command:

```
user@host> show log messages_firewall_any
Mar 20 08:03:11 hostname 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 1-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:

```
user@host> show log messages_firewall_any
Mar 20 08:08:45 hostname feb FW: so-0/1/0.0   A icmp 192.168.207.222

192.168.207.223     0     0 (515 packets)
```

Related Documentation

- System Logging Overview on page 45
- Logging of Packet Headers Evaluated by a Firewall Filter Term on page 48
- System log messages with the **DFWD_** prefix, described in the *Junos OS System Log Messages Reference*
- System log messages with the **PFE_FW_*** prefix, described in the *Junos OS System Log Messages Reference*

CHAPTER 15

Multiple Firewall Filters Associated With a Single Interface

Example: Applying Lists of Multiple Firewall Filters

This example shows how to apply lists of multiple firewall filters.

Requirements

Before you begin, be sure that you have:

- Installed your router or switch, and supported PIC, DPC, or MPC and performed the initial router or switch configuration.
- Configured basic Ethernet in the topology.
- Configured a logical interface to run the IP version 4 (IPv4) protocol (**family inet**) and configured the logical interface with an interface address. This example uses logical interface **ge-1/3/0.0** configured with the IP address 1.1.1.2/30.

  NOTE: **For completeness, the configuration section of this example includes setting an IP address for logical interface ge-1/3/0.0.**

- Verified that traffic is flowing in the topology and that ingress and egress IPv4 traffic is flowing through logical interface **ge-1/3/0.0**.
- Verified that you have access to the remote host that is connected to this router's or switch's logical interface **ge-1/3/0.0**.

Overview

In this example, you configure three IPv4 firewall filters and apply each filter directly to the same logical interface by using a list.

Topology

This example applies the following firewall filters as a *list of input filters* at logical interface **ge-1/3/0.0**. Each filter contains a single term that evaluates IPv4 packets and accepts packets based on the value of the **destination port** field in the TCP header:

- Filter **filter_FTP** matches on the FTP port number (**21**).
- Filter **filter_SSH** matches on the SSH port number (**22**).
- Filter **filter_Telnet** matches on the Telnet port number (**23**).

If an inbound packet does not match any of the filters in the input list, the packet is discarded.

NOTE: The Junos OS uses filters in a list in the order in which the filter names appear in the list. In this simple example, the order is irrelevant because all of the filters specify the same action.

Any of the filters can be applied to other interfaces, either alone (using the **input** or **output** statement) or in combination with other filters (using the **input-list** or **output-list** statement). The objective is to configure multiple "minimalist" firewall filters that you can reuse in interface-specific filter lists.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

- Configure Multiple IPv4 Firewall Filters on page 199
- Apply the Filters to a Logical Interface as an Input List and an Output List on page 199
- Confirm and Commit Your Candidate Configuration on page 200

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter filter_FTP term 0 from protocol tcp
set firewall family inet filter filter_FTP term 0 from destination-port 21
set firewall family inet filter filter_FTP term 0 then count pkts_FTP
set firewall family inet filter filter_FTP term 0 then accept
set firewall family inet filter filter_SSH term 0 from protocol tcp
set firewall family inet filter filter_SSH term 0 from destination-port 23
set firewall family inet filter filter_SSH term 0 then count pkts_SSH
set firewall family inet filter filter_SSH term 0 then accept
set firewall family inet filter filter_Telnet term 0 from protocol tcp
set firewall family inet filter filter_Telnet term 0 from destination-port 22
```

```
set firewall family inet filter filter_Telnet term 0 then count pkts_Telnet
set firewall family inet filter filter_Telnet term 0 then accept
set firewall family inet filter filter_discard term 1 then count pkts_discarded
set firewall family inet filter filter_discard term 1 then discard
set interfaces ge-1/3/0 unit 0 family inet address 1.1.1.2/30
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_FTP
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_SSH
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_Telnet
set interfaces ge-1/3/0 unit 0 family inet filter input-list filter_discard
```

Configure Multiple IPv4 Firewall Filters

Step-by-Step Procedure

To configure the IPv4 firewall filters:

1. Navigate the CLI to the hierarchy level at which you configure IPv4 firewall filters.

   ```
   [edit]
   user@host# edit firewall family inet
   ```

2. Configure the first firewall filter to count and accept packets for port 21.

   ```
   [edit firewall family inet]
   user@host# set filter filter_FTP term 0 from protocol tcp
   user@host# set filter filter_FTP term 0 from destination-port 21
   user@host# set filter filter_FTP term 0 then count pkts_FTP
   user@host# set filter filter_FTP term 0 then accept
   ```

3. Configure the second firewall filter to count and accept packets for port 23.

   ```
   [edit firewall family inet]
   user@host# set filter filter_SSH term 0 from protocol tcp
   user@host# set filter filter_SSH term 0 from destination-port 23
   user@host# set filter filter_SSH term 0 then count pkt_SSH
   user@host# set filter filter_SSH term 0 then accept
   ```

4. Configure the third firewall filter to count and accept packets from port 22.

   ```
   [edit firewall family inet]
   user@host# set filter filter_Telnet term 0 from protocol tcp
   user@host# set filter filter_Telnet term 0 from destination-port 22
   user@host# set filter filter_Telnet term 0 then count pkts_Telnet
   user@host# set filter filter_Telnet term 0 then accept
   ```

5. Configure the last firewall filter to count the discarded packets.

   ```
   [edit firewall family inet]
   user@host# set filter filter_discard term 1 then count pkts_discarded
   user@host# set filter filter_discard term 1 then discard
   ```

Apply the Filters to a Logical Interface as an Input List and an Output List

Step-by-Step Procedure

To apply the six IPv4 firewall filters as a list of input filters and a list of output filters:

1. Navigate the CLI to the hierarchy level at which you apply IPv4 firewall filters to logical interface **ge-1/3/0.0**.

   ```
   [edit]
   user@host# edit interfaces ge-1/3/0 unit 0 family inet
   ```

2. Configure the IPv4 protocol family for the logical interface.

```
[edit interfaces ge-1/3/0 unit 0 family inet]
user@host# set address 1.1.1.2/30
```

3. Apply the filters as a list of input filters.

```
[edit interfaces ge-1/3/0 unit 0 family inet]
user@host# set filter input-list [ filter_FTP filter_SSH filter_Telnet filter_discard ]
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the firewall filters by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
  filter filter_FTP {
    term 0 {
      from {
        protocol tcp;
        destination-port 21;
      }
      then {
        count pkts_FTP;
        accept;
      }
    }
  }
  filter filter_SSH {
    term 0 {
      from {
        protocol tcp;
        destination-port 23;
      }
      then {
        count pkts_SSH;
        accept;
      }
    }
  }
  filter filter_Telnet {
    term 0 {
      from {
        protocol tcp;
        destination-port 22;
      }
      then {
        count pkts_Telnet;
        accept;
      }
    }
```

```
    }
    filter filter_discard {
      term 1 {
        then {
          count pkts_discarded;
          discard;
        }
      }
    }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-1/3/0 {
  unit 0 {
    family inet {
      filter {
        input-list [ filter_FTP filter_SSH filter_Telnet filter_discard ];
      }
      address 1.1.1.2/30;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

Confirm that the configuration is working properly.

- Verifying That Inbound Packets Are Accepted Only If Destined for the FTP, SSH or Telnet Port on page 201

Verifying That Inbound Packets Are Accepted Only If Destined for the FTP, SSH or Telnet Port

Purpose Verify that all three filters are active for the logical interface.

Action To verify that input packets are accepted according to the three filters:

1. From the remote host that is connected to this router's (or switch's) logical interface **ge-1/3/0.0**, send a packet with destination port number 21 in the header. The packet should be accepted.
2. From the remote host that is connected to this router's (or switch's) logical interface **ge-1/3/0.0**, send a packet with destination port number 23 in the header. The packet should be accepted.

3. From the remote host that is connected to this router's (or switch's) logical interface **ge-1/3/0.0**, send a packet with destination port number 22 in the header. The packet should be accepted.
4. From the remote host that is connected to this router's (or switch's) logical interface **ge-1/3/0.0**, send a packet with a destination port number *other than* 21, 22, or 23. The packet should be discarded.
5. To display counter information for the list of filters applied to the input at **ge-1/3/0.0-i** enter the **show firewall filter ge-1/3/0.0-i** operational mode command. The command output displays the number of bytes and packets that match filter terms associated with the following counters:
   - **pkts_FTP-ge-1/3/0.0-i**
   - **pkts_SSH-ge-1/3/0.0-i**
   - **pkts_Telnet-ge-1/3/0.0-i**
   - **pkts_discard-ge-1/3/0.0-i**

Related Documentation

- Understanding Multiple Firewall Filters Applied as a List on page 54
- Guidelines for Applying Multiple Firewall Filters as a List on page 58

Example: Nesting References to Multiple Firewall Filters

This example shows how to configure nested references to multiple firewall filters.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you configure a firewall filter for a match condition and action combination that can be shared among multiple firewall filters. You then configure two firewall filters that reference the first firewall filter. Later, if the common filtering criteria needs to be changed, you would modify only the one shared firewall filter configuration.

Topology

The **common_filter** firewall filter discards packets that have a UDP source or destination port field number of **69**. Both of the two additional firewall filters, **filter1** and **filter2**, reference the **common_filter**.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

- Configure the Nested Firewall Filters on page 203
- Apply Both Nested Firewall Filters to Interfaces on page 204
- Confirm and Commit Your Candidate Configuration on page 204

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter common_filter term common_term from protocol udp
set firewall family inet filter common_filter term common_term from port tftp
set firewall family inet filter common_filter term common_term then discard
set firewall family inet filter filter1 term term1 filter common-filter
set firewall family inet filter filter1 term term2 from address 192.168.0.0/16
set firewall family inet filter filter1 term term2 then reject
set firewall family inet filter filter2 term term1 filter common-filter
set firewall family inet filter filter2 term term2 from protocol udp
set firewall family inet filter filter2 term term2 from port bootps
set firewall family inet filter filter2 term term2 then accept
set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.1/24
set interfaces ge-0/0/0 unit 0 family inet filter input filter1
set interfaces ge-0/0/3 unit 0 family inet address 10.1.3.1/24
set interfaces ge-0/0/0 unit 0 family inet filter input filter2
```

Configure the Nested Firewall Filters

Step-by-Step Procedure

To configure two nested firewall filters that share a common filter:

1. Navigate the CLI to the hierarchy level at which you configure IPv4 firewall filters.

   ```
   [edit]
   user@host# edit firewall family inet
   ```

2. Configure the common filter that will be referenced by multiple other filters.

   ```
   [edit firewall family inet]
   user@host# set filter common_filter term common_term from protocol udp
   user@host# set filter common_filter term common_term from port tftp
   user@host# set filter common_filter term common_term then discard
   ```

3. Configure a filter that references the common filter.

   ```
   [edit firewall family inet]
   user@host# set filter filter1 term term1 filter common-filter
   user@host# set filter filter1 term term2 from address 192.168.0.0/16
   user@host# set filter filter1 term term2 then reject
   ```

4. Configure a second filter that references the common filter.

   ```
   [edit firewall family inet]
   user@host# set filter filter2 term term1 filter common-filter
   user@host# set filter filter2 term term2 from protocol udp
   ```

```
user@host# set filter filter2 term term2 from port bootps
user@host# set filter filter2 term term2 then accept
```

Apply Both Nested Firewall Filters to Interfaces

Step-by-Step Procedure

To apply both nested firewall filters to logical interfaces:

1. Apply the first nested filter to a logical interface input.

```
[edit]
user@host# set interfaces ge-0/0/0 unit 0 family inet address 10.1.0.1/24
user@host# set interfaces ge-0/0/0 unit 0 family inet filter input filter1
```

2. Apply the second nested filter to a logical interface input.

```
[edit]
user@host# set interfaces ge-0/0/3 unit 0 family inet address 10.1.3.1/24
user@host# set interfaces ge-0/0/0 unit 0 family inet filter input filter2
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
    filter common_filter {
        term common_term {
            from {
                protocol udp;
                port tftp;
            }
            then {
                discard;
            }
        }
    }
    filter filter1 {
        term term1 {
            filter common-filter;
        }
        term term2 {
            from {
                address 192.168/16;
            }
            then {
                reject;
            }
        }
    }
    filter filter2 {
        term term1 {
```

```
        filter common-filter;
      }
      term term2 {
        from {
          protocol udp;
          port bootps;
        }
        then {
          accept;
        }
      }
    }
  }
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/0 {
  unit 0 {
    family inet {
      filter {
        input filter1;
      }
      address 10.1.0.1/24;
    }
  }
}
ge-0/0/3 {
  unit 0 {
    family inet {
      filter {
        input filter2;
      }
      address 10.1.3.1/24;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show firewall filter filter1** and **show firewall filter filter2** operational mode commands.

Related Documentation

- Understanding Multiple Firewall Filters in a Nested Configuration on page 51
- Guidelines for Nesting References to Multiple Firewall Filters on page 52

CHAPTER 16

Single Firewall Filter Associated With Multiple Interfaces

Example: Configuring Interface-Specific Firewall Filter Counters

This example shows how to configure and apply an interface-specific standard stateless firewall filter.

Requirements

Interface-specific stateless firewall filters are supported on T Series, M120, M320, and MX Series routers only.

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you create an interface-specific stateless firewall filter that counts and accepts packets with source or destination addresses in a specified prefix and the IP protocol type field set to a specific value.

Topology

You configure the interface-specific stateless firewall filter **filter_s_tcp** to count and accept packets with IP source or destination addresses in the **10.0.0.0/12** prefix and the IP protocol type field set to **tcp** (or the numeric value **6**).

The name of the firewall filter counter is **count_s_tcp**.

You apply the firewall filter to multiple logical interfaces:

- **at-1/1/1.0** input
- **so-2/2/2.2** output

Applying the filter to these two interfaces results in two instances of the filter: **filter_s_tcp-at-1/1/1.0-i** and **filter_s_tcp-so-2/2/2.2-o**, respectively.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter filter_s_tcp interface-specific
set firewall family inet filter filter_s_tcp term 1 from address 10.0.0.0/12
set firewall family inet filter filter_s_tcp term 1 from protocol tcp
set firewall family inet filter filter_s_tcp term 1 then count count_s_tcp
set firewall family inet filter filter_s_tcp term 1 then accept
set interfaces at-1/1/1 unit 0 family inet filter input filter_s_tcp
set interfaces so-2/2/2 unit 2 family inet filter filter_s_tcp
```

Configure the Interface-Specific Firewall Filter

Step-by-Step Procedure

To configure the interface-specific firewall filter:

1. Create the IPv4 firewall filter **filter_s_tcp**.

   ```
   [edit]
   user@host# edit firewall family inet filter filter_s_tcp
   ```

2. Enable interface-specific instances of the filter.

   ```
   [edit firewall family inet filter filter_s_tcp]
   user@host# set interface-specific
   ```

3. Configure the match conditions for the term.

   ```
   [edit firewall family inet filter filter_s_tcp]
   user@host# set term 1 from address 10.0.0.0/12
   user@host# set term 1 from protocol tcp
   ```

4. Configure the actions for the term.

   ```
   [edit firewall family inet filter filter_s_tcp]
   ```

```
user@host# set term 1 then count count_s_tcp
user@host# set term 1 then accept
```

Apply the Interface-Specific Firewall Filter to Multiple Interfaces

Step-by-Step Procedure

To apply the filter **filter_s_tcp** to logical interfaces **at-1/1/1.0** and **so-2/2/2.2**:

1. Apply the interface-specific filter to packets received on logical interface **at-1/1/1.0**.

   ```
   [edit]
   user@host# set interfaces at-1/1/1 unit 0 family inet filter input filter_s_tcp
   ```

2. Apply the interface-specific filter to packets transmitted from logical interface **so-2/2/2.2**.

   ```
   [edit]
   user@host# set interfaces so-2/2/2 unit 2 family inet filter filter_s_tcp
   ```

Confirm Your Candidate Configuration

Step-by-Step Procedure

To confirm your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

   ```
   [edit]
   user@host# show firewall
   family inet {
     filter filter_s_tcp {
       interface-specific;
       term 1 {
         from {
           address {
             10.0.0.0/12;
           }
           protocol tcp;
         }
         then {
           count count_s_tcp;
           accept;
         }
       }
     }
   }
   ```

2. Confirm the configuration of the interfaces by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

   ```
   [edit]
   user@host# show interfaces
   at-1/1/1 {
     unit 0
       family inet {
         filter {
   ```

```
            input filter_s_tcp;
          }
        }
      ]
    }
    so-2/2/2 {
      unit 2
        family inet {
          filter {
            output filter_s_tcp;
          }
        }
      }
    }
```

Clear the Counters and Commit Your Candidate Configuration

Step-by-Step Procedure

To clear the counters and commit your candidate configuration:

1. From operational command mode, use the **clear firewall all** command to clear the statistics for all firewall filters.

   To clear only the counters used in this example, include the interface-specific filter instance names:

   ```
   [edit]
   user@host> clear firewall filter filter_s_tcp-at-1/1/1.0-i
   user@host> clear firewall filter filter_s_tcp-so-2/2/2.2-o
   ```

2. Commit your candidate configuration.

   ```
   [edit]
   user@host# commit
   ```

Verification

Confirm that the configuration is working properly.

- Verifying That the Filter Is Applied to Each of the Multiple Interfaces on page 210
- Verifying That the Counters Are Collected Separately by Interface on page 211

Verifying That the Filter Is Applied to Each of the Multiple Interfaces

Purpose Verify that the filter is applied to each of the multiple interfaces.

Action Run the **show interfaces** command with the **detail** or **extensive** output level.

1. Verify that the filter is applied to the input for **at-1/1/1.0**:

   ```
   user@host> show interfaces at-1/1/1 detail
   Physical interface: at-1/1/1, Enabled, Physical link is Up
     Interface index: 300, SNMP ifIndex: 194, Generation: 183

   ...
     Logical interface at-1/1/1.0 (Index 64) (SNMP ifIndex 204) (Generation 5)
       Flags: Point-To-Point SNMP-Traps 0x4000 Encapsulation: ATM-SNAP
   ```

```
...
    Protocol inet, MTU: 4470, Generation: 13, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Input Filters: filter_s_tcp-at-1/1/1.0-i,,,,,
```

2. Verify that the filter is applied to the output for **so-2/2/2.2**:

```
user@host> show interfaces so-2/2/2 detail
Physical interface: so-2/2/2, Enabled, Physical link is Up
  Interface index: 129, SNMP ifIndex: 502, Generation: 132

...
  Logical interface so-2/2/2.2 (Index 70) (SNMP ifIndex 536) (Generation 135)

    Flags: Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP

...
    Protocol inet, MTU: 4470, Generation: 146, Route table: 0
      Flags: Sendbcast-pkt-to-re
      Output Filters: filter_s_tcp-so-2/2/2.2-o,,,,,
```

Verifying That the Counters Are Collected Separately by Interface

Purpose Make sure that the **count_s_tcp** counters are collected separately for the two logical interfaces.

Action Run the **show firewall** command.

```
user@host> show firewall filter filter_s_tcp
Filter: filter_s_tcp
Counters:
Name                                Bytes              Packets
count_s_tcp-at-1/1/1.0-i              420                    5
count_s_tcp-so-2/2/2.2-o             8888                  101
```

Related Documentation

- Interface-Specific Firewall Filter Instances Overview on page 61
- Statement Hierarchy for Configuring Interface-Specific Firewall Filters on page 269
- Statement Hierarchy for Applying Interface-Specific Firewall Filters on page 270

Example: Filtering Packets Received on an Interface Group

This example shows how to configure a standard stateless firewall filter to match packets tagged for a particular interface group.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you configure two router or switch interfaces to belong to the interface group. You also configure a stateless firewall filter that matches packets that have been tagged as received on that interface group, contain a source or destination address within a particular prefix, and contain a TCP source or destination port of a specific type and port number. The filter counts, logs, and rejects packets that match this criteria. The filter counts, logs, and rejects all other packets. By applying this firewall filter to only one of the two interfaces in the interface group, you can apply the filtering mechanism to all packets input to the two interfaces.

Topology

You configure the interface group number **1** to consist of the management port **fxp0.0** and the loopback port **lo0.0**.

You configure the firewall filter **filter_if_group** to accept only packets from interface group **1**, received from or destined for IP addresses in the 192.168.80.114/32 prefix, and received from or destined for TCP port number **79**.

You apply the firewall filter **filter_if_group** to the router's (or switch's) loopback interface.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

- Configure the Stateless Firewall Filter on page 213
- Assign Interfaces to the Interface Group on page 213
- Apply the Firewall Filter to the Loopback Interface on page 213
- Confirm and Commit Your Candidate Configuration on page 214

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet filter filter_if_group term term1 from interface-group 1
set firewall family inet filter filter_if_group term term1 from address 192.168.80.114/32
set firewall family inet filter filter_if_group term term1 from protocol tcp
set firewall family inet filter filter_if_group term term1 from port finger
set firewall family inet filter filter_if_group term term1 then count if_group_counter1
set firewall family inet filter filter_if_group term term1 then log
set firewall family inet filter filter_if_group term term1 then reject
set firewall family inet filter filter_if_group term term2 then count if_group_counter2
set firewall family inet filter filter_if_group term term2 then log
set firewall family inet filter filter_if_group term term2 then accept
set interfaces fxp0 unit 0 family inet filter group 1
set interfaces fxp0 unit 0 family inet address 192.168.5.38/24
set interfaces lo0 unit 0 family inet filter group 1
set interfaces lo0 unit 0 family inet address 10.0.0.1/32
set interfaces lo0 unit 0 family inet address 192.168.77.1/32
set interfaces lo0 unit 0 family inet filter input filter_if_group
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **filter_if_group**:

1. Create the stateless firewall filter **filter_if_group**.

```
[edit]
user@host# edit firewall family inet filter filter_if_group
```

2. Configure term **term1** to match packets received on interface group **1**, with the source or destination address field in the 192.168.80.114/32 prefix, and with the TCP source or destination port number **79**.

```
[edit firewall family inet filter filter_if_group]
user@host# set term term1 from interface-group 1
user@host# set term term1 from address 192.168.80.114/32
user@host# set term term1 from protocol tcp
user@host# set term term1 from port finger
```

3. Configure term **term1** to count, log, and reject matching packets.

```
[edit firewall family inet filter filter_if_group]
user@host# set term term1 then count if_group_counter1
user@host# set term term1 then log
user@host# set term term1 then reject
```

4. Configure the term **term2** to count, log, and accept all other packets.

```
[edit firewall family inet filter filter_if_group]
user@host# set term term2 then count if_group_counter2
user@host# set term term2 then log
user@host# set term term2 then accept
```

Assign Interfaces to the Interface Group

Step-by-Step Procedure

To assign logical interfaces to the interface group number **1** referenced by the firewall filter match term **term1**:

1. Assign the management port to interface group number **1**.

```
[edit]
user@host# set interfaces fxp0 unit 0 family inet filter group 1
user@host# set interfaces fxp0 unit 0 family inet address 192.168.5.38/24
```

2. Assign a second logical interface to interface group number **1**.

```
[edit]
user@host# set interfaces lo0 unit 0 family inet filter group 1
user@host# set interfaces lo0 unit 0 family inet address 10.0.0.1/32
user@host# set interfaces lo0 unit 0 family inet address 192.168.77.1/32
```

Apply the Firewall Filter to the Loopback Interface

Step-by-Step Procedure

- To apply the firewall filter to the router's (or switch's) loopback interface:

```
[edit]
user@host# set interfaces lo0 unit 0 family inet filter input filter_if_group
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
   filter filter_if_group {
      term term1 {
         from {
            interface-group 1;
            address {
               192.168.80.114/32;
            }
            protocol tcp;
            port finger;
         }
         then {
            count if_group_counter1;
            log;
            reject;
         }
      }
      term term2 {
         then {
            count if_group_counter2;
            log;
            accept;
         }
      }
   }
}
```

2. Confirm the configuration of the interfaces by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
fxp0 {
   unit 0 {
      family inet {
         filter {
            group 1;
         }
         address 192.168.5.38/24;
      }
   }
}
lo0 {
   unit 0 {
      family inet {
```

```
            filter {
              input filter_if_group;
              group 1;
            }
            address 10.0.0.1/32;
            address 192.168.77.1/32;
          }
        }
      }
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To display the firewall filter counters, enter the **show firewall filter filter_if_group** operational mode command:

```
user@host> show firewall filter filter_if_group

Filter: filter_if_group
Counters:
Name                                                Bytes              Packets
if_group_counter1                                       0                    0
if_group_counter2                                 6452105                82667
```

To display the local log of packet headers for packets evaluated by the firewall filter, enter the **show firewall log** operational mode command.

Related Documentation

- Filtering Packets Received on a Set of Interface Groups Overview on page 63
- Statement Hierarchy for Assigning Interfaces to Interface Groups on page 271
- Statement Hierarchy for Configuring a Filter to Match on a Set of Interface Groups on page 271

CHAPTER 17

Filter-Based Tunneling Across IP Networks

Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling

This example shows how to configure a unidirectional generic routing encapsulation (GRE) tunnel to transport IPv6 unicast transit traffic across an IPv4 transport network. To provide network connectivity to the two disjoint IPv6 networks, two MX Series 3D Universal Edge Routers are configured with interfaces that can originate and understand both IPv4 and IPv6 packets. The configuration does not require the creation of tunnel interfaces on Tunnel Services physical interface cards (PICs) or on MPC3E Modular Port Concentrators (MPCs). Instead, you attach firewall filters to Ethernet logical interfaces hosted on Modular Interface Cards (MICs) or MPCs in the two MX Series routers.

Requirements

This example uses the following Juniper Networks hardware and Junos OS software:

- Transport network—An IPv4 network running Junos OS Release 12.3R2 or later.
- PE routers—Two MX80 routers installed as provider edge (PE) routers that connect the IPv4 network to two disjoint IPv6 networks that require a logical path from one network to the other.
- Encapsulating interface—On the encapsulator (the ingress PE router), one Ethernet logical interface configured on the built-in 10-Gigabit Ethernet MIC.
- De-encapsulating interfaces—On the de-encapsulator (the egress PE router), Ethernet logical interfaces configured on three ports of the built-in 10-Gigabit Ethernet MIC.

Before you begin configuring this example:

1. On each PE router, use the **show chassis fpc pic-status** operational mode command to determine which router line cards support filter-based GRE IPv4 tunneling and then use the **interfaces** configuration statement to configure encapsulating and de-encapsulating interfaces.

   - At PE1, the encapsulator, configure *one encapsulating interface* on a supported line card.
   - At PE2, the de-encapsulator, configure *three de-encapsulating interfaces* on a supported line card.

2. Check that IPv4 routing protocols are enabled across the network to support routing paths from the encapsulator to the de-encapsulator.

   Configure routing information by manually adding static routes to route tables or by configuring static or dynamic route-sharing protocols. For more information, see *Transport and Internet Protocols Feature Guide for Routing Devices*.

3. At PE1, *ping* the PE2 IPv4 loopback address to verify that the de-encapsulator is reachable from the encapsulator.

4. At PE2, *ping* the CE2 router IPv6 loopback address to verify that the destination customer edge router is reachable from the de-encapsulator..

   IPv6 routing paths from PE2 to CE2 can be provided by static routes manually added to routing tables or by static or dynamic route-sharing protocols.

   - By default, PE2 forwards packets based on interface routes (direct routes) imported from the primary routing table.
   - As an option, the de-encapsulating filter can specify that the Packet Forwarding Engine uses an alternate routing table to forward payload packets to the destination customer network. In an optional configuration task in this example, you specify an alternate routing table by installing static routes from PE2 to C1 in the routing instance **blue**. You configure the routing information base (RIB) group **blue_group** to specify that the route information of **inet6.0** is shared with **blue.inet6.0**, then you associate the PE2 interfaces with routes stored in both the default routes and the routing instance.

Overview

In this example you configure a unidirectional filter-based GRE IPv4 tunnel from Router PE1 to Router PE2, providing a logical path from IPv6 network C1 to IPv6 network C2.

NOTE: To enable *bidirectional* filter-based GRE tunneling, you must configure a second tunnel in the reverse direction.

As an optional task in this example, you can create a RIB group, which specifies the sharing of routing information (including routes learned from peers, local routes resulting from

the application of protocol policies to the learned routes, and the routes advertised to peers) of multiple routing tables.

Topology

Figure 9 on page 219 shows the path of IPv6 traffic transported from network C1 to network C2, across an IPv4 transport network using a filter-based tunnel from PE1 to PE2 and without requiring tunnel interfaces.

Figure 9: Filter-Based Tunnel from PE1 to PE2 in an IPv4 Network

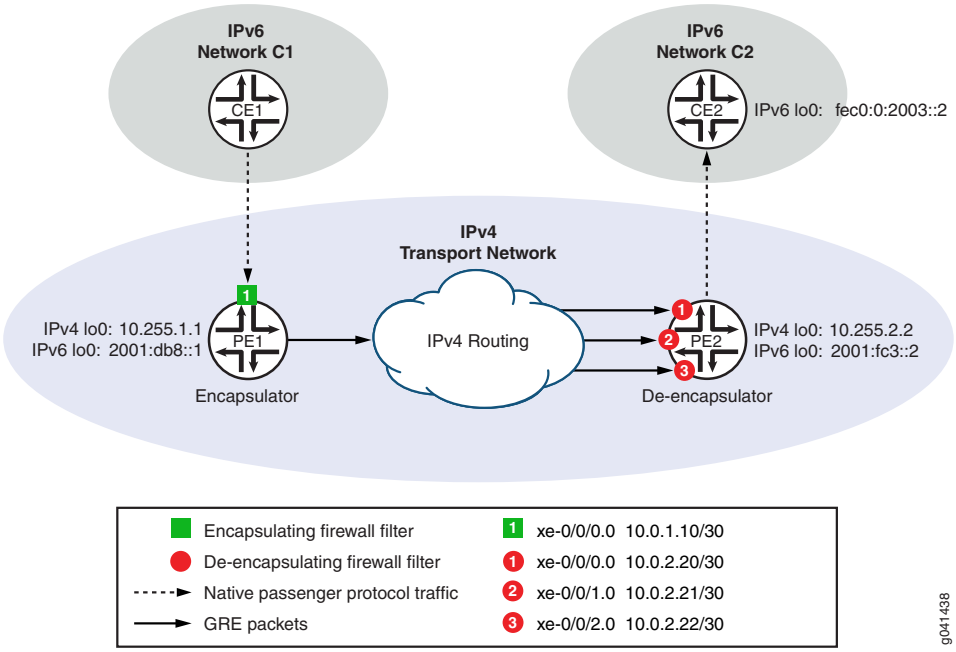


Table 15 on page 220 summarizes the configuration of Router PE1 as the encapsulator. Table 16 on page 220 summarizes the configuration of Router PE2 as the de-encapsulator.

Table 15: Encapsulator Components on PE1

Component	CLI Names		Description
Encapsulator	Device name: IPv4 loopback: IPv6 loopback:	**PE1** **10.255.1.1** **2001:db8::1**	MX80 router installed as an ingress PE router. PE1 connects the IPv4 network the customer edge router CE1 in the IPv6 source network C1.
Encapsulating interface	Interface name: IPv4 address: IPv6 address:	**xe-0/0/0.0** **10.0.1.10/30** **::10.34.1.10/120**	Customer-facing logical interface hosted on a 10-Gigabit Ethernet MIC. CE1 sends this interface IPv6 traffic that originates at end-user hosts and is destined for applications or hosts on the IPv6 destination network C2.
Encapsulation filter	Filter name:	**gre_encap_1**	IPv6 firewall filter whose action causes the Packet Forwarding Engine to encapsulate matched packets using the specified tunnel characteristics. Encapsulation consists of adding a GRE header, adding an IPv4 packet header, and then forwarding the resulting GRE packet through the GRE IPv4 tunnel.
Tunnel source interface	Interface name: IPv4 address:	**xe-0/0/2.0** **10.0.1.12**	Core-facing egress interface to the tunnel.
GRE tunnel template	Tunnel name:	**tunnel_1**	Defines the GRE IPv4 tunnel from Router PE1 (**10.255.1.1**) to Router PE2(**10.255.2.2**), using the tunneling protocol supported on IPv4 (**gre**).

Table 16: De-Encapsulator Components on PE2

Component	CLI Names		Description
De-encapsulator	Device name: IPv4 loopback: IPv6 loopback:	**PE2** **10.255.2.2** **2001:fc3::2**	MX80 router installed as an egress PE router to receive GRE packets forwarded from ingress router PE1 across a GRE IPv4 tunnel.
De-encapsulating interfaces	Interface name: IPv4 address: Interface name: IPv4 address: Interface name: IPv4 address:	**xe-0/0/0.0** **10.0.2.24/30** **xe-0/0/1.0** **10.0.2.21/30** **xe-0/0/2.0** **10.0.2.22/30**	Core-facing ingress logical interfaces hosted on 10-Gigabit Ethernet MICs. The interfaces receive GRE packets routed through the GRE IPv4 tunnel from PE1.
De-encapsulation filter	Filter name:	**gre_decap_1**	IPv4 firewall filter that applies the **decapsulate** action to GRE packets. The filter action causes the Packet Forwarding Engine to de-encapsulate matched packets. De-encapsulation consists of removing the outer GRE header and then forwarding the inner IPv6 payload packet to its original destination on the destination IPv6 network by performing destination lookup on the default routing table.
Tunnel egress interface	Interface name: IPv4 address: IPv6 address:	**xe-0/0/3.0** **10.0.2.23/30** **::20.34.2.23/120**	Customer-facing interface though which the router forwards de-encapsulated IPv6 packets to the destination IPv6 network C2.

Configuration

To transport IPv6 packets from CE1 to CE2 across an IPv4 transport network using a filter-based tunnel from PE1 to PE2 and without configuring tunnel interfaces, perform these tasks:

- Configuring PE1 to Encapsulate IPv6 Packets on page 222
- Configuring PE2 to De-Encapsulate GRE Packets on page 224
- Optional: Configuring PE2 with an Alternate Routing Table on page 226

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Configuring PE1 to Encapsulate IPv6 Packets

```
set interfaces lo0 unit 0 family inet address 10.255.1.1
set interfaces lo0 unit 0 family inet6 address 2001:db8::1
set interfaces xe-0/0/0 unit 0 family inet address 10.0.1.10/30
set interfaces xe-0/0/0 unit 0 family inet6 address ::10.34.1.10/120
set interfaces xe-0/0/0 unit 0 family inet6 filter input gre_encap_1
set interfaces xe-0/0/2 unit 0 family inet address 10.0.1.12/30
set firewall family inet6 filter gre_encap_1 term t1 then count c_gre_encap_1
set firewall family inet6 filter gre_encap_1 term t1 then encapsulate tunnel_1
set firewall tunnel-end-point tunnel_1 ipv4 source-address 10.255.1.1
set firewall tunnel-end-point tunnel_1 ipv4 destination-address 10.255.2.2
set firewall tunnel-end-point tunnel_1 gre
```

Configuring PE2 to De-Encapsulate GRE Packets

```
set interfaces lo0 unit 0 family inet address 10.255.2.2
set interfaces lo0 unit 0 family inet6 address 2001:fc3::2
set interfaces xe-0/0/0 unit 0 family inet address 10.0.2.20/30
set interfaces xe-0/0/1 unit 0 family inet address 10.0.2.21/30
set interfaces xe-0/0/2 unit 0 family inet address 10.0.2.22/30
set interfaces xe-0/0/3 unit 0 family inet address 10.0.2.23/30
set interfaces xe-0/0/3 unit 0 family inet6 address ::20.34.2.23/120
set forwarding-options family inet filter input gre_decap_1
set firewall family inet filter gre_decap_1 term t1 from source-address
10.255.1.1/32
set firewall family inet filter gre_decap_1 term t1 from destination-address
10.255.2.2/32
set firewall family inet filter gre_decap_1 term t1 then count c_gre_decap_1
set firewall family inet filter gre_decap_1 term t1 then decapsulate gre
```

Optional: Configuring PE2 with an Alternate Routing Table

```
set routing-instances blue instance-type forwarding
set routing-instances blue routing-options rib blue.inet6.0 static route 0::/0
next-hop fec0:0:2003::2
set routing-options passive
set routing-options rib inet6.0
set routing-options rib-groups blue_group import-rib inet6.0
set routing-options rib-groups blue_group import-rib blue.inet6.0
set routing-options interface-routes rib-group inet6 blue_group
set firewall family inet filter gre_decap_1 term t1 then decapsulate gre
routing-instance blue
```

Configuring PE1 to Encapsulate IPv6 Packets

Step-by-Step Procedure

To configure Router PE1 to encapsulate IPv6 packets arriving from CE1:

1. Configure the router loopback addresses.

```
[edit]
user@PE1# set interfaces lo0 unit 0 family inet address 10.255.1.1
user@PE1# set interfaces lo0 unit 0 family inet6 address 2001:db8::1
```

2. Configure the encapsulating interface IPv4 and IPv6 addresses and attach the encapsulating filter to the IPv6 input.

```
[edit]
user@PE1# set interfaces xe-0/0/0 unit 0 family inet address 10.0.1.10/30
user@PE1# set interfaces xe-0/0/0 unit 0 family inet6 address ::10.34.1.10/120
user@PE1# set interfaces xe-0/0/0 unit 0 family inet6 filter input gre_encap_1
```

3. Configure the core-facing egress interface to the tunnel.

```
[edit]
user@PE2# set interfaces xe-0/0/2 unit 0 family inet address 10.0.1.12/30
```

4. Define an IPv6 firewall filter that causes the Packet Forwarding Engine to encapsulate all packets.

```
[edit]
user@PE1# set firewall family inet6 filter gre_encap_1 term t1 then count
  c_gre_encap_1
user@PE1# set firewall family inet6 filter gre_encap_1 term t1 then encapsulate
  tunnel_1
```

NOTE: The encapsulate firewall filter action is a *terminating* filter action. A filter-terminating action halts all evaluation of a firewall filter for a specific packet. The router performs the specified action, and no additional terms are examined.

5. Define a GRE IPv4 tunnel template named tunnel_1 that specifies the host IP addresses of the one tunnel source interface and three tunnel destination interfaces.

```
[edit]
user@PE1# set firewall tunnel-end-point tunnel_1 ipv4 source-address 10.255.1.1
user@PE1# set firewall tunnel-end-point tunnel_1 ipv4 destination-address 10.255.2.2
user@PE1# set firewall tunnel-end-point tunnel_1 gre
```

NOTE: You can tunnel multiple but distinct flows from 10.0.1.10 (the tunnel source interface on PE1) to 10.0.2.20 – 10.0.2.22 (the de-encapsulating interfaces on PE2) if you use the GRE option key *number* to uniquely identify each tunnel.

6. If you are done configuring the device, commit the configuration.

```
[edit ]
user@PE1# commit
```

Results From configuration mode, confirm your configuration by entering the **show firewall** and **show interfaces** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Router PE1 Confirm the firewall filter and tunnel template on the encapsulator.

```
user@PE2# show firewall
family inet6 {
  filter gre_encap_1 {
    term t1 {
      then {
        count c_gre_encap_1;
        encapsulate tunnel_1;
      }
    }
  }
}
tunnel-end-point tunnel_1 {
  ipv4 {
    source-address 10.255.1.1;
    destination-address 10.255.2.2;
  }
  gre;
}
```

Router PE1 Confirm the interfaces on the encapsulator.

```
user@PE1# show interfaces
lo0 {
  unit 0 {
    family inet {
      address 10.255.1.1;
    }
    family inet6 {
      address 2001:db8::1;
    }
  }
}
xe-0/0/0 {
  unit 0 {
    family inet {
      address 10.0.1.10/30;
    }
    family inet6 {
      address ::10.34.1.10/120;
      filter input gre_encap_1;
    }
  }
}
xe-0/0/2 {
```

```
    unit 0 {
      family inet {
        address 10.0.1.12/30;
      }
    }
  }
```

Configuring PE2 to De-Encapsulate GRE Packets

Step-by-Step Procedure

To configure Router PE2 to de-encapsulate GRE packets arriving from the IPv4 tunnel:

1. Configure the router loopback address.

   ```
   [edit]
   user@PE2# set interfaces lo0 unit 0 family inet address 10.255.2.2
   user@PE2# set interfaces lo0 unit 0 family inet6 address 2001:fc3::2
   ```

2. Configure the de-encapsulating interfaces.

   ```
   [edit]
   user@PE2# set interfaces xe-0/0/0 unit 0 family inet address 10.0.2.20/30
   user@PE2# set interfaces xe-0/0/1 unit 0 family inet address 10.0.2.21/30
   user@PE2# set interfaces xe-0/0/2 unit 0 family inet address 10.0.2.22/30
   ```

3. Configure the customer-facing egress interface to CE2.

   ```
   [edit]
   user@PE2# set interfaces xe-0/0/3 unit 0 family inet address 10.0.2.23/30
   user@PE2# set interfaces xe-0/0/3 unit 0 family inet6 address ::20.34.2.23/120
   ```

4. Apply the ingress de-encapsulating firewall filter to all forwarded packets.

   ```
   [edit]
   user@PE2# set forwarding-options family inet filter input gre_decap_1
   ```

5. Define IPv4 filter **gre_decap_1**.

   Define an IPv4 filter that de-encapsulates and forwards all GRE packets.

   ```
   [edit]
   user@PE2# set firewall family inet filter gre_decap_1
   ```

6. Configure term **t1** to match packets transported across the tunnel **tunnel_1** defined on Router PE1. The tunnel sends packets from Router PE1 (configured with IPv4 loopback address 10.255.1.1) to Router PE2 (configured with IPv4 loopback address 10.255.2.2).

   ```
   [edit firewall family inet filter gre_decap_1]
   user@PE2# set term t1 from source-address 10.255.1.1
   user@PE2# set term t1 from destination-address 10.255.2.2
   ```

7. Configure term **t1** to count and de-encapsulate matched packets.

   ```
   [edit firewall family inet filter gre_decap_1]
   user@PE2# set term t1 then count c_gre_decap_1
   user@PE2# set term t1 then decapsulate gre
   ```

If the de-encapsulating filter action **decapsulate** references the **blue** routing instance, make sure that the routing instance is configured and that the RIB group **blue_group** defines the sharing of the alternate routes into the primary table.

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE2# commit
```

Results From configuration mode, confirm your configuration by entering the **show firewall**, **show forwarding-options**, and **show interfaces** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Router PE2 Confirm the firewall filter on the de-encapsulator.

```
user@PE2# show firewall
family inet {
  filter gre_decap_1 {
    term t1 {
      from {
        source-address 10.255.1.1;
        destination-address 10.255.2.2;
      }
      then {
        count c_gre_decap_1;
        decapsulate gre routing-instance blue;
      }
    }
  }
}
```

NOTE: If the de-encapsulating filter action **decapsulate** references the **blue** routing instance, make sure that the routing instance is configured and that the RIB group **blue_group** defines the sharing of the alternate routes into the primary table.

Router PE2 Confirm the forwarding options (for attaching the de-encapsulating firewall filter to all input forwarded packets) on the de-encapsulator.

```
user@PE2# show forwarding-options
forwarding-options {
  family inet {
    filter {
      input gre_decap_1;
    }
  }
}
```

Router PE2 Confirm the interfaces on the de-encapsulator.

```
user@PE2# show interfaces
```

```
lo0 {
  unit 0 {
    family inet {
      address 10.255.2.2;
    }
    family inet6 {
      address 2001:fc3::2;
    }
  }
}
xe-0/0/0 {
  unit 0 {
    family inet {
      address 10.0.2.20/30;
      filter input gre_decap_1;
    }
  }
}
xe-0/0/1 {
  unit 0 {
    family inet {
      address 10.0.2.21/30;
      filter input gre_decap_1;
    }
  }
}
xe-0/0/2 {
  unit 0 {
    family inet {
      address 10.0.2.22/30;
      filter input gre_decap_1;
    }
  }
}
xe-0/0/3 {
  unit 0 {
    family inet {
      address 10.0.2.23/30;
    }
    family inet6 {
      address ::20.34.2.23/120;
    }
  }
}
```

Optional: Configuring PE2 with an Alternate Routing Table

Step-by-Step Procedure

To configure Router PE2 with an alternate routing table:

1. Configure the routing instance **blue**, and add static routes to CE2.

```
[edit ]
user@PE2# set routing-instances blue instance-type forwarding
user@PE2# set routing-instances blue routing-options rib blue.inet6.0 static route
  0::/0 next-hop fec0:0:2003::2
```

The Junos OS software generates the routing table **blue.inet6.0** using the routing information learned within the instance.

2. Enable routes to remain in routing and forwarding tables, even if the routes become inactive. This allows a static route to remain in the table if the next hop is unavailable.

```
[edit ]
user@PE2# set routing-options passive
```

3. Create a RIB group by explicitly creating the default routing table.

```
[edit ]
user@PE2# set routing-options rib inet6.0
```

4. Define the RIB group **blue_group**.

```
[edit ]
user@PE2# set routing-options rib-groups blue_group import-rib inet6.0
user@PE2# set routing-options rib-groups blue_group import-rib blue.inet6.0
```

In the **import-rib** statement, specify the primary routing table first.

5. Associate the router interfaces with routing information specified by the RIB group.

```
[edit ]
user@PE2# set routing-options interface-routes rib-group inet6 blue_group
```

6. If you are done configuring the device, commit the configuration.

```
[edit ]
user@PE2# commit
```

Results From configuration mode, confirm your configuration by entering the **show firewall**, **show routing-instances**, and **show routing-options**commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Router PE2 If you configured an alternate routing table on Router PE2, confirm the routing instance configuration.

```
user@PE2# show routing-instances
blue {
   instance-type forwarding;
   routing-options {
      static route 0::/0 next-hop fec0:0:2003::2;
   }
}
```

Router PE2 If you configured an alternate routing table on Router PE2, confirm the RIB group and direct routing configurations.

```
user@PE2# show routing-options
interface-routes {
   rib-group blue_group;
}
passive;
rib inet6.0;
rib-groups {
```

```
    blue_group {
      import-rib [ inet6.0 blue.inet6.0 ];
    }
  }
```

Verification

Confirm that the configurations are working properly.

Verifying Routing Information

Purpose Verify that the direct routes include the alternate routing table information.

Action To perform the verification:

1. (Optional) To verify the routing instance **blue** on PE2, use the **show route instance** operational mode command to display the primary table and number of routes for that routing instance.

```
user@PE2> show route instance blue summary
Instance             Type
        Primary RIB                                  Active/holddown/hidden
blue                 forwarding
        blue.inet6.0                                   2/0/0
```

2. (Optional) To view the routing table associated with the routing instance **blue** on PE2, use the **show route table** operational mode command

```
user@PE2> show route table blue.inet6.0

blue.inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

abcd::192:168:239:17/128
                   *[Direct/0] 00:02:26
                    > via lo0.0
fe80::2a0:a50f:fc64:e032/128
                   *[Direct/0] 00:02:26
                    > via lo0.0
```

3. (Optional) To verify that the alternate routes from routing instance blue have been imported to the PE2 forwarding table, use the **show route forwarding-table** operational mode command to display the contents of the router forwarding table and the routing instance forwarding table.

```
user@PE2> show route forwarding-table blue
Routing table: blue.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   689     1
0.0.0.0/32         perm     0                    dscd   687     1
224.0.0.0/4        perm     0                    mdsc   688     1
```

```
224.0.0.1/32       perm     0 224.0.0.1          mcst   684     1
255.255.255.255/32 perm     0                    bcst   685     1

Routing table: blue.iso
ISO:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   695     1

Routing table: blue.inet6
Internet6:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   701     1
::/128             perm     0                    dscd   699     1
abcd::192:168:239:17/128
                   user     0                    rtbl     2     3
fe80::2a0:a50f:fc64:e032/128
                   user     0                    rtbl     2     3
ff00::/8           perm     0                    mdsc   700     1
ff02::1/128        perm     0 ff02::1            mcst   697     1
```

Verifying Encapsulation on PE1

Purpose Verify the encapsulating interface on PE1.

Action To perform the verification:

1. Use the **show interfaces filters** operational mode command to verify that the encapsulating firewall filter is attached to the ingress of the encapsulating interface.

   ```
   user@PE1> show interfaces filters xe-0/0/0.0
   Interface        Admin  Link  Proto  Input Filter        Output Filter
   xe-0/0/0.0       up     down  inet6  gre_encap_1
   ```

2. Use the **show interfaces** operational mode command to verify that the encapsulating interface is receiving packets.

   ```
   user@PE1> show interfaces xe-0/0/0.0 detail | filter "Ingress traffic"
   ...
   Physical interface: xe-0/0/0, Enabled, Physical link is Up
   ...
     Ingress traffic statistics at Packet Forwarding Engine:
      Input  bytes  :           6970299398                   0 bps
      Input  packets:             81049992                   0 pps
      Drop   bytes  :                    0                   0 bps
      Drop   packets:                    0                   0 pps
   ...
   ```

3. Use the **show firewall filter** operational mode command to verify that ingress passenger protocol traffic triggers the encapsulating filter.

   ```
   user@PE1> show firewall filter gre_encap_1
   Filter: gre_encap_1
   Counters:
   Name                              Bytes             Packets
   c_gre_encap_1               6970299398            81049992
   ```

Meaning If the encapsulating filter is attached to the encapsulating interface, and the encapsulating interface receives passenger protocol traffic, and the firewall filter statistics show that

ingress passenger protocol traffic is being encapsulated, then GRE packets are being forwarded through the tunnel.

Verifying De-Encapsulation on PE2

Purpose Verify the de-encapsulating interfaces on PE2.

Action To perform the verification:

1. On PE1, use the **ping** operational mode command to verify that PE2 is reachable.

```
user@PE1> ping 10.255.2.2
PING 10.255.2.2 (10.255.2.2): 56 data bytes
64 bytes from 10.255.2.2: icmp_seq=0 ttl=64 time=0.576 ms
64 bytes from 10.255.2.2: icmp_seq=1 ttl=64 time=0.269 ms
^C [abort]
```

2. On PE2, use the **show interfaces filter** operational mode command to verify that the de-encapsulating firewall filter is attached to the ingress of the de-encapsulating interfaces.

```
user@PE2> show interfaces filter | match xe-
Interface        Admin  Link  Proto  Input Filter        Output Filter
xe-0/0/0.0       up     down  inet   gre_decap_1
xe-0/0/1.0       up     down  inet   gre_decap_1
xe-0/0/2.0       up     down  inet   gre_decap_1
```

3. On PE2, use the **show interfaces** operational mode command to verify that the de-encapsulating interfaces are receiving packets.

```
user@PE2> show interfaces xe-0/0/0.0 detail | filter "Ingress traffic"
Physical interface: xe-0/0/0, Enabled, Physical link is Up
 ...
  Ingress traffic statistics at Packet Forwarding Engine:
   Input  bytes  :           6970299398                    0 bps
   Input  packets:             81049992                    0 pps
   Drop   bytes  :                    0                    0 bps
   Drop   packets:                    0                    0 pps
 ...
```

```
user@PE2> show interfaces xe-0/0/1.0 detail | filter "Ingress traffic"
Physical interface: xe-0/0/2, Enabled, Physical link is Up
 ...
```

```
user@PE2> show interfaces xe-0/0/2.0 detail | filter "Ingress traffic"
Physical interface: xe-0/0/2, Enabled, Physical link is Up
 ...
```

Depending on how routing is configured and which links are up and which links are down, some of the de-encapsulating interfaces might not be receiving packets although the tunnel is operating properly.

4. On PE2, use the **show firewall filter** operational mode command to verify that ingress GRE traffic triggers the de-encapsulating filter.

```
user@PE2> show firewall filter gre_decap_1
```

```
Filter: gre_decap_1
Counters:
Name                                                Bytes              Packets
c_gre_decap_1                                  6970299398             81049992
```

Meaning The verification confirms the following operational states and activities of the encapsulator:

- PE2 is reachable from the PE1.
- The de-encapsulating filter is attached to the input of all de-encapsulating interfaces.
- The de-encapsulator is receiving traffic at de-encapsulating interfaces as expected.
- GRE packets received at the de-encapsulating interfaces trigger the de-encapsulating firewall filter action.

Related Documentation

- Understanding Filter-Based Tunneling Across IPv4 Networks on page 65
- Interfaces That Support Filter-Based Tunneling Across IPv4 Networks on page 68
- Components of Filter-Based Tunneling Across IPv4 Networks on page 69
- Firewall Filter Terminating Actions on page 384
- **tunnel-end-point on page 313**
- *clear firewall*
- *show chassis fpc*
- *show firewall*
- *show firewall log*
- *show interfaces (10-Gigabit Ethernet)*
- *show interfaces (Aggregated Ethernet)*
- *show interfaces (Gigabit Ethernet)*
- *show route forwarding-table*
- *Junos OS Support for IPv4 Routing Protocols*
- *Junos OS Support for IPv6 Routing Protocols*

CHAPTER 18

Firewall Filters for Filter-Based Forwarding

Example: Configuring Filter-Based Forwarding on the Source Address

This example shows how to configure filter-based forwarding. The filter classifies packets to determine their forwarding path within the ingress routing device.

Requirements

In this example, no special configuration beyond device initialization is required.

Overview

Filter-based forwarding is supported for IP version 4 (IPv4) and IP version 6 (IPv6).

Use filter-based forwarding for service provider selection when customers have Internet connectivity provided by different ISPs yet share a common access layer. When a shared media (such as a cable modem) is used, a mechanism on the common access layer looks at Layer 2 or Layer 3 addresses and distinguishes between customers. You can use filter-based forwarding when the common access layer is implemented using a combination of Layer 2 switches and a single router.

With filter-based forwarding, all packets received on an interface are considered. Each packet passes through a filter that has match conditions. If the match conditions are met for a filter and you have created a routing instance, filter-based forwarding is applied to a packet. The packet is forwarded based on the next hop specified in the routing instance. For static routes, the next hop can be a specific LSP.

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

To configure filter-based forwarding, perform the following tasks:

- Create a match filter on an ingress router. To specify a match filter, include the **filter *filter-name*** statement at the **[edit firewall]** hierarchy level. A packet that passes through the filter is compared against a set of rules to classify it and to determine its membership in a set. Once classified, the packet is forwarded to a routing table specified in the accept action in the filter description language. The routing table then forwards the packet to the next hop that corresponds to the destination address entry in the table.
- Create routing instances that specify the routing table(s) to which a packet is forwarded, and the destination to which the packet is forwarded at the **[edit routing-instances]** hierarchy level. For example:

```
[edit]
routing-instances {
  routing-table-name1 {
    instance-type forwarding;
    routing-options {
      static {
        route 0.0.0.0/0 nexthop 10.0.0.1;
      }
    }
  }
  routing-table-name2 {
    instance-type forwarding;
    routing-options {
      static {
        route 0.0.0.0/0 nexthop 10.0.0.2;
      }
    }
  }
}
```

- Create a routing table group that adds interface routes to the forwarding routing instances used in filter-based forwarding (FBF), as well as to the default routing instance inet.0. This part of the configuration resolves the routes installed in the routing instances to directly connected next hops on that interface. Create the routing table group at the **[edit routing-options]** hierarchy level.

NOTE: Specify inet.0 as one of the routing instances that the interface routes are imported into. If the default instance inet.0 is not specified, interface routes are not imported into the default routing instance.

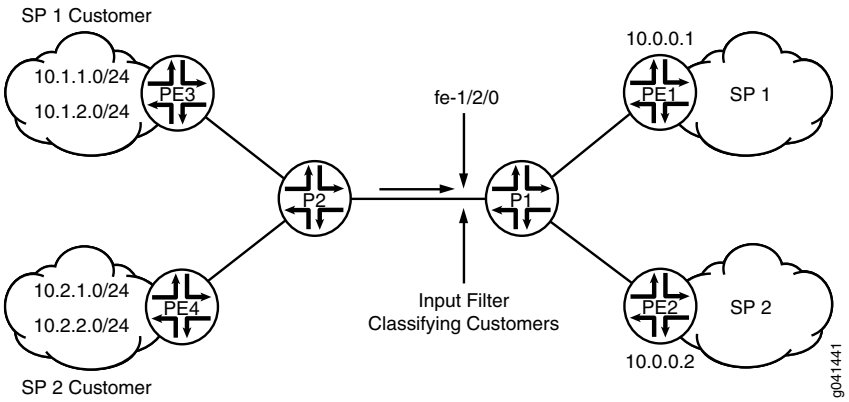
This example shows a packet filter that directs customer traffic to a next-hop router in the domains, SP 1 or SP 2, based on the packet's source address.

If the packet has a source address assigned to an SP 1 customer, destination-based forwarding occurs using the sp1-route-table.inet.0 routing table. If the packet has a source address assigned to an SP 2 customer, destination-based forwarding occurs using the sp2-route-table.inet.0 routing table. If a packet does not match either of these conditions, the filter accepts the packet, and destination-based forwarding occurs using the standard inet.0 routing table.

Figure 10 on page 235 shows the topology used in this example.

On Device P1, an input filter classifies packets received from Device PE3 and Device PE4. The packets are routed based on the source addresses. Packets with source addresses in the 10.1.1.0/24 and 10.1.2.0/24 networks are routed to Device PE1. Packets with source addresses in the 10.2.1.0/24 and 10.2.2.0/24 networks are routed to Device PE2.

Figure 10: Filter-Based Forwarding



To establish connectivity, OSPF is configured on all of the interfaces. For demonstration purposes, loopback interface addresses are configured on the routing devices to represent networks in the clouds.

The "CLI Quick Configuration" on page 235 section shows the entire configuration for all of the devices in the topology. The "Configuring the Firewall Filter" on page 237 and "Configuring the Routing Instances on the Device P1" on page 238 sections shows the step-by-step configuration of the ingress routing device, Device P1.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Device P1

```
set firewall filter classify-customers term sp1-customers from source-address 10.1.1.0/24
set firewall filter classify-customers term sp1-customers from source-address 10.1.2.0/24
set firewall filter classify-customers term sp1-customers then log
set firewall filter classify-customers term sp1-customers then routing-instance
  sp1-route-table
set firewall filter classify-customers term sp2-customers from source-address 10.2.1.0/24
set firewall filter classify-customers term sp2-customers from source-address 10.2.2.0/24
set firewall filter classify-customers term sp2-customers then log
```

```
set firewall filter classify-customers term sp2-customers then routing-instance
  sp2-route-table
set firewall filter classify-customers term default then accept
set interfaces fe-1/2/0 unit 0 family inet filter input classify-customers
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.10/30
set interfaces fe-1/2/1 unit 0 family inet address 172.16.0.13/30
set interfaces fe-1/2/2 unit 0 family inet address 172.16.0.17/30
set protocols ospf rib-group fbf-group
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances sp1-route-table instance-type forwarding
set routing-instances sp1-route-table routing-options static route 0.0.0.0/0 next-hop
  172.16.0.13
set routing-instances sp2-route-table instance-type forwarding
set routing-instances sp2-route-table routing-options static route 0.0.0.0/0 next-hop
  172.16.0.17
set routing-options rib-groups fbf-group import-rib inet.0
set routing-options rib-groups fbf-group import-rib sp1-route-table.inet.0
set routing-options rib-groups fbf-group import-rib sp2-route-table.inet.0
```

Device P2

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.2/30
set interfaces fe-1/2/1 unit 0 family inet address 172.16.0.6/30
set interfaces fe-1/2/2 unit 0 family inet address 172.16.0.9/30
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE1

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.14/30
set interfaces lo0 unit 0 family inet address 1.1.1.1/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE2

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.18/30
set interfaces lo0 unit 0 family inet address 2.2.2.2/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE3

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.1/30
set interfaces lo0 unit 0 family inet address 10.1.1.1/32
set interfaces lo0 unit 0 family inet address 10.1.2.1/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE4

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.5/30
set interfaces lo0 unit 0 family inet address 10.2.1.1/32
set interfaces lo0 unit 0 family inet address 10.2.2.1/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Device PE4

```
set interfaces fe-1/2/0 unit 0 family inet address 172.16.0.5/30
set interfaces lo0 unit 0 family inet address 10.2.1.1/32
set interfaces lo0 unit 0 family inet address 10.2.2.1/32
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
```

Configuring the Firewall Filter

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the firewall filter on the main router:

1. Configure the source addresses for SP1 customers.

   ```
   [edit firewall filter classify-customers term sp1-customers]
   user@host# set from source-address 10.1.1.0/24
   user@host# set from source-address 10.1.2.0/24
   ```

2. Configure the actions that are taken when packets are received with the specified source addresses.

   To track the action of the firewall filter, a log action is configured. The sp1-route-table.inet.0 routing table on Device P1 routes the packets.

   ```
   [edit firewall filter classify-customers term sp1-customers]
   user@host# set then log
   user@host# set then routing-instance sp1-route-table
   ```

3. Configure the source addresses for SP2 customers.

   ```
   [edit firewall filter classify-customers term sp2-customers]
   user@host# set from source-address 10.2.1.0/24
   user@host# set from source-address 10.2.2.0/24
   ```

4. Configure the actions that are taken when packets are received with the specified source addresses.

   To track the action of the firewall filter, a log action is configured. The sp2-route-table.inet.0 routing table on Device P1 routes the packet.

   ```
   [edit firewall filter classify-customers term sp2-customers]
   user@host# set then log
   user@host# set then routing-instance sp2-route-table
   ```

5. Configure the action to take when packets are received from any other source address.

   All of these packets are simply accepted and routed using the default IPv4 unicast routing table, inet.0.

   ```
   [edit firewall filter classify-customers term default]
   user@host# set then accept
   ```

6. Configure the action to take when packets are received from any other source address.

   All of these packets are simply accepted and routed using the default IPv4 unicast routing table, inet.0.

   ```
   [edit firewall filter classify-customers term default]
   user@host# set then accept
   ```

Configuring the Routing Instances on the Device P1

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure the routing instances:

1. Configure the interfaces.

```
[edit interfaces fe-1/2/0]
user@host# set unit 0 family inet address 172.16.0.10/30

[edit interfaces fe-1/2/1]
user@host# set unit 0 family inet address 172.16.0.13/30

[edit interfaces fe-1/2/2]
user@host# set unit 0 family inet address 172.16.0.17/30
```

2. Assign the **classify-customers** firewall filter to router interface fe-1/2/0.0 as an input packet filter.

```
[edit interfaces fe-1/2/0]
user@host# set unit 0 family inet filter input classify-customers
```

3. Configure connectivity, using either a routing protocol or static routing.

   As a best practice, disable routing on the management interface.

```
[edit protocols ospf area 0.0.0.0]
user@host# set interface all
user@host# set interface fxp0.0 disable
```

4. Create the routing instances.

   These routing instances are referenced in the **classify-customers** firewall filter.

   The forwarding instance type provides support for filter-based forwarding, where interfaces are not associated with instances. All interfaces belong to the default instance, in this case Device P1.

```
[edit routing-instances]
user@host# set sp1-route-table instance-type forwarding

user@host# set sp2-route-table instance-type forwarding
```

5. Resolve the routes installed in the routing instances to directly connected next hops.

```
[edit routing-instances sp1-route-table routing-options]
user@host# set static route 0.0.0.0/0 next-hop 172.16.0.13

user@host# set static route 0.0.0.0/0 next-hop 172.16.0.17
```

6. Group together the routing tables to form a routing table group.

   The first routing table, inet.0, is the primary routing table, and the additional routing tables are the secondary routing tables.

   The primary routing table determines the address family of the routing table group, in this case IPv4.

   ```
   [edit routing-options]
   user@host# set rib-groups fbf-group import-rib inet.0
   user@host# set rib-groups fbf-group import-rib sp1-route-table.inet.0
   user@host# set rib-groups fbf-group import-rib sp2-route-table.inet.0
   ```

7. Apply the routing table group to OSPF.

   This causes the OSPF routes to be installed into all the routing tables in the group.

   ```
   [edit protocols ospf]
   user@host# set rib-group fbf-group
   ```

8. Apply the routing table group to OSPF.

   This causes the OSPF routes to be installed into all the routing tables in the group.

   ```
   [edit protocols ospf]
   user@host# set rib-group fbf-group
   ```

9. If you are done configuring the device, commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

Results

Confirm your configuration by issuing the **show interfaces**, **show firewall**, **show protocols**, **show routing-instances**, and **show routing-options** commands.

```
user@host# show interfaces
fe-1/2/0 {
    unit 0 {
        family inet {
            filter {
                input classify-customers;
            }
            address 172.16.0.10/30;
        }
    }
}
fe-1/2/1 {
    unit 0 {
        family inet {
            address 172.16.0.13/30;
        }
    }
}
fe-1/2/2 {
    unit 0 {
        family inet {
```

```
        address 172.16.0.17/30;
      }
    }
}

user@host# show firewall
filter classify-customers {
    term sp1-customers {
        from {
            source-address {
                10.1.1.0/24;
                10.1.2.0/24;
            }
        }
        then {
            log;
            routing-instance sp1-route-table;
        }
    }
    term sp2-customers {
        from {
            source-address {
                10.2.1.0/24;
                10.2.2.0/24;
            }
        }
        then {
            log;
            routing-instance sp2-route-table;
        }
    }
    term default {
        then accept;
    }
}

user@host# show protocols
ospf {
    rib-group fbf-group;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}

user@host# show routing-instances
sp1-route-table {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 172.16.0.13;
        }
    }
}
sp2-route-table {
```

```
    instance-type forwarding;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 172.16.0.17;
      }
    }
}
```

```
user@host# show routing-options
rib-groups {
    fbf-group {
        import-rib [ inet.0 sp1-route-table.inet.0 sp2-route-table.inet.0 ];
    }
}
```

Verification

Confirm that the configuration is working properly.

Pinging with Specified Source Addresses

Purpose Send some ICMP packets across the network to test the firewall filter.

Action

1. Run the **ping** command, pinging the lo0.0 interface on Device PE1.

   The address configured on this interface is 1.1.1.1.

   Specify the source address 10.1.2.1, which is the address configured on the lo0.0 interface on Device PE3.

   ```
   user@PE3> ping 1.1.1.1 source 10.1.2.1
   PING 1.1.1.1 (1.1.1.1): 56 data bytes
   64 bytes from 1.1.1.1: icmp_seq=0 ttl=62 time=1.444 ms
   64 bytes from 1.1.1.1: icmp_seq=1 ttl=62 time=2.094 ms
   ^C
   --- 1.1.1.1 ping statistics ---
   2 packets transmitted, 2 packets received, 0% packet loss
   round-trip min/avg/max/stddev = 1.444/1.769/2.094/0.325 ms
   ```

2. Run the **ping** command, pinging the lo0.0 interface on Device PE2.

   The address configured on this interface is 2.2.2.2.

   Specify the source address 10.2.1.1, which is the address configured on the lo0.0 interface on Device PE4.

   ```
   user@PE4> ping 2.2.2.2 source 10.2.1.1
   PING 2.2.2.2 (2.2.2.2): 56 data bytes
   64 bytes from 2.2.2.2: icmp_seq=0 ttl=62 time=1.473 ms
   64 bytes from 2.2.2.2: icmp_seq=1 ttl=62 time=1.407 ms
   ^C
   --- 2.2.2.2 ping statistics ---
   2 packets transmitted, 2 packets received, 0% packet loss
   round-trip min/avg/max/stddev = 1.407/1.440/1.473/0.033 ms
   ```

Meaning Sending these pings activates the firewall filter actions.

Verifying the Firewall Filter

Purpose Make sure the firewall filter actions take effect.

Action

1. Run the **show firewall log** command on Device P1.

```
user@P1> show firewall log
Log :
Time      Filter    Action Interface     Protocol        Src Addr
Dest Addr
13:52:20  pfe       A      fe-1/2/0.0    ICMP            10.2.1.1
2.2.2.2
13:52:19  pfe       A      fe-1/2/0.0    ICMP            10.2.1.1
2.2.2.2
13:51:53  pfe       A      fe-1/2/0.0    ICMP            10.1.2.1
1.1.1.1
13:51:52  pfe       A      fe-1/2/0.0    ICMP            10.1.2.1
1.1.1.1
```

Related Documentation

- *Example: Configuring Multitopology Routing Based on Applications*
- *Copying and Redirecting Traffic with Port Mirroring and Filter-Based Forwarding*
- *Using Filter-Based Forwarding to Export Monitored Traffic to Multiple Destinations*
- Filter-Based Forwarding Overview on page 75

Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address

Understanding Filter-Based Forwarding to a Specific Outgoing Interface or Destination IP Address

Policy-based routing (also known as filter-based forwarding) refers to the use of firewall filters that are applied to an interface to match certain IP header characteristics and to route only those matching packets differently than the packets would normally be routed.

Starting in Junos OS Release 12.2, you can use **then next-interface**, **then next-ip**, or **then next-ip6** as an action in a firewall filter.

For example:

```
from {
  set of match conditions
}
then {
  IP-address (or)
  IPv6-address (or)
```



```
    Interface name
}
```

The set of match conditions can be as follows:

- Layer-3 properties (for example, the source or destination IP address or the TOS byte)
- Layer-4 properties (for example, the source or destination port)

The route for the given IPv4 or IPv6 address has to be present in the routing table for policy-based routing to take effect. Similarly, the route through the given interface has to be present in the forwarding table for **next-interface** action to take effect. This can be achieved by configuring an interior gateway protocol (IGP), such as OSPF or IS-IS, to advertise Layer 3 routes.

The firewall filter matches the conditions and forwards the packet to one of the following:

- An IPv4 address (using the **next-ip** firewall filter action)
- An IPv6 address (using the **next-ip6** firewall filter action)
- An interface (using the **next-interface** firewall filter action)

Suppose, for example, that you want to offer services to your customers, and the services reside on different servers. An example of a service might be hosted DNS or hosted FTP. As customer traffic arrives at the Juniper Networks routing device, you can use filter-based forwarding to send traffic to the servers by applying a match condition on a MAC address or an IP address or simply an incoming interface and send the packets to a certain outgoing interface that is associated with the appropriate server. Some of your destinations might be IPv4 or IPv6 addresses, in which case the **next-ip** or **next-ip6** action is useful.

Optionally, you can associate the outgoing interfaces or IP addresses with routing instances.

For example:

```
firewall {
    filter filter1 {
        term t1 {
            from {
                source-address {
                    10.1.1.3/32;
                }
            }
            then {
                next-interface {
                    xe-0/1/0.1;
                    routing-instance rins1;
                }
            }
        }
        term t2 {
            from {
                source-address {
                    10.1.1.4/32;
                }
```

```
        }
        then {
          next-interface {
            xe-0/1/0.2;
            routing-instance rins2;
          }
        }
      }
    }
  }
  routing-instances {
    rins1 {
      instance-type virtual-router;
      interface xe-0/1/0.1;
    }
    rins2 {
      instance-type virtual-router;
      interface xe-0/1/0.2;
    }
  }
```

Example: Configuring Filter-Based Forwarding to a Specific Outgoing Interface

This example shows how to use **then next-interface** as an action in a firewall filter.

Requirements

This example has the following hardware and software requirements:

- MX Series 3D Universal Edge Router as the routing device with the firewall filter configured.
- Junos OS Release 12.2 running on the routing device with the firewall filter configured.
- The filter with the **next-interface** (or **next-ip**) action can only be applied to an interface that is hosted on a Trio MPC. If you apply the filter to an I-chip based DPC, the commit operation fails.
- The outgoing interface referred to in the **next-interface** ***interface-name*** action can be hosted on a Trio MPC or an I-chip based DPC.

Overview

In this example, Device R1 has two loopback interface addresses configured: 172.16.1.1 and 172.16.2.2.

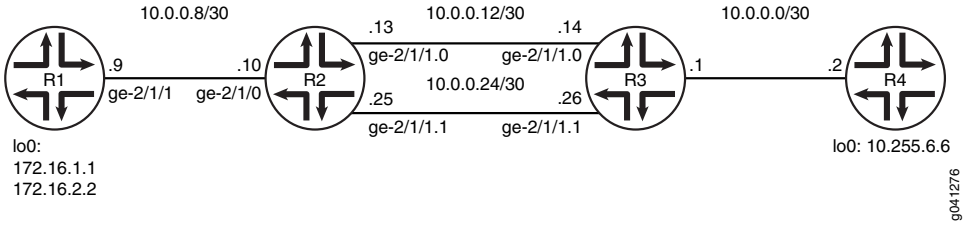
On Device R2, a firewall filter has multiple terms configured. Each term matches one of the source addresses in incoming traffic, and routes the traffic to specified outgoing

interfaces. The outgoing interfaces are configured as VLAN-tagged interfaces between Device R2 and Device R3.

IS-IS is used for connectivity among the devices.

Figure 11 on page 245 shows the topology used in this example.

Figure 11: Filter-Based Forwarding to Specified Outgoing Interfaces



This example shows the configuration on Device R2.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Device R2

```
set interfaces ge-2/1/0 unit 0 family inet filter input filter1
set interfaces ge-2/1/0 unit 0 family inet address 10.0.0.10/30
set interfaces ge-2/1/0 unit 0 description to-R1
set interfaces ge-2/1/0 unit 0 family iso
set interfaces ge-2/1/1 vlan-tagging
set interfaces ge-2/1/1 description to-R3
set interfaces ge-2/1/1 unit 0 vlan-id 1001
set interfaces ge-2/1/1 unit 0 family inet address 10.0.0.13/30
set interfaces ge-2/1/1 unit 0 family iso
set interfaces ge-2/1/1 unit 1 vlan-id 1002
set interfaces ge-2/1/1 unit 1 family inet address 10.0.0.25/30
set interfaces ge-2/1/1 unit 1 family iso
set interfaces lo0 unit 0 family inet address 10.255.4.4/32
set interfaces lo0 unit 0 family iso address 49.0001.0010.0000.0404.00
set firewall family inet filter filter1 term t1 from source-address 172.16.1.1/32
set firewall family inet filter filter1 term t1 then next-interface ge-2/1/1.0
set firewall family inet filter filter1 term t2 from source-address 172.16.2.2/32
set firewall family inet filter filter1 term t2 then next-interface ge-2/1/1.1
set protocols isis interface all level 1 disable
set protocols isis interface fxp0.0 disable
set protocols isis interface lo0.0
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319 in the *CLI User Guide*.

To configure Device R2:

1. Configure the interfaces.

```
[edit interfaces]
user@R2# set ge-2/1/0 unit 0 family inet filter input filter1
user@R2# set ge-2/1/0 unit 0 family inet address 10.0.0.10/30
user@R2# set ge-2/1/0 unit 0 description to-R1
user@R2# set ge-2/1/0 unit 0 family iso

user@R2# set ge-2/1/1 vlan-tagging
user@R2# set ge-2/1/1 description to-R3

user@R2# set ge-2/1/1 unit 0 vlan-id 1001
user@R2# set ge-2/1/1 unit 0 family inet address 10.0.0.13/30
user@R2# set ge-2/1/1 unit 0 family iso

user@R2# set ge-2/1/1 unit 1 vlan-id 1002
user@R2# set ge-2/1/1 unit 1 family inet address 10.0.0.25/30
user@R2# set ge-2/1/1 unit 1 family iso

user@R2# set lo0 unit 0 family inet address 10.255.4.4/32
user@R2# set lo0 unit 0 family iso address 49.0001.0010.0000.0404.00
```

2. Configure the firewall filter.

```
[edit firewall family inet filter filter1]
user@R2# set term t1 from source-address 172.16.1.1/32
user@R2# set term t1 then next-interface ge-2/1/1.0

user@R2# set term t2 from source-address 172.16.2.2/32
user@R2# set term t2 then next-interface ge-2/1/1.1
```

3. Enable IS-IS on the interfaces.

```
[edit protocols is-is]
user@R2# set interface all level 1 disable
user@R2# set interface fxp0.0 disable
user@R2# set interface lo0.0
```

Results From configuration mode, confirm your configuration by entering the **show interfaces**, **show firewall**, and **show protocols** commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@R2# show interfaces
ge-2/1/0 {
    unit 0 {
        description to-R1;
        family inet {
            filter {
                input filter1;
            }
            address 10.0.0.10/30;
        }
        family iso;
    }
}
ge-2/1/1 {
    description to-R3;
```

```
    vlan-tagging;
    unit 0 {
      vlan-id 1001;
      family inet {
        address 10.0.0.13/30;
      }
      family iso;
    }
    unit 1 {
      vlan-id 1002;
      family inet {
        address 10.0.0.25/30;
      }
      family iso;
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.255.4.4/32;
      }
      family iso {
        address 49.0001.0010.0000.0404.00;
      }
    }
  }
```

```
user@R2# show firewall
family inet {
  filter filter1 {
    term t1 {
      from {
        source-address {
          172.16.1.1/32;
        }
      }
      then {
        next-interface {
        ge-2/1/1.0;
      }
    }
    term t2 {
      from {
        source-address {
          172.16.2.2/32;
        }
      }
      then {
        next-interface {
          ge-2/1/1.1;
        }
      }
    }
  }
}
```

```
user@R2# show protocols
```

```
isis {
  interface all {
    level 1 disable;
  }
  interface fxp0.0 {
    disable;
  }
  interface lo0.0;
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

Confirm that the configuration is working properly.

Checking the Paths Used

Purpose Make sure that the expected paths are used when sending traffic from Device R1 to Device R4.

Action On Device R1, enter the **traceroute** command.

```
user@R1> traceroute 10.255.6.6 source 172.16.1.1
traceroute to 10.255.6.6 (10.255.6.6) from 172.16.1.1, 30 hops max, 40 byte packets

 1  10.0.0.10 (10.0.0.10)  0.976 ms  0.895 ms  0.815 ms
 2  10.0.0.14 (10.0.0.14)  0.868 ms  0.888 ms  0.813 ms
 3  10.255.6.6 (10.255.6.6)  1.715 ms  1.442 ms  1.382 ms

user@R1> traceroute 10.255.6.6 source 172.16.2.2
traceroute to 10.255.6.6 (10.255.6.6) from 172.16.2.2, 30 hops max, 40 byte packets

 1  10.0.0.10 (10.0.0.10)  0.973 ms  0.907 ms  0.782 ms
 2  10.0.0.26 (10.0.0.26)  0.844 ms  0.890 ms  0.852 ms
 3  10.255.6.6 (10.255.6.6)  1.384 ms  1.516 ms  1.462 ms
```

Meaning The output shows that the second hop changes, depending on the source address used in the **traceroute** command.

To verify this feature, a traceroute operation is performed on Device R1 to Device R4. When the source IP address is 172.16.1.1, packets are forwarded out the ge-2/1/1.0 interface on Device R2. When the source IP address is 172.16.2.2, packets are forwarded out the ge-2/1/1.1 interface on Device R2.

Related Documentation

- Example: Configuring Filter-Based Forwarding on Logical Systems on page 173
- Example: Configuring Filter-Based Forwarding on the Source Address on page 233
- Firewall Filter Nonterminating Actions on page 377

CHAPTER 19

Firewall Filters for CoS Multifield Classification or Rate Limiting

Example: Configuring a Filter to Set the DSCP Bit to Zero

This example shows how to configure a standard stateless firewall filter based on the Differentiated Services code point (DSCP).

Requirements

No special configuration beyond device initialization is required before configuring this example.

Overview

In this example, you use a stateless firewall filter to match packets on DSCP bit patterns. If the DSCP is **2**, the packet is classified to the **best-effort** forwarding class, and the DSCP is set to **0**. If the DSCP is **3**, the packet is classified to the **best-effort** forwarding class.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

CLI Quick Configuration

To quickly configure this example, copy the following configuration commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall filter filter1 term 1 from dscp 2
set firewall filter filter1 term 1 then forwarding-class best-effort
set firewall filter filter1 term 1 then dscp 0
set firewall filter filter1 term 2 from dscp 3
set firewall filter filter1 term 2 then forwarding-class best-effort
set interfaces so-0/1/0 unit 0 family inet filter input filter1
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **filter1**:

1. Create the stateless firewall filter.

   ```
   [edit]
   user@host# edit firewall filter filter1
   ```

2. Configure the first term to match a packet with a DSCP of **2**, change the DSCP to **0**, and classify the packet to the **best-effort** forwarding class.

   ```
   [edit firewall filter filter1]
   user@host# set term 1 from dscp 2
   user@host# set term 1 then forwarding-class best-effort
   user@host# set term 1 then dscp 0
   ```

3. Configure the other term to match a packet with a DSCP of **3** and classify the packet to the **best-effort** forwarding class.

   ```
   [edit firewall filter filter1]
   user@host# set term 2 from dscp 3
   user@host# set term 2 then forwarding-class best-effort
   ```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the stateless firewall filter to the logical interface corresponding to the VPN routing and forwarding (VRF) instance:

1. Configure the logical interface to which you will apply the stateless firewall filter.

   ```
   [edit]
   user@host# edit interfaces so-0/1/0 unit 0 family inet
   ```

2. Apply the stateless firewall filter to the logical interface.

   ```
   [ input filter1]
   user@host# set filter input filter1
   ```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
filter filter1 {
  term term1 {
    from {
      dscp 2;
    }
    then {
      forwarding-class best-effort;
      dscp 0;
    }
  }
  term term2 {
    from {
      dscp 3;
    }
    then {
      forwarding-class best-effort;
    }
  }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
so-0/1/0 {
  unit 0 {
    family inet {
      filter input filter1;
    }
  }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the following operational mode commands:

- **show class-of-service**—Displays the entire class-of-service (CoS) configuration, including system-chosen defaults.
- **show class-of-service classifier type dscp**—Displays only the classifiers of the DSCP for IPv4 type.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Example: Configuring a Filter to Count and Sample Accepted Packets on page 155

Example: Configuring a Rate-Limiting Filter Based on Destination Class

This example shows how to configure a rate-limiting stateless firewall filter.

Requirements

No special configuration beyond device initialization is required before configuring this example.

Before you begin, configure the destination class **class1**.

Overview

In this example, you use a stateless firewall filter to set rate limits based on a destination class.

To activate a policer from within a stateless firewall filter configuration:

- Create a template for the policer by including the **policer** ***policer-name*** statement.
- Reference the policer in a filter term that specifies the policer in the **policer** ***policer-name*** nonterminating action.

You can also activate a policer by including the **policer (input | output)** ***policer-template-name*** statement at a logical interface.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall filter rl_dclass1 policer police_class1 if-exceeding bandwidth-limit 25
set firewall filter rl_dclass1 policer police_class1 if-exceeding burst-size-limit 1000
set firewall filter rl_dclass1 policer police_class1 then discard
set firewall filter rl_dclass1 term term1 from destination-class class1
set firewall filter rl_dclass1 term term1 then policer police_class1
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
set interfaces ge-0/0/1 unit 0 family inet filter input ospf_or_131
```

Configure the Stateless Firewall Filter

Step-by-Step Procedure

To configure the stateless firewall filter **rl_dclass1** with policer **police_class1** for destination class **class1**:

1. Create the stateless firewall filter **rl_dclass1**.

   ```
   [edit]
   user@host# edit firewall filter rl_dclass1
   ```

2. Configure the policer template **police_class1**.

   ```
   [edit firewall filter rl_dclass1]
   user@host# set policer police_class1 if-exceeding bandwidth-limit 25
   user@host# set policer police_class1 if-exceeding burst-size-limit 1000
   user@host# set policer police_class1 then discard
   ```

3. Configure a filter term that uses policer **police_class1** to rate-limit traffic for destination class **class1**.

   ```
   [edit firewall filter rl_dclass1]
   user@host# set term term1 from destination-class class1
   user@host# set term term1 then policer police_class1
   ```

Apply the Stateless Firewall Filter to a Logical Interface

Step-by-Step Procedure

To apply the filter **rl_dclass1** to a logical interface:

1. Configure the logical interface to which you will apply the stateless firewall filter.

   ```
   [edit]
   user@host# edit interfaces ge-0/0/1 unit 0 family inet
   ```

2. Configure the interface address for the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set address 10.1.2.3/30
```

3. Apply the stateless firewall filter to the logical interface.

```
[edit interfaces ge-0/0/1 unit 0 family inet]
user@host# set filter input rl_dclass1
```

Confirm and Commit Your Candidate Configuration

Step-by-Step Procedure

To confirm and then commit your candidate configuration:

1. Confirm the configuration of the stateless firewall filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
filter rl_dclass1 {
    policer police_class1 {
        if-exceeding {
            bandwidth-limit 25;
            burst-size-limit 1000;
        }
        then {
            discard;
        }
    }
    term term1 {
        from {
            destination-class class1;
        }
        then {
            policer police_class1;
        }
    }
}
```

2. Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input rl_dclass1;
            }
            address 10.1.2.3/30;
        }
    }
}
```

3. If you are done configuring the device, commit your candidate configuration.

```
[edit]
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the **show class-of-service ge-0/0/1** operational mode command.

Related Documentation

- Understanding How to Use Firewall Filters on page 29
- Filtering Packets Received on an Interface Set Overview on page 63
- Statement Hierarchy for Defining an Interface Set on page 273
- Statement Hierarchy for Configuring a Filter to Match on an Interface Set on page 273
- Example: Filtering Packets Received on an Interface Set on page 116

CHAPTER 20

Service Filter Configuration

Example: Configuring and Applying Service Filters

This example shows how to configure and apply service filters.

Requirements

This example use the logical interface **xe-0/1/0.0** on any of the following hardware components:

- Adaptive Services (AS) PIC on an M Series or T Series router
- Multiservices (MS) PIC on an M Series or T Series router
- Multiservices (MS) DPC on an MX Series router
- EX Series switch

Before you begin, make sure that you have:

- Installed your supported router (or switch) and PICs or DPCs and performed the initial router (or switch) configuration.
- Configured basic Ethernet in the topology, and verified that traffic is flowing in the topology and that IPv4 traffic is flowing through logical interface **xe-0/1/0.0**.
- Configured the service set **vrf_svcs** with service input and output rules and default settings for services at a service interface.

For guidelines for configuring service sets, see *Configuring Service Sets to be Applied to Services Interfaces*.

Overview

In this example, you create three types of service filters for IPv4 traffic: one input service filter, one postservice input filter, and one output service filter.

Topology

You apply the input service filter and postservice input filter to input traffic at logical interface **xe-0/1/0.0**, and you apply the output service filter to the output traffic at the same logical interface.

- Filtering IPv4 traffic before it is accepted for input service processing—At logical interface **xe-0/1/0.0**, you use the service filter **in_filter_presvc** to filter IPv4 input traffic before the traffic can be accepted for processing by services associated with service set **vrf_svcs**. The **in_filter_presvc** service filter counts packets sent from ICMP port 179, directs these packets to the input services associated with the service set **vrf_svcs**, and discards all other packets.
- Filtering IPv4 traffic after it has completed input service processing—At logical interface **xe-0/1/0.0**, you use the service filter **in_filter_postsvc** to filter traffic that is returning to the services interface after the input service set **in_filter_presvc** is executed. The **in_filter_postsvc** service filter counts packets sent from ICMP port 179 and then discards them.
- Filtering IPv4 traffic before it is accepted for output service processing—At logical interface **xe-0/1/0.0**, you use the service-filter **out_filter_presvc** to filter IPv4 output traffic before the traffic can be accepted for processing by the services associated with service set **vrf_svcs**. The **out_filter_presvc** service filter counts packets destined for TCP port 179 and then directs the packets to the output services associated with the service set **vrf_svcs**.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

- Configuring the Three Service Filters on page 259
- Applying the Three Service Filters on page 260

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet service-filter in_filter_presvc term t1 from protocol tcp
set firewall family inet service-filter in_filter_presvc term t1 from source-port bgp
set firewall family inet service-filter in_filter_presvc term t1 then count svc_in_pkts
set firewall family inet service-filter in_filter_presvc term t1 then service
set firewall family inet service-filter in_filter_postsvc term t2 from protocol tcp
set firewall family inet service-filter in_filter_postsvc term t2 from source-port bgp
set firewall family inet service-filter in_filter_postsvc term t2 then count svc_in_pkts_rtn
```

```
set firewall family inet service-filter in_filter_postsvc term t2 then skip
set firewall family inet service-filter out_filter_presvc term t3 from protocol icmp
set firewall family inet service-filter out_filter_presvc term t3 from destination-port bgp
set firewall family inet service-filter out_filter_presvc term t3 then count svc_out_pkts
set firewall family inet service-filter out_filter_presvc term t3 then service
set interfaces xe-0/1/0 unit 0 family inet service input service-set vrf_svcs service-filter
  in_filter_presvc
set interfaces xe-0/1/0 unit 0 family inet service input post-service-filter in_filter_postsvc
set interfaces xe-0/1/0 unit 0 family inet service output service-set vrf_svcs service-filter
  out_filter_presvc
```

Configuring the Three Service Filters

Step-by-Step Procedure

To configure the three service filters:

1. Configure the input service filter.

```
[edit]
user@host# edit firewall family inet service-filter in_filter_presvc

[edit firewall family inet service-filter in_filter_presvc]
user@host# set term t1 from protocol tcp
user@host# set term t1 from source-port bgp
user@host# set term t1 then count svc_in_pkts
user@host# set term t1 then service
```

2. Configure the postservice input filter.

```
[edit]
user@host# edit firewall family inet service-filter in_filter_postsvc

[edit firewall family inet service-filter in_filter_postsvc]
user@host# set term t2 from protocol tcp
user@host# set term t2 from source-port bgp
user@host# set term t2 then count svc_in_pkts_rtn
user@host# set term t2 then skip
```

3. Configure the output service filter.

```
[edit]
user@host# edit firewall family inet service-filter out_filter_presvc

[edit firewall family inet service-filter out_filter_presvc]
user@host# set term t3 from protocol icmp
user@host# set term t3 from destination-port bgp
user@host# set term t3 then count svc_out_pkts
user@host# set term t3 then service
```

Results

Confirm the configuration of the input and output service filters and the postservice input filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this procedure to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
```

```
service-filter in_filter_presvc {
  term t1 {
    from {
      protocol tcp;
      source-port bgp;
    }
    then {
      count svc_in_pkts;
      service;
    }
  }
}
service-filter in_filter_postsvc {
  term t2 {
    from {
      protocol tcp;
      source-port bgp;
    }
    then {
      count svc_in_pkts_rtn;
      skip;
    }
  }
}
service-filter out_filter_presvc {
  term t3 {
    from {
      protocol icmp;
      destination-port bgp;
    }
    then {
      count svc_out_pkts;
      service;
    }
  }
}
}
```

Applying the Three Service Filters

Step-by-Step Procedure

To apply the three service filters:

1. Access the IPv4 protocol on the input interface **xe-0/1/0.0**.

   ```
   [edit]
   user@host# edit interfaces xe-0/1/0 unit 0 family inet
   ```

2. Apply the input service filter and the postservice input filter.

   ```
   [edit interfaces xe-0/1/0 unit 0 family inet]
   user@host# set service input service-set vrf_svcs service-filter in_filter_presvc
   user@host# set service input post-service-filter in_filter_postsvc
   user@host# set service output service-set vrf_svcs service-filter out_filter_presvc
   ```

Results Confirm the configuration of the interfaces by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
xe-0/1/0 {
  unit 0 {
    family inet {
      service {
        input {
          service-set vrf_svcs service-filter in_filter_presvc;
          post-service-filter in_filter_postsvc;
        }
        output {
          service-set vrf_svcs service-filter out_filter_presvc;
        }
      }
    }
  }
}
```

When you are done configuring the device, commit your candidate configuration.

Verification

Confirm that the configuration is working properly.

- Verifying That Inbound Traffic Is Filtered Before Input Service on page 261
- Verifying That Inbound Traffic Is Filtered After Input Service Processing on page 261
- Verifying That Outbound Traffic Is Filtered Before Output Service Processing on page 262

Verifying That Inbound Traffic Is Filtered Before Input Service

Purpose Verify that inbound packets sent from TCP port 179 are sent for processing by the *input* services associated with the service set **vrf_svcs**.

Action Display the count of packets sent for processing by the *input* services associated with the service set **vrf_svcs**.

```
[edit]
user@host> show firewall filter in_filter_presvc-vrf_svcs counter svc_in_pkts
```

Verifying That Inbound Traffic Is Filtered After Input Service Processing

Purpose Verify that inbound packets sent from TCP port 179 are returned from processing by the *input* services associated with the service set **vrf_svcs**.

Action Display the count of packets returned from processing by the *input* services associated with the service set **vrf_svcs**.

```
[edit]
user@host> show firewall filter in_filter_postsvc-vrf_svcs counter svc_in_pkts_rtn
```

Verifying That Outbound Traffic Is Filtered Before Output Service Processing

Purpose Verify that outbound packets sent to ICMP port 179 are sent for processing by the *output* services associated with the service set **vrf_svcs**.

Action Display the count of packets sent for processing by the *output* services associated with the service set **vrf_svcs**.

```
[edit]
user@host> show firewall filter out_filter_presvc-vrf_svcs counter svc_out_pkts
```

Related Documentation

- Service Filter Overview on page 79
- How Service Filters Evaluate Packets on page 80
- Guidelines for Configuring Service Filters on page 82
- Guidelines for Applying Service Filters on page 84

CHAPTER 21

Simple Filter Configuration

Example: Configuring and Applying a Simple Filter

This example shows how to configure a simple filter.

Requirements

This example uses one of the following hardware components:

- One Gigabit Ethernet intelligent queuing (IQ2) PIC installed on an M120, M320, or T Series router
- One Enhanced Queuing Dense Port Concentrator (EQ DPC) installed on an MX Series router or an EX Series switch

Before you begin, make sure that you have:

- Installed your supported router (or switch) and PIC or DPC and performed the initial router (or switch) configuration.
- Configured basic Ethernet in the topology, and verified that traffic is flowing in the topology and that ingress IPv4 traffic is flowing into logical interface **ge-0/0/1.0**.

Overview

This simple filter sets the loss priority to low for TCP traffic with source address **1.1.1.1**, sets the loss priority to high for HTTP (Web) traffic with source addresses in the 4.0.0.0/8 range, and sets the loss priority to low for all traffic with destination address **6.6.6.6**.

Topology

The simple filter is applied as an input filter (arriving packets are checking for destination address **6.6.6.6**, not queued output packets) on interface **ge-0/0/1.0**.

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see "Using the CLI Editor in Configuration Mode" on page 319.

To configure this example, perform the following tasks:

- Configuring the Simple Firewall Filter on page 264
- Applying the Simple Filter to the Logical Interface Input on page 266

CLI Quick Configuration

To quickly configure this example, copy the following commands into a text file, remove any line breaks, and then paste the commands into the CLI at the **[edit]** hierarchy level.

```
set firewall family inet simple-filter sf_classify_1 term 1 from source-address 1.1.1.1/32
set firewall family inet simple-filter sf_classify_1 term 1 from protocol tcp
set firewall family inet simple-filter sf_classify_1 term 1 then loss-priority low
set firewall family inet simple-filter sf_classify_1 term 2 from source-address 4.0.0.0/8
set firewall family inet simple-filter sf_classify_1 term 2 from protocol tcp
set firewall family inet simple-filter sf_classify_1 term 2 from source-port http
set firewall family inet simple-filter sf_classify_1 term 2 then loss-priority high
set firewall family inet simple-filter sf_classify_1 term 3 from destination-address
    6.6.6.6/32
set firewall family inet simple-filter sf_classify_1 term 3 then loss-priority low
set firewall family inet simple-filter sf_classify_1 term 3 then forwarding-class best-effort
set interfaces ge-0/0/1 unit 0 family inet simple-filter input sf_classify_1
set interfaces ge-0/0/1 unit 0 family inet address 10.1.2.3/30
```

Configuring the Simple Firewall Filter

Step-by-Step Procedure

To configure the simple filter:

1. Create the simple filter **sf_classify_1**.

   ```
   [edit]
   user@host# edit firewall family inet simple-filter sf_classify_1
   ```

2. Configure classification of TCP traffic based on the source IP address.

   ```
   [edit firewall family inet simple-filter sf_classify_1]
   user@host# set term 1 from source-address 1.1.1.1/32
   user@host# set term 1 from protocol tcp
   user@host# set term 1 then loss-priority low
   ```

3. Configure classification of HTTP traffic based on the source IP address.

   ```
   [edit firewall family inet simple-filter sf_classify_1]
   user@host# set term 2 from source-address 4.0.0.0/8
   user@host# set term 2 from protocol tcp
   user@host# set term 2 from source-port http
   user@host# set term 2 then loss-priority high
   ```

4. Configure classification of other traffic based on the destination IP address.

   ```
   [edit firewall family inet simple-filter sf_classify_1]
   user@host# set term 3 from destination-address 6.6.6.6/32
   ```

```
user@host# set term 3 then loss-priority low
user@host# set term 3 then forwarding-class best-effort
```

Results Confirm the configuration of the simple filter by entering the **show firewall** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show firewall
family inet {
  simple-filter sf_classify_1 {
    term 1 {
      from {
        source-address {
          1.1.1.1/32;
        }
        protocol {
          tcp;
        }
      }
      then loss-priority low;
    }
    term 2 {
      from {
        source-address {
          4.0.0.0/8;
        }
        source-port {
          http;
        }
        protocol {
          tcp;
        }
      }
      then loss-priority high;
    }
    term 3 {
      from {
        destination-address {
          6.6.6.6/32;
        }
      }
      then {
        loss-priority low;
        forwarding-class best-effort;
      }
    }
  }
}
```

Applying the Simple Filter to the Logical Interface Input

Step-by-Step Procedure

To apply the simple filter to the logical interface input:

1. Configure the logical interface to which you will apply the simple filter.

   ```
   [edit]
   user@host# edit interfaces ge-0/0/1 unit 0 family inet
   ```

2. Configure the interface address for the logical interface.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set address 10.1.2.3/30
   ```

3. Apply the simple filter to the logical interface input.

   ```
   [edit interfaces ge-0/0/1 unit 0 family inet]
   user@host# set simple-filter input sf_classify_1
   ```

Results

Confirm the configuration of the interface by entering the **show interfaces** configuration mode command. If the command output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/1 {
    unit 0 {
        family inet {
            simple-filter {
                input sf_classify_1;
            }
            address 10.1.2.3/30;
        }
    }
}
```

When you are done configuring the device, commit your candidate configuration.

Verification

Confirm that the configuration is working properly.

- Displaying the Mapping of Forwarding Class Maps and Names to Queue Numbers on page 266
- Displaying CoS Queue Counters for the Interface on page 267
- Displaying CoS Queue Counter Details for the Physical Interface on page 267

Displaying the Mapping of Forwarding Class Maps and Names to Queue Numbers

Purpose

Display the mapping of forwarding class names to queue numbers.

Action

Enter the **show class-of-service forwarding-class** operational mode command.

```
[edit]
user@host> show class-of-service forwarding-class
```

For information about the command output, see "**show class-of-service forwarding-class**" in the *Junos OS Operational Mode Commands*.

Displaying CoS Queue Counters for the Interface

Purpose Verify that the class-of-service (CoS) queue counters for the interface reflect the simple filter applied to the logical interface.

Action Enter the **show interfaces** command for the physical interface on which the simple filter is applied, and specify **detail** or **extensive** output level.

```
[edit]
user@host> show interfaces ge-0/0/1 detail
```

In the **Physical interface** section, under **Ingress queues**, the **Queue counters** section displays ingress queue counters for each forwarding class.

For more detailed information about the command output, see "**show interfaces (Gigabit Ethernet)**" or "**show interfaces (10-Gigabit Ethernet)**" in the *Junos OS Operational Mode Commands*.

Displaying CoS Queue Counter Details for the Physical Interface

Purpose Verify that the CoS queue counter details for the physical interface reflect the simple filter applied to the logical interface.

Action Enter the **show interfaces queue** command for the physical interface on which the simple filter is applied, and specify the **ingress** option.

```
[edit]
user@host> show interfaces queue ge-0/0/1 ingress
```

For information about the command output, see "**show interfaces queue**" in the *Junos OS Operational Mode Commands*.

Related Documentation

- Simple Filter Overview on page 87
- How Simple Filters Evaluate Packets on page 87
- Guidelines for Configuring Simple Filters on page 89
- Guidelines for Applying Simple Filters on page 92

CHAPTER 22

Firewall Filters Statement Hierarchies

Statement Hierarchy for Configuring Interface-Specific Firewall Filters

To enable interface-specific instances for stateless firewall filters, include the **interface-specific** statement in the **filter** *filter-name* or **service-filter** *service-filter-name* stanza. Any counters specified as actions in an interface-specific filter are maintained separately per filter instance. Any policers specified as actions in an interface-specific filter are applied per filter instance.

```
firewall {
   family family-name {
      (filter filter-name | service-filter service-filter-name) {
         ...
         interface-specific;
         ...
         term term-name {
            from {
               match-conditions;
            }
```

```
            then {
                count counter-name;
                policer policer-name;
            }
        }
        ...
    }
  }
]
```

You can include the firewall configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

Related Documentation

- Interface-Specific Firewall Filter Instances Overview on page 61
- Statement Hierarchy for Applying Interface-Specific Firewall Filters on page 270
- Example: Configuring Interface-Specific Firewall Filter Counters on page 207

Statement Hierarchy for Applying Interface-Specific Firewall Filters

To apply an interface-specific stateless firewall filter to a logical interface, include the **input *filter-name*** or **output *filter-name*** statement in the **filter** or **service-filter** stanza of the interfaces configuration:

```
interfaces {
  interface-name {
    unit unit-number {
      family family-name {
        filter {
          input filter-name-1;
          output filter-name-2;
        }
        service-filter {
          input service-filter-name-1;
          output service-filter-name-2;
        }
      }
    }
  }
}
```

You can include the interface configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

Related Documentation

- Interface-Specific Firewall Filter Instances Overview on page 61
- Statement Hierarchy for Configuring Interface-Specific Firewall Filters on page 269
- Example: Configuring Interface-Specific Firewall Filter Counters on page 207

Statement Hierarchy for Assigning Interfaces to Interface Groups

To assign a logical interface to an interface group, specify the group number by including the **group** *interface-group-number* statement in the **filter** stanza:

```
interfaces {
  interface-name {
    unit unit-number {
      family ( inet | inet6 | vpls | ccc | bridge) {
        filter {
          group interface-group-number;
        }
      }
    }
  }
}
```

NOTE: The number 0 is not a valid number for an interface group.

You can configure the firewall filter at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems** *logical-system-name***]**

Related Documentation

- Filtering Packets Received on a Set of Interface Groups Overview on page 63
- Statement Hierarchy for Configuring a Filter to Match on a Set of Interface Groups on page 271
- Example: Filtering Packets Received on an Interface Group on page 211

Statement Hierarchy for Configuring a Filter to Match on a Set of Interface Groups

You can configure a standard stateless firewall filter or a service filter term that matches packets tagged for a specified interface group or set of interface groups.

To configure a standard stateless firewall filter that matches packets tagged for a specified interface group or set of interface groups, configure a filter term that uses the **interface-group** *interface-group-number* match condition:

```
firewall {
  family (inet | inet6 | vpls | ccc | bridge) {
    filter filter-name {
      term term-name {
        from {
          interface-group interface-group-number;
        }
        then {
          filter-actions;
        }
      }
```

```
        }
    }
}
```

To configure a service filter that matches packets tagged for a specified interface group or set of interface groups, configure a filter term that uses the **interface-group** ***interface-group-name*** match condition:

```
firewall {
    family (inet | inet6) {
        service-filter filter-name {
            term term-name {
                from {
                    interface-group interface-group-number;
                }
                then {
                    service-filter-actions;
                }
            }
        }
    }
}
```

You can configure the firewall filter at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems** ***logical-system-name*****]**

Related Documentation

- Filtering Packets Received on a Set of Interface Groups Overview on page 63
- Statement Hierarchy for Assigning Interfaces to Interface Groups on page 271
- Example: Filtering Packets Received on an Interface Group on page 211

Statement Hierarchy for Applying Filters to an Interface Group

To apply a standard stateless firewall filter to an interface group, include the **input** ***filter-name*** or **output** ***filter-name*** in the **filter** stanza:

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

You can include the interface configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

Related Documentation

- Interface-Specific Firewall Filter Instances Overview on page 61
- Statement Hierarchy for Assigning Interfaces to Interface Groups on page 271
- Statement Hierarchy for Configuring a Filter to Match on a Set of Interface Groups on page 271
- Example: Filtering Packets Received on an Interface Group on page 211

Statement Hierarchy for Defining an Interface Set

To configure a named group of interfaces that can be referenced in a stateless firewall filter match condition, use the **interface-set** statement to define the interface-set name and two or more interfaces:

```
firewall {
  interface-set interface-set-name {
    interface-name;
  }
}
```

You can include the statements at one of the following hierarchy levels:

- **[edit firewall]**
- **[edit logical-systems *logical-system-name* firewall]**

To specify that the interface set contains all interfaces of a particular type, you can use the '*' (asterisk) wildcard character. For example, use **fe-*** to specify all Fast Ethernet interfaces.

Related Documentation

- Filtering Packets Received on an Interface Set Overview on page 63
- Statement Hierarchy for Configuring a Filter to Match on an Interface Set on page 273
- Example: Configuring a Rate-Limiting Filter Based on Destination Class on page 252
- Example: Filtering Packets Received on an Interface Set on page 116

Statement Hierarchy for Configuring a Filter to Match on an Interface Set

To configure a standard stateless firewall filter that matches packets tagged for a specified interface group or set of interface groups, configure a filter term that uses the **interface-group *interface-group-name*** match condition:

```
firewall {
  family (any | inet | inet6 | mpls | vpls | bridge) {
    filter filter-name {
      term term-name {
```

```
            from {
              interface-set interface-set-name;
            }
            then {
              filter-actions;
            }
          }
        }
      }
    }
```

Related Documentation

- Filtering Packets Received on an Interface Set Overview on page 63
- Statement Hierarchy for Defining an Interface Set on page 273
- Example: Configuring a Rate-Limiting Filter Based on Destination Class on page 252
- Example: Filtering Packets Received on an Interface Set on page 116

Statement Hierarchy for Configuring FBF for IPv4 or IPv6 Traffic

You can configure stateless firewall filters for filter-based forwarding by configuring filter terms that specify the **forwarding-class** *class-name* nonterminating action or the **routing-instance** *routing-instance-name* terminating action:

```
firewall {
  family (inet | inet6) {
    filter filter-name {
      term term-name {
        from {
          ipv4-or-ipv6-match-conditions;
        }
        then {
          forwarding-class class-name; #optional
          other-optional-nonterminating-actions;
          routing-instance routing-instance-name <topology topology-name>;
        }
      }
    }
  }
}
```

You can include the firewall configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems** *logical-system-name***]**

Related Documentation

- Filter-Based Forwarding Overview on page 75
- Firewall Filter Match Conditions for IPv4 Traffic on page 341
- Standard Firewall Filter Match Conditions for IPv6 Traffic on page 350
- Statement Hierarchy for Configuring Routing Instances for FBF on page 277
- Example: Configuring Filter-Based Forwarding on the Source Address on page 233

Statement Hierarchy for Configuring FBF for IPv4 Traffic on ACX Series Routers

On ACX Series routers, you can configure stateless firewall filters for filter-based forwarding (FBF) by configuring filter terms that specify the optional nonterminating actions or the **routing-instance** ***routing-instance-name*** terminating action:

```
firewall {
  family inet {
    filter filter-name {
      term term-name {
        from {
          ipv4-match-conditions;
        }
        then {
          optional-nonterminating-actions;
          routing-instance routing-instance-name ;
        }
      }
    }
  }
}
```

You can include the firewall configuration at the **[edit]** hierarchy level:

The **[edit logical-systems** ***logical-system-name*****]** hierarchy level is not supported on the ACX Series routers.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Standard Firewall Filter Match Conditions and Actions on ACX Series Routers Overview on page 389
- Standard Firewall Filter Terminating Actions on ACX Series Routers on page 397
- Standard Firewall Filter Nonterminating Actions on ACX Series Routers on page 394
- Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers on page 391
- Standard Firewall Filter Match Conditions for MPLS Traffic on ACX Series Routers on page 394

Statement Hierarchy for Configuring FBF for MPLS-Tagged IPv4 Traffic

Matching on IPv4 Address and TCP/UDP Port Fields

To configure a firewall filter term that matches on IP source and destination address fields, and TCP and UDP ports in the IPv4 header of packets in an MPLS flow, you can specify supported match conditions as shown here:

```
[edit]
```

```
interfaces {
   interface-name {
      unit logical-unit-number {
         family family {
            address ip-address;
         }
         family mpls {
            filter {
               input filter-name;
            }
         }
      }
   }
}
firewall {
   family mpls {
      filter filter-name {
         term term-name {
            from {
               ip-version ipv4 {
                  destination-address {
                     ip-address;
                  }
                  source-address {
                     ip-address;
                  }
                  protocol tcp {
                     destination-port tcp-port;
                     destination-port-except tcp-port;
                     source-port tcp-port;
                     source-port-except tcp-port;
                  }
                  protocol udp {
                     destination-port udp-port;
                     destination-port-except udp-port;
                     source-port udp-port;
                     source-port-except udp-port;
                  }
               }
            }
            then {
               ...
            }
         }
      }
   }
}
```

Configuration Example

```
interfaces {
   ge-6/0/0 {
      unit 0 {
         family inet {
            address 20.20.20.1/30;
         }
```

```
            family mpls {
              filter {
                input mpls-ipv4-filter1;
              }
            }
          }
        }
      }
      firewall {
        family mpls {
          filter mpls-ipv4-filter1 {
            term term1 {
              from {
                ip-version ipv4 {
                  source-address {
                    10.0.0.1/32;
                  }
                  protocol tcp {
                    destination-port ftp;
                  }
                }
              }
              then {
                count counter1;
                discard;
              }
            }
          }
        }
      }
```

Related Documentation

- Filter-Based Forwarding Overview on page 75
- Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic on page 360
- Statement Hierarchy for Configuring Routing Instances for FBF on page 277
- Example: Configuring Filter-Based Forwarding on the Source Address on page 233

Statement Hierarchy for Configuring Routing Instances for FBF

A routing instance is a collection of routing tables, interfaces, and routing protocol parameters. The set of interfaces belongs to the routing tables, and the routing protocol parameters control the information in the routing tables.

To configure a routing instance for filter-based forwarding:

1. The **instance-type** must be **forwarding**. The **forwarding** routing instance type supports filter-based forwarding, where interfaces are not associated with instances. For this instance type, there is no one-to-one mapping between an interface and a routing instance. All interfaces belong to the default instance **inet.0**.
2. The name of the routing instance name must be the one referenced in the firewall filter action.

NOTE: In Junos OS Release 9.0 and later, you can no longer specify a routing-instance name of **default** or include special characters within the name of a routing instance.

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 *Routing Databases Overview*.

```
routing-instances {
  routing-table-name {
    instance-type forwarding;
    routing-options {
      static {
        route destination-prefix nexthop address;
      }
    }
  }
}
```

You can include the **forwarding** routing instance at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

Related Documentation

- Filter-Based Forwarding Overview on page 75
- Statement Hierarchy for Configuring FBF for IPv4 or IPv6 Traffic on page 274
- Statement Hierarchy for Configuring FBF for MPLS-Tagged IPv4 Traffic on page 275
- Example: Configuring Filter-Based Forwarding on the Source Address on page 233

Statement Hierarchy for Applying FBF Filters to Interfaces

To apply filter-based forwarding to a logical interface, include the **input** or **output** statement in the **filter** stanza.

NOTE: An interface configured with filter-based forwarding does not support source-class usage (SCU) filter matching and unicast reverse-path forwarding (RPF) check filters.

```
interfaces {
```

```
interface-name {
  unit unit-number {
    family (inet | inet6 | mpls) {
      filter {
        input filter-name;
        output filter-name;
      }
      address address;
    }
  }
}
```

You can include the interfaces configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

Related Documentation

- Filter-Based Forwarding Overview on page 75
- Statement Hierarchy for Configuring FBF for IPv4 or IPv6 Traffic on page 274
- Statement Hierarchy for Configuring FBF for MPLS-Tagged IPv4 Traffic on page 275
- Statement Hierarchy for Configuring Routing Instances for FBF on page 277
- Example: Configuring Filter-Based Forwarding on the Source Address on page 233

Statement Hierarchy for Configuring Firewall Filter Accounting Profiles

To configure an accounting profile that you can apply to a firewall filter, include the **filter-profile *filter-profile-name*** statement in the **accounting-options** stanza.

```
accounting-options {
  filter-profile filter-profile-name {
    file log-filename {
      archive-sites {
        site-urls;
      }
      files number;
      size bytes;
      start-time time;
      transfer-interval minutes;
    }
    interval minutes;
    counters {
      counter-name-1;
      counter-name-2;
    }
  }
}
```

You can include the accounting options configuration at one of the following hierarchy levels:

- **[edit]**
- **[edit logical-systems *logical-system-name*]**

To specify the name of the accounting data log file in the **/var/log** directory to be used in conjunction with the accounting profile, include the **file *log-filename*** statement.

To specify how often statistics are collected for the accounting profile, include the **interval *minutes*** statement.

To specify the names of the firewall filter counters for which filter profile statistics are collected, include the **counters** statement.

Related Documentation

- Accounting for Firewall Filters Overview on page 45
- Statement Hierarchy for Applying Firewall Filter Accounting Profiles on page 280
- Example: Configuring Statistics Collection for a Firewall Filter on page 187

Statement Hierarchy for Applying Firewall Filter Accounting Profiles

You can apply an accounting profile to a standard stateless firewall filter for any supported protocol family except **family any**.

To apply a filter-accounting profile to a stateless firewall filter, include the **accounting-profile *accounting-profile-name*** statement at the firewall **filter** stanza:

```
firewall {
    family family-name {
        filter filter-name {
            accounting-profile accounting-profile-name;
            interface-specific;
            physical-interface-policer;
            term {
                filter filter-profile-name;
            }
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    actions;
                }
            }
        }
    }
}
```

You can include the firewall configuration at one of the following hierarchy levels:

- **[edit]**

- [edit logical-systems *logical-system-name*]

Related Documentation

- Accounting for Firewall Filters Overview on page 45
- Statement Hierarchy for Configuring Firewall Filter Accounting Profiles on page 279
- Example: Configuring Statistics Collection for a Firewall Filter on page 187

CHAPTER 23

Firewall Filter Configuration Statements

[edit firewall] Hierarchy Level

Several statements in the **[edit firewall]** hierarchy are valid at numerous locations within the hierarchy. To make the complete hierarchy easier to read, the repeated statements are listed in the following sections, which are referenced at the appropriate locations in "Complete [edit firewall] Hierarchy" on page 288.

Common Firewall Actions

This section lists statements that are valid at the following hierarchy levels, and is referenced at those levels in "Complete [edit firewall] Hierarchy" on page 288 instead of the statements being repeated.

- **[edit firewall family (any | bridge | ccc | inet | inet6 | mpls | vpls) filter *filter-name* term *term-name* then]**
- **[edit firewall filter *filter-name* term *term-name* then]**

The common firewall actions are as follows:

```
count counter-name;
forwarding-class class-name;
loss-priority (high | low | medium-high | medium-low);
next term;
policer policer-name;
three-color-policer policer-name {
   (single-rate single-rate-policer-name | two-rate two-rate-policer-name);
}
```

Common IP Firewall Actions

This section lists statements that are valid at the following hierarchy levels, and is referenced at those levels in "Complete [edit firewall] Hierarchy" on page 288 instead of the statements being repeated.

- **[edit firewall family inet filter *filter-name* term *term-name* then]**
- **[edit firewall family inet6 filter *filter-name* term *term-name* then]**
- **[edit firewall filter *filter-name* term *term-name* then]**

The common IP firewall actions are as follows:

```
log;
logical-system logical-system-name <routing-instance routing-instance-name>
   <topology topology-name>;
port-mirror;
port-mirror-instance instance-name;
routing-instance routing-instance-name> <topology topology-name>;
sample;
service-filter-hit;
syslog;
topology topology-name;
```

Common IPv4 Firewall Actions

This section lists statements that are valid at the following hierarchy levels, and is referenced at those levels in "Complete [edit firewall] Hierarchy" on page 288 instead of the statements being repeated.

- **[edit firewall family inet filter *filter-name* term *term-name* then]**
- **[edit firewall filter *filter-name* term *term-name* then]**

The common IP version 4 (IPv4) firewall actions are as follows:

```
(accept | discard <accounting collector-name> | reject <administratively-prohibited |
   bad-host-tos | bad-network-tos | fragmentation-needed | 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 | tcp-reset>);
ipsec-sa sa-name;
load-balance sa-name;
next-hop-group group-name;
prefix-action action-name;
```

Common IP Firewall Match Conditions

This section lists statements that are valid at the following hierarchy levels, and is referenced at those levels in "Complete [edit firewall] Hierarchy" on page 288 instead of the statements being repeated.

- **[edit firewall family inet dialer-filter *filter-name* term *term-name* from]** (with the exceptions noted at this level in "Complete [edit firewall] Hierarchy" on page 288)
- **[edit firewall family inet filter *filter-name* term *term-name* from]**
- **[edit firewall family inet6 dialer-filter *filter-name* term *term-name* from]** (with the exceptions noted at this level in "Complete [edit firewall] Hierarchy" on page 288)
- **[edit firewall family inet6 filter *filter-name* term *term-name* from]**
- **[edit firewall filter *filter-name* term *term-name* from]**

The common IP firewall match conditions are as follows:

```
address {
    ip-prefix</prefix-length> <except>;
}
destination-address {
    ip-prefix</prefix-length> <except>;
}
destination-class [ class-names ] | destination-class-except [ class-names ]);
(destination-port [ port-names ] | destination-port-except [ port-names ]);
destination-prefix-list {
    list-name <except>;
}
(forwarding-class [ class-names ] | forwarding-class-except [ class-names ]);
(icmp-code [ codes ] | icmp-code-except [ codes ]);
(icmp-type [ types ] | icmp-type-except [ types ]);
interface interface-name;
(interface-group [ group-names ] | interface-group-except [ group-names ]);
interface-set set-name;
(loss-priority [ priorities ] | loss-priority-except [ priorities ]);
(packet-length [ values ] | packet-length-except [ values ]);
(port [ port-names ] | port-except [ port-names ]);
prefix-list {
    list-name <except>;
}
service-filter-hit;
source-address {
    ip-prefix</prefix-length> <except>;
}
(source-class [ class-names ] | source-class-except [ class-names ]);
(source-port [ port-names ] | source-port-except [ port-names ]);
source-prefix-list {
    list-name <except>;
}
tcp-established;
tcp-flags flag;
tcp-initial;
```

Common IPv4 Firewall Match Conditions

This section lists statements that are valid at the following hierarchy levels, and is referenced at those levels in "Complete [edit firewall] Hierarchy" on page 288 instead of the statements being repeated.

- **[edit firewall family inet dialer-filter *filter-name* term *term-name* from]** (with the exceptions noted at this level in "Complete [edit firewall] Hierarchy" on page 288)
- **[edit firewall family inet filter *filter-name* term *term-name* from]**
- **[edit firewall filter *filter-name* term *term-name* from]**

The common IPv4 firewall match conditions are as follows:

```
(ah-spi [ values ] | ah-spi-except [ values ]);
(dscp [ code-point-values ] | dscp-except [ code-point-values ]);
(esp-spi [ values ] | esp-spi-except [ values ]);
first-fragment;
fragment-flags flag;
(fragment-offset [ offsets ] | fragment-offset-except [ offsets ]);
(ip-options [ option-names ] | ip-options-except [ option-names ]);
is-fragment;
(precedence [ precedence-names ] | precedence-except [ precedence-names ]);
(protocol [ protocol-names ] | protocol-except [ protocol-names ]);
(ttl [ ttl-values ] | ttl-except [ ttl-values ]);
```

NOTE: On M and T series routers, firewall filters cannot count **ip-options** packets on a per option type and per interface basis. A limited work around is to use the **show pfe statistics ip options** command to see **ip-options** statistics on a per Packet Forwarding Engine (PFE) basis. See *show pfe statistics ip* for sample output.

Common Layer 2 Firewall Match Conditions

This section lists statements that are valid at the following hierarchy levels, and is referenced at those levels in "Complete [edit firewall] Hierarchy" on page 288 instead of the statements being repeated.

- **[edit firewall family bridge filter *filter-name* term *term-name* from]**
- **[edit firewall family vpls filter *filter-name* term *term-name* from]**

The common Layer 2 firewall match conditions are as follows:

```
destination-mac-address {
  mac-address <except>;
}
(destination-port [ port-names ] | destination-port-except [ port-names ]);
(dscp [ code-point-values ] | dscp-except [ code-point-values ]);
(ether-type [ protocol-types ] | ether-type-except [ protocol-types ]);
(forwarding-class [ class-names ] | forwarding-class-except [ class-names ]);
(icmp-code [ codes ] | icmp-code-except [ codes ]);
```

```
(icmp-type [ types ] | icmp-type-except [ types ]);
(interface-group [ group-names ] | interface-group-except [ group-names ]);
ip-address {
   ip-prefix</prefix-length> <except>;
}
ip-destination-address {
   ip-prefix</prefix-length> <except>;
}
(ip-precedence [ precedence-names ] | ip-precedence-except [ precedence-names ]);
(ip-protocol [ protocol-names ] | ip-protocol-except [ protocol-names ]);
ip-source-address ip-prefix</prefix-length>;
(learn-vlan-1p-priority [ priorities ] | learn-vlan-1p-priority [ priorities ]);
(learn-vlan-id [ vlan-ids ] | learn-vlan-id-except [ vlan-ids ]);
(loss-priority [ priorities ] | loss-priority-except [ priorities ]);
(port [ port-names ] | port-except [ port-names ]);
source-mac-address {
   mac-address <except>;
}
(source-port [ port-names ] | source-port-except [ port-names ]);
tcp-flags flag;
(traffic-type [ broadcast known-unicast multicast unknown-unicast ] |
   traffic-type-except [ broadcast known-unicast multicast unknown-unicast ]);
(user-vlan-1p-priority [ priorities ] | user-vlan-1p-priority [ priorities ]);
(user-vlan-id [ vlan-ids ] | user-vlan-id-except [ vlan-ids ]);
(vlan-ether-type [ protocol-types ] | vlan-ether-type-except [ protocol-types ]);
```

Complete [edit firewall] Hierarchy

```
firewall {
  family (any | bridge | ccc | inet | inet6 | mpls | vpls) {
    ... the family subhierarchies appear after the main [edit firewall] hierarchy ...
  }
  filter filter-name {
    accounting-profile [ profile-names ];
    enhanced-mode;
    interface-specific;
    physical-interface-policer;
    term term-name {
      filter filter-name;
      from {
        ... statements in Common IP Firewall Match Conditions on page 285 AND
          statements in Common IPv4 Firewall Match Conditions on page 286 ...
      }
      then {
        ... statements in Common Firewall Actions on page 283 AND
          statements in Common IP Firewall Actions on page 284 AND
          statements in Common IPv4 Firewall Actions on page 284 ...
      }
    }
  }
  hierarchical-policer policer-name {
    aggregate {
      if-exceeding {
        bandwidth-limit bps;
        burst-size-limit bytes;
      }
      then {
        discard;
        forwarding-class class-name;
        loss-priority (high | low | medium-high | medium-low);
      }
    }
    premium {
      if-exceeding {
        bandwidth-limit bps;
        burst-size-limit bytes;
      }
      then {
        discard;
      }
    }
  }
  interface-set interface-set-name {
    interface-name;
  }
  load-balance-group group-name {
    next-hop-group [ group-names ];
  }
  policer policer-name {
    filter-specific;
    if-exceeding {
```

```
        (bandwidth-limit bps | bandwidth-percent percentage);
        burst-size-limit bytes;
      }
      logical-bandwidth-policer;
      logical-interface-policer;
      physical-interface-policer;
      then {
        discard;
        forwarding-class class-name;
        loss-priority (high | low | medium-high | medium-low);
      }
    }
    three-color-policer policer-name {
      action {
        loss-priority high then discard;
      }
      logical-interface-policer;
      single-rate {
        (color-aware | color-blind);
        committed-burst-size bytes;
        committed-information-rate bps;
        excess-burst-size bytes;
      }
      two-rate {
        (color-aware | color-blind);
        committed-burst-size bytes;
        committed-information-rate bps;
        peak-burst-size bytes;
        peak-information-rate bps;
      }
    }
}

firewall {
    family any {
        filter filter-name {
            term term-name {
                from {
                    (forwarding-class [ class-names ] | forwarding-class-except [ class-names ]);
                    interface interface-name;
                    interface-set set-name;
                    (loss-priority [ priorities ] | loss-priority-except [ priorities ]);
                    (packet-length [ values ] | packet-length-except [ values ]);
                }
                then {
                    ... statements in Common Firewall Actions on page 283 PLUS ...
                    (accept | discard);
                }
            }
        }
    }
}

firewall {
    family bridge {
        filter filter-name {
```

```
        accounting-profile [ profile-names ];
        interface-specific;
        term term-name {
          filter filter-name;
          from {
            ... statements in Common Layer 2 Firewall Match Conditions on page 286 ...
          }
          then {
            ... statements in Common Firewall Actions on page 283 PLUS ...
            (accept | discard);
            port-mirror;
            port-mirror-instance instance-name;
          }
        }
      }
    }
}

firewall {
    family ccc {
      filter filter-name {
        accounting-profile [ profile-names ];
        interface-specific;
        term term-name {
          filter filter-name;
          from {
            (forwarding-class [ class-names ] | forwarding-class-except [ class-names ]);
            (interface-group [ group-names ] | interface-group-except [ group-names ]);
            (learn-vlan-1p-priority [ priorities ] | learn-vlan-1p-priority [ priorities ]);
            (loss-priority [ priorities ] | loss-priority-except [ priorities ]);
            (user-vlan-1p-priority [ priorities ] | user-vlan-1p-priority [ priorities ]);
          }
          then {
            ... statements in Common Firewall Actions on page 283 PLUS ...
            (accept | discard);
            port-mirror-instance instance-name;
          }
        }
      }
    }
}

firewall {
    family ethernet-switching {
      filter filter-name {
        interface-specific;
        term term-name {
          from {
            destination-address {
              ip-prefix</prefix-length>;
            }
            destination-mac-address {
              mac-address;
            }
            destination-port [ port-names ];
            destination-prefix-list {
```

```
            list-name;
          }
          dot1q-tag [ tag-values ];
          dot1q-user-priority [ priority-values ];
          dscp [ code-point-values ];
          ether-type [ protocol-names ];
          fragment-flags flag;
          icmp-code [ codes ];
          icmp-type [ types ];
          interface interface-name;
          is-fragment;
          precedence [ precedence-names ];
          protocol [ protocol-names ];
          source-address {
            ip-prefix</prefix-length>;
          }
          source-mac-address {
            mac-address;
          }
          source-port [ port-names ];
          source-prefix-list {
            list-name;
          }
          tcp-established;
          tcp-flags flag;
          tcp-initial;
          vlan [ vlan-names ];
        }
        then {
          (accept | discard);
          analyzer analyzer-name;
          count counter-name;
          forwarding-class class-name;
          interface interface-name;
          log;
          loss-priority (high | low);
          policer policer-name;
          syslog;
          vlan vlan-name;
        }
      }
    }
  }
}

firewall {
  family inet {
    dialer-filter filter-name {
      accounting-profile [ profile-names ];
      term term-name {
        from {
          ... statements in Common IP Firewall Match Conditions on page 285 AND
            statements in Common IPv4 Firewall Match Conditions on page 286 EXCEPT
            FOR ...
          (ah-spi [ values ] | ah-spi-except [ values ]);   # NOT valid at this level
```

```
        (destination-class [ class-names ] | destination-class-except [ class-names ]);  #
          NOT valid at this level
        interface interface-name;   # NOT valid at this level
        (loss-priority [ priorities ] | loss-priority-except [ priorities ]);   # NOT valid at this
          level
        service-filter-hit;   # NOT valid at this level
        (source-class [ class-names ] | source-class-except [ class-names ]);   # NOT
          valid at this level
      }
      then {
        (ignore | note);
        log;
        sample;
        syslog;
      }
    }
  }
  filter filter-name {
    accounting-profile [ profile-names ];
    enhanced-mode;
    interface-shared;
    interface-specific;
    physical-interface-filter;
    term term-name {
      filter filter-name;
      from {
        ... statements in Common IP Firewall Match Conditions on page 285 AND
          statements in Common IPv4 Firewall Match Conditions on page 286 ...
      }
      then {
        ... statements in Common Firewall Actions on page 283 AND
          statements in Common IP Firewall Actions on page 284 AND
          statements in Common IPv4 Firewall Actions on page 284 ...
      }
    }
  }
  prefix-action name {
    count;
    destination-prefix-length prefix-length;
    filter-specific;
    policer policer-name;
    source-prefix-length prefix-length;
    subnet-prefix-length prefix-length;
  }
  service-filter filter-name {
    term term-name {
      from {
        address {
          ip-prefix</prefix-length>;
        }
        (ah-spi [ values ] | ah-spi-except [ values ]);
        destination-address {
          ip-prefix</prefix-length>;
        }
        (destination-port [ port-names ] | destination-port-except [ port-names ]);
        destination-prefix-list {
```

```
            list-name;
          }
          (esp-spi [ values ] | esp-spi-except [ values ]);
          first-fragment;
          fragment-flags flag;
          (fragment-offset [ offsets ] | fragment-offset-except [ offsets ]);
          (interface-group [ group-names ] | interface-group-except [ group-names ]);
          (ip-options [ option-names ] | ip-options-except [ option-names ]);
          is-fragment;
          (loss-priority [ priorities ] | loss-priority-except [ priorities ]);
          (port [ port-names ] | port-except [ port-names ]);
          prefix-list {
            list-name;
          }
          (protocol [ protocol-names ] | protocol-except [ protocol-names ]);
          source-address {
            ip-prefix</prefix-length>;
          }
          (source-port [ port-names ] | source-port-except [ port-names ]);
          source-prefix-list {
            list-name;
          }
          tcp-flags flag-name;
        }
        then {
          count counter-name;
          log;
          port-mirror;
          sample;
          (service | skip);
        }
      }
    }
    simple-filter filter-name {
      term term-name {
        from {
          destination-address ip-prefix</prefix-length>;
          destination-port port-name;
          forwarding-class [ class-names ];
          protocol protocol-name;
          source-address ip-prefix</prefix-length>;
          source-port port-name;
        }
        then {
          forwarding-class class-name;
          loss-priority (high | low | medium-high | medium-low);
          policer policer-name;
        }
      }
    }
  }
}

firewall {
  family inet6 {
    dialer-filter filter-name {
```

```
    accounting-profile [ profile-names ];
    term term-name {
      from {
        ... statements in Common IP Firewall Match Conditions on page 285 PLUS ...
        (next-header [ protocol-types ] | next-header-except [ protocol-types ]);
        ... BUT NOT ...
          (destination-class [ class-names ] |
            destination-class-except [ class-names ]);   # NOT valid at this level
          (forwarding-class [ class-names ] |
            forwarding-class-except [ class-names ]);   # NOT valid at this level
          interface interface-name;   # NOT valid at this level
          (interface-group [ group-names ] | interface-group-except [ group-names ]);  #
            NOT valid at this level
          (loss-priority [ priorities ] | loss-priority-except [ priorities ]);   # NOT valid at
            this level
          service-filter-hit;   # NOT valid at this level
          (source-class [ class-names ] | source-class-except [ class-names ]);   # NOT
            valid at this level
          tcp-established;   # NOT valid at this level
          tcp-flags flag;   # NOT valid at this level
        tcp-initial;   # NOT valid at this level
      }
      then {
        (ignore | note);
        log;
        sample;
        syslog;
      }
    }
}
filter filter-name {
    accounting-profile [ profile-names ];
    interface-specific;
    term term-name {
      filter filter-name;
      from {
        ... statements in Common IP Firewall Match Conditions on page 285 PLUS ...
        (next-header [ protocol-types ] | next-header-except [ protocol-types ]);
        (traffic-class [ code-point-values ] | traffic-class-except [ code-point-values ]);
      }
      then {
        ... statements in Common Firewall Actions on page 283 AND
          statements in Common IP Firewall Actions on page 284 PLUS ...
        (accept | discard | reject <address-unreachable | administratively-prohibited |
          beyond-scope | fragmentation-needed | no-route | port-unreachable |
          tcp-reset>);
      }
    }
}
service-filter filter-name {
    term term-name {
      from {
        address {
          ip-prefix</prefix-length>;
        }
        (ah-spi [ values ] | ah-spi-except [ values ]);
```

```
            destination-address {
              ip-prefix</prefix-length>;
            }
            (destination-port [ port-names ] | destination-port-except [ port-names ]);
            destination-prefix-list {
              list-name;
            }
            (esp-spi [ values ] | esp-spi-except [ values ]);
            (interface-group [ group-names ] | interface-group-except [ group-names ]);
            (next-header [ protocol-types ] | next-header-except [ protocol-types ]);
            (port [ port-names ] | port-except [ port-names ]);
            prefix-list {
              list-name;
            }
            source-address {
              ip-prefix</prefix-length>;
            }
            (source-port [ port-names ] | source-port-except [ port-names ]);
            source-prefix-list {
              list-name;
            }
            tcp-flags flag-name;
          }
          then {
            count counter-name;
            log;
            port-mirror;
            sample;
            (service | skip);
          }
        }
      }
    }
}

firewall {
  family mpls {
    dialer-filter filter-name {
      accounting-profile [ profile-names ];
      term term-name {
        from {
          (exp [ exp-bits ] | exp-except [ exp-bits ]);
        }
        then {
          (ignore | note);
          log;
          sample;
          syslog;
        }
      }
    }
    filter filter-name {
      accounting-profile [ profile-names ];
      interface-specific;
      term term-name {
        filter filter-name;
```

```
            from {
              (exp [ exp-bits ] | exp-except [ exp-bits ]);
              (forwarding-class [ class-names ] | forwarding-class-except [ class-names ]);
              interface interface-name;
              interface-set set-name;
              (loss-priority [ priorities ] | loss-priority-except [ priorities ]);
            }
            then {
              ... statements in Common Firewall Actions on page 283 PLUS ...
              (accept | discard);
              sample;
            }
          }
        }
      }
    }

    firewall {
      family vpls {
        filter filter-name {
          accounting-profile [ profile-names ];
          interface-specific;
          term term-name {
            filter filter-name;
            from {
              ... statements in Common Layer 2 Firewall Match Conditions on page 286 ...
            }
            then {
              ... statements in Common Firewall Actions on page 283 PLUS ...
              (accept | discard);
              port-mirror;
              port-mirror-instance instance-name;
            }
          }
        }
      }
    }
```

Related Documentation

- *Notational Conventions Used in Junos OS Configuration Hierarchies*

accounting-profile

Syntax accounting-profile *name*;

Hierarchy Level [edit firewall family *family-name* filter *filter-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Enable collection of accounting data for the specified filter.

Options ***name***—Name assigned to the accounting profile.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Accounting for Firewall Filters Overview on page 45

enhanced-mode

Syntax enhanced-mode;

Hierarchy Level [edit firewall filter *filter-name*],
[edit firewall family *family-name* filter *filter-name*],
[edit logical-systems *logical-system-name* firewall filter *filter-name*],
[edit logical-systems *logical-system-name* firewall family *family-name* filter *filter-name*]

Release Information Statement introduced in Junos OS Release 11.4.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Limit static service filters or API-client filters to term-based filter format only for inet or inet6 families when enhanced network services mode is configured at the **[edit chassis network-services]** hierarchy level. When used with one of the chassis enhanced network services modes, firewall filters are generated in term-based format for use with MPC modules.

If enhanced network services are not configured for the chassis, the **enhanced-mode** statement is ignored and any enhanced mode firewall filters are generated in both term-based and compiled format (the default).

NOTE: You cannot attach enhanced mode filters to local loopback, management, or MS-DPC interfaces. These interfaces are processed by the Routing Engine kernel and DPC modules and can accept only compiled firewall filter format.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- *Network Services Mode Overview*
- *Firewall Filters and Enhanced Network Services Mode Overview*
- *Configuring a Filter for Use with Enhanced Network Services Mode*

family (Firewall)

Syntax

```
family protocol-family-name {
  filter filter-name {
    ... filter-configuration ...
  }
  prefix-action prefix-action-name {
    ... prefix-action-configuration ...
  }
  service-filter filter-name
    ... filter-configuration ...
  }
  simple-filter filter-name {
    ... filter-configuration ...
  }
}
```

Hierarchy Level

[edit firewall],
[edit logical-systems *logical-system-name* firewall]

Release Information

Statement introduced before Junos OS Release 7.4.
Logical systems support introduced in Junos OS Release 9.3.
simple-filter statement introduced in Junos OS Release 7.6.
any family type introduced in Junos OS Release 8.0 (not supported on PTX Series Packet Transport Routers).
bridge family type introduced in Junos OS Release 8.4 (MX Series routers only).

Description

Configure a firewall filter for IP version 4 (IPv4) or IP version 6 (IPv6) traffic. On the MX Series routers only, configure a firewall filter for Layer 2 traffic in a bridging environment.

Options

family-name—Version or type of addressing protocol:

- **any**—Protocol-independent match conditions.
- **bridge**—(MX Series routers only) Layer 2 packets that are part of bridging domain.
- **ccc**—Layer 2 switching cross-connects.
- **inet**—IPv4 addressing protocol.
- **inet6**—IPv6 addressing protocol.
- **mpls**—MPLS.
- **vpls**—Virtual private LAN service (VPLS).

The remaining statements are explained separately.

NOTE: The packet lengths that a policer considers depends on the address family of the firewall filter. See *Understanding the Frame Length for Policing Packets*.

Required Privilege Level

interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Guidelines for Configuring Service Filters on page 82
- Guidelines for Configuring Simple Filters on page 89

filter (Applying to a Logical Interface)

Syntax

```
filter {
  group filter-group-number;
  input filter-name;
  input-list [ filter-names ];
  output filter-name;
  output-list [ filter-names ];
}
```

Hierarchy Level

Protocol-independent firewall filter on MX Series router logical interface:

[edit interfaces *interface-name* unit *logical-unit-number*],
[edit logical-systems *logical-system-name* interfaces *interface-name* unit *logical-unit-number*]

All other standard firewall filters on all other devices:

[edit interfaces *interface-name* unit *logical-unit-number* family *family*],
[edit logical-systems *logical-system-name* interfaces *interface-name* unit *logical-unit-number* family *family*]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Apply a stateless firewall filter to a logical interface at a specific protocol level.

Options

group *filter-group-number*—Number of the group to which the interface belongs. Range: 1 through 255

input *filter-name*—Name of one filter to evaluate when packets are received on the interface.

input-list [*filter-names*]—Names of filters to evaluate when packets are received on the interface. Up to 16 filters can be included in a filter input list.

output *filter-name*—Name of one filter to evaluate when packets are transmitted on the interface.

output-list [*filter-names*]—Names of filters to evaluate when packets are transmitted on the interface. Up to 16 filters can be included in a filter output list.

Required Privilege Level

interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Guidelines for Applying Firewall Filters on page 25

filter (Configuring)

Syntax

```
filter filter-name {
  accounting-profile name;
  enhanced-mode;
  interface-shared;
  interface-specific;
  physical-interface-filter;
  term term-name {
    ... term configuration ...
  }
}
```

Hierarchy Level

[edit dynamic-profiles *profile-name* firewall family *family-name*],
[edit firewall family *family-name*],
[edit logical-systems *logical-system-name* firewall family *family-name*]

Release Information

Statement introduced before Junos OS Release 7.4.
Logical systems support introduced in Junos OS Release 9.3.
physical-interface-filter statement introduced in Junos OS Release 9.6.
Support at the **[edit dynamic-profiles ... family family-name]** hierarchy level introduced in Junos OS Release 11.4.
Support for the **interface-shared**> statement introduced in Junos OS Release 12.2.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Configure firewall filters.

Options

filter-name—Name that identifies the filter. This must be a non-reserved string of not more than 64 characters. To include spaces in the name, enclose it in quotation marks (" "). In Junos OS Release 9.0 and later, you can no longer use special characters within the name of a firewall filter. Firewall filter names are restricted from having the form __.*__ (beginning and ending with underscores) or __.* (beginning with an underscore.

The remaining statements are explained separately.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Guidelines for Applying Firewall Filters on page 25
- *Configuring Multifield Classifiers*
- *Using Multifield Classifiers to Set PLP*
- simple-filter (Configuring) on page 309

firewall

Syntax

```
firewall {
    atm-policer atm-policer-name {
        ... atm-policer-configuration ...
    }
    family protocol-family-name {
        ... protocol-family-configuration ...
    }
    filter ipv4-filter-name {
        ... ipv4-filter-configuration ...
    }
    hierarchical-policer hierarchical-policer-name {
        ... hierarchical-policer-configuration ...
    }
    interface-set interface-set-name {
        ... interface-set-configuration ...
    }
    policer two-color-policer-name {
        ... two-color-policer-configuration ...
    }
    three-color-policer three-color-policer-name {
        ... three-color-policer-configuration ...
    }
}
```

Hierarchy Level

[edit],
[edit logical-systems *logical-system-name*]
[edit dynamic-profiles *profile-name*],

Release Information

Statement introduced before Junos OS Release 7.4.
Logical systems support introduced in Junos OS Release 9.3.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Configure firewall filters.

The statements are explained separately.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Guidelines for Configuring Service Filters on page 82
- Guidelines for Configuring Simple Filters on page 89
- *Configuring Multifield Classifiers*
- *Using Multifield Classifiers to Set PLP*

interface-set

Syntax

```
interface-set interface-set-name {
  interface-name;
}
```

Hierarchy Level

[edit firewall],
[edit logical-systems *logical-system-name* firewall]

Release Information

Statement introduced before Junos OS Release 7.4.
Logical systems support introduced in Junos OS Release 9.3.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Configure an interface set.

Options

interface-name—Names of each interface to include in the interface set. You must specify more than one name.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Filtering Packets Received on an Interface Set Overview on page 63

interface-shared

Syntax

interface-shared;

Hierarchy Level

[edit firewall family *family-name* filter *filter-name*],
[edit dynamic-profiles *profile-name*firewall family *family-name* filter *filter-name*]

Release Information

Statement introduced in Junos OS Release 12.2.

Description

Set the interface-shared attribute for a firewall filter.

Required Privilege Level

interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- *Dynamic Firewall Filters Overview*
- *Classic Filters Overview*
- *Basic Classic Filter Syntax*

interface-specific (Firewall Filters)

Syntax interface-specific;

Hierarchy Level [edit firewall family *family-name* filter *filter-name*],
[edit logical-systems *logical-system-name* firewall family *family-name* filter *filter-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Logical systems support introduced in Junos OS Release 9.3.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Configure interface-specific names for firewall counters.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation
- *Configuring Firewall Filters and Policers for VPLS*
- Interface-Specific Firewall Filter Instances Overview on page 61

ip-version

Syntax ip-version *ip-version*;

Hierarchy Level [edit firewall family *family-name* filter *filter-name* term *term-name* from]

Release Information Statement introduced in Junos OS Release 10.1R1.
Option **ipv6** introduced in Junos OS Release 12.2R1.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Configure the IP version for the firewall filter.

Options ***ip-version***—Version of the IP addressing.
- **ipv4**—IP version 4
- **ipv6**—IP version 6

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation
- Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic on page 360

prefix-list

Syntax

```
prefix-list name {
  ip-addresses;
  apply-path path;
}
```

Hierarchy Level

[edit dynamic policy-options],
[edit logical-systems *logical-system-name* policy-options],
[edit policy-options]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5.
Support for configuration in the dynamic database introduced in Junos OS Release 9.5 for EX Series switches.
Support for the **vpls** protocol family introduced in Junos OS Release 10.2.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Define a list of IPv4 or IPv6 address prefixes for use in a routing policy statement or firewall filter statement.

You can configure up to 85,325 prefixes in each prefix list. To configure more than 85,325 prefixes, configure multiple prefix lists and apply them to multiple firewall filter terms.

Options

name—Name that identifies the list of IPv4or IPv6 address prefixes.

ip-addresses—List of IPv4 or IPv6 address prefixes, one IP address per line in the configuration.

The remaining statement is explained separately.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- *Understanding Prefix Lists for Use in Routing Policy Match Conditions*
- Firewall Filter Match Conditions Based on Address Fields on page 328
- *Example: Configuring Routing Policy Prefix Lists*
- Example: Configuring a Filter to Limit TCP Access to a Port Based On a Prefix List on page 97

protocol

Syntax protocol (tcp|udp);

Hierarchy Level [edit firewall family *family-name* filter *filter-name* term *term-name* from ip-version *ip-version*]

Release Information Statement introduced in Junos OS Release 10.1R1.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Configure the protocol field of IPv4 or the next-header field of the IPv6 address.

Options **tcp**—Transmission Control Protocol.

udp—User Datagram Protocol.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic on page 360

service-filter (Firewall)

Syntax

```
service-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      actions;
    }
  }
}
```

Hierarchy Level

[edit firewall family (inet | inet6],
[edit logical-systems *logical-system-name* firewall family (inet | inet6)]

Release Information

Statement introduced before Junos OS Release 7.4.
Logical systems support introduced in Junos OS Release 9.3.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Configure service filters.

Options

filter-name—Name that identifies the service filter. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose it in quotation marks (“ ”).

The remaining statements are explained separately.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Guidelines for Configuring Service Filters on page 82
- Guidelines for Applying Service Filters on page 84

simple-filter (Configuring)

Syntax

```
simple-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      forwarding-class class-name;
      loss-priority (high | low | medium);
    }
  }
}
```

Hierarchy Level

[edit firewall family inet],
[edit logical-systems *logical-system-name* firewall family inet]

Release Information

Statement introduced in Junos OS Release 7.6.
Logical systems support introduced in Junos OS Release 9.3.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Configure simple filters.

Options

filter-name—Name that identifies the simple filter. The name must be a non-reserved string of not more than 64 characters. No special characters are restricted. To include spaces in the name, enclose them in quotation marks (“ ”).

from—Match packet fields to values. If the **from** option is not included, all packets are considered to match and the actions and action modifiers in the **then** statement are taken.

match-conditions—One or more conditions to use to make a match.

term *term-name*—Define a simple-filter term. The name that identifies the term can contain letters, numbers, and hyphens (-), and can be up to 255 characters long. To include spaces in the name, enclose them in quotation marks (“ ”).

then—Actions to take on matching packets. If the **then** option is not included and a packet matches all the conditions in the **from** statement, the packet is accepted.

NOTE: Only **forwarding-class** and **loss-priority** are valid actions in a simple filter configuration.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- *simple-filter (Applying to an Interface)*
- Simple Filter Overview on page 87

- How Simple Filters Evaluate Packets on page 87
- Guidelines for Configuring Simple Filters on page 89
- Guidelines for Applying Simple Filters on page 92

source-address

Syntax source-address *address*;

Hierarchy Level [edit firewall family *family-name* filter *filter-name* term *term-name* from ip-version *ip-version*]

Release Information Statement introduced in Junos OS Release 10.1R1.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Configure the IPv4 or IPv6 address of the source node that sends the packet.

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation
- Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic on page 360

source-port

Syntax source-port <*source-port*>;

Hierarchy Level [edit firewall family *family-name* filter *filter-name* term *term-name* from ip-version *ip-version* protocol (tcp|udp)]

Release Information Statement introduced in Junos OS Release 10.1R1.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description Configure the source port of the Layer 4 header.

Options ***source-port***—The source port of the Layer 4 header.
Range: 0 through 65,535

Required Privilege Level firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation
- Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic on page 360

term (Firewall Filter)

Syntax

```
term term-name {
  from {
    match-conditions;
    ip-version ipv4 {
      match-conditions-mpls-ipv4-address;
        protocol (tcp | udp) {
          match conditions-mpls-ipv4-port;
        }
    }
  }
  then {
    actions;
  }
}
```

Hierarchy Level

[edit firewall family *family-name* filter *filter-name*],
[edit firewall family *family-name* service-filter *filter-name*],
[edit firewall family *family-name* simple-filter *filter-name*],
[edit logical-systems *logical-system-name* firewall family *family-name* filter *filter-name*],
[edit logical-systems *logical-system-name* firewall family *family-name* service-filter *filter-name*],
[edit logical-systems *logical-system-name* firewall family *family-name* simple-filter *filter-name*]

Release Information

Statement introduced before Junos OS Release 7.4.
filter option introduced in Junos OS Release 7.6.
Logical systems support introduced in Junos OS Release 9.3.
ip-version ipv4 support introduced in Junos OS Release 10.1.
Statement introduced in Junos OS Release 12.3R2 for EX Series switches.

Description

Define a firewall filter term.

Options

actions—(Optional) Actions to perform on the packet if conditions match. You can specify one *terminating action* supported for the specified filter type. If you do not specify a terminating action, the packets that match the conditions in the **from** statement are accepted by default. As an option, you can specify one or more *nonterminating actions* supported for the specified filter type.

filter-name—(Optional) For family ***family-name*** filter ***filter-name*** only, reference another standard stateless firewall filter from within this term.

from—(Optional) Match packet fields to values. If not included, all packets are considered to match and the actions and action modifiers in the **then** statement are taken.

match-conditions—One or more conditions to use to make a match on a packet.

match-conditions-mpls-ipv4-address—(MPLS-tagged IPv4 traffic only) One or more IP address match conditions to match on the IPv4 packet header. Supports network-based service in a core network with IPv4 packets as an inner payload of an MPLS packet with labels stacked up to five deep.

match-conditions-mpls-ipv4-port—(MPLS-tagged IPv4 traffic only) One or more UDP or TCP port match conditions to use to match a packet in an MPLS flow. Supports network-based service in a core network with IPv4 packets as an inner payload of an MPLS packet with labels stacked up to five deep.

term-name—Name that identifies the term. The name can contain letters, numbers, and hyphens (-) and can be up to 64 characters long. To include spaces in the name, enclose it in quotation marks (" ").

then—(Optional) Actions to take on matching packets. If not included and a packet matches all the conditions in the **from** statement, the packet is accepted.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- *Configuring Multifield Classifiers*
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32

tunnel-end-point

Syntax

```
tunnel-end-point tunnel-name {
  encapsulation-protocol [ protocol-options ];
  transport-protocol {
    destination-address destination-host-address;
    source-address source-host-address;
  }
}
```

Hierarchy Level

[edit firewall],
[edit logical-systems *logical-system-name* firewall]

Release Information

Statement introduced in Junos OS Release 12.3R2.

Description

On an MX Series router installed as an *encapsulator* (an ingress PE router) for filter-based IP tunneling, define a *tunnel template*. The template specifies a set of characteristics for transporting passenger protocol packets across an IP transport network.

Options

destination-host-address—IP address or address range of the de-encapsulator (the remote egress PE router).

encapsulation-protocol—Encapsulation protocol:

- **gre**—Filter-based GRE encapsulation is supported on IPv4 transport networks.

protocol-options—(Optional) Protocol-specific encapsulation options:

- **key *number***—An integer value that uniquely identifies a GRE IPv4 tunnel if multiple traffic flows share the same ***source-address*** and ***destination-address*** pair. Range: 1 through 0xFFFFFFFF (4,294,967,295 decimal)

  If a tunnel definition specifies GRE IPv4 tunneling using a key, the system includes the key in the GRE header whenever a Packet Forwarding Engine is instructed to use that tunnel definition to encapsulate a packet.

source-host-address—IP address or address range of the encapsulator (the local ingress PE router).

transport-protocol—The IP network protocol used to transport encapsulated passenger protocol packets:

- **ipv4**—IPv4 can transport IPv4, IPv6, or MPLS packets encapsulated using filter-based generic routing encapsulation (GRE).

tunnel-name—Name that identifies the tunnel template using a non-reserved string of not more than 64 characters. To include spaces in the name, enclose it in quotation marks (“ ”). You can reference a tunnel template name in an ingress firewall filter of type **inet**, **inet6**, **any**, or **mpls** by configuring the **encapsulate** terminating action.

Required Privilege Level

firewall—To view this statement in the configuration.
firewall-control—To add this statement to the configuration.

Related Documentation

- Understanding Filter-Based Tunneling Across IPv4 Networks on page 65
- Interfaces That Support Filter-Based Tunneling Across IPv4 Networks on page 68
- Components of Filter-Based Tunneling Across IPv4 Networks on page 69
- Example: Transporting IPv6 Traffic Across IPv4 Using Filter-Based Tunneling on page 217
- Firewall Filter Terminating Actions on page 384

PART 3

Administration

CHAPTER 24

Firewall Filters Standards

Supported Standards for Filtering

The Junos OS supports the following RFCs related to filtering:

- RFC 792, *Internet Control Message Protocol*
- RFC 2460, *Internet Protocol, Version 6 (IPv6)*
- RFC 2474, *Definition of the Differentiated Services (DS) Field*
- RFC 2475, *An Architecture for Differentiated Services*
- RFC 2597, *Assured Forwarding PHB Group*
- RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*
- RFC 4291, *IP Version 6 Addressing Architecture*
- RFC 4443, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*

Related Documentation

- Firewall Filters Overview on page 17
- Service Filter Overview on page 79
- Simple Filter Overview on page 87
- Firewall Filters in Logical Systems Overview on page 31

CHAPTER 25

Firewall Filters Reference

Using the CLI Editor in Configuration Mode

This topic describes some of the basic commands that you must use to enter configuration mode in the command-line interface (CLI) editor, navigate through the configuration hierarchy, get help, and commit or revert the changes that you make during the configuration session.

Task	Command/Statement	Example
Edit Your Configuration		
Enter configuration mode. When you first log in to the device, the device is in operational mode. You must explicitly enter configuration mode. When you do, the CLI prompt changes from **user@host>** to **user@host#** and the hierarchy level appears in square brackets.	**configure**	user@host> **configure** [edit] user@host#
Create a statement hierarchy. You can use the **edit** command to simultaneously create a hierarchy and move to that new level in the hierarchy. You cannot use the **edit** command to change the value of identifiers.	edit *hierarchy-level value*	[edit] user@host# **edit security zones security-zone myzone** [edit security zones security-zone myzone] user@host#
Create a statement hierarchy and set identifier values. The **set** command is similar to **edit** except that your current level in the hierarchy does not change.	set *hierarchy-level value*	[edit] user@host# **set security zones security-zone myzone** [edit] user@host#
Navigate the Hierarchy		

Task	Command/Statement	Example
Navigate down to an existing hierarchy level.	edit *hierarchy-level*	[edit] user@host# **edit security zones** [edit security zones] user@host#
Navigate up one level in the hierarchy.	up	[edit security zones] user@host# **up** [edit security] user@host#
Navigate to the top of the hierarchy.	top	[edit security zones] user@host# **top** [edit] user@host#
Commit or Revert Changes		
Commit your configuration.	commit	[edit] user@host# **commit** commit complete
Roll back changes from the current session. Use the **rollback** command to revert all changes from the current configuration session. When you run the **rollback** command before exiting your session or committing changes, the software loads the most recently committed configuration onto the device. You must enter the **rollback** statement at the **edit** level in the hierarchy.	rollback	[edit] user@host# **rollback** load complete
Exit Configuration Mode		
Commit the configuration and exit configuration mode.	commit and-quit	[edit] user@host# **commit and-quit** user@host>
Exit configuration mode without committing your configuration. You must navigate to the top of the hierarchy using the **up** or **top** commands before you can exit configuration mode.	exit	[edit] user@host# **exit** The configuration has been changed but not committed Exit with uncommitted changes? [yes,no] (yes)
Get Help		

Task	Command/Statement	Example
Display a list of valid options for the current hierarchy level.	?	[edit] user@host# **edit security zones ?** Possible completions: <[Enter]> Execute this command > functional-zone Functional zone > security-zone Security zones | Pipe through a command [edit]

Related Documentation

- *Understanding Junos OS CLI Configuration Mode*
- *Entering and Exiting the Junos OS CLI Configuration Mode*
- *Displaying the Current Junos OS Configuration*

CHAPTER 26

Introduction to Firewall Filter Match Conditions

Firewall Filter Match Conditions Based on Numbers or Text Aliases

This topic covers the following information:

Matching on a Single Numeric Value

You can specify a firewall filter match condition based on whether a particular packet field value is a specified numeric value. In the following example, a match occurs if the packet source port number is **25**:

```
[edit firewall family inet filter filter1 term term1 from]
user@host# set source-port 25
```

Matching on a Range of Numeric Values

You can specify a firewall filter match condition based on whether a particular packet field value falls within a specified range of numeric values. In the following example, a match occurs for source ports values from **1024** through **65,535**, inclusive:

```
[edit firewall family inet filter filter2 term term1 from]
user@host# set source-port 1024-65535
```

Matching on a Text Alias for a Numeric Value

You can specify a firewall filter match condition based on whether a particular packet field value is a numeric value that you specify by using a text string as an *alias* for the numeric value. In the following example, a match occurs if the packet source port number is **25**. For the **source-port** and **destination-port** match conditions, the text alias**smtp** corresponds to the numeric value **25**.

```
[edit firewall family inet filter filter3 term term1 from]
user@host# set source-port smtp
```

Matching on a List of Numeric Values or Text Aliases

You can specify a firewall filter match condition based on whether a particular packet field value matches any one of multiple numeric values or text aliases that you specify within square brackets and delimited by spaces. In the following example, a match occurs if the packet source port number is any of the following values: **20** (which corresponds to the text aliases **ftp-data**), **25**, or any value from **1024** through **65535**.

```
[edit firewall family inet filter filter3 term term1 from]
user@host# set source-port [ smtp ftp-data 25 1024-65535 ]
```

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Match Conditions Based on Bit-Field Values on page 324
- Firewall Filter Match Conditions Based on Address Fields on page 328
- Firewall Filter Match Conditions Based on Address Classes on page 336

Firewall Filter Match Conditions Based on Bit-Field Values

Match Conditions for Bit-Field Values

Table 17 on page 325 lists the firewall filter match conditions that are based on whether certain bit fields in a packet are set or not set. The second and third columns list the types of traffic for which the match condition is supported.

Table 17: Binary and Bit-Field Match Conditions for Firewall Filters

Bit-Field Match Condition	Match Values	Protocol Families for Standard Firewall Filters	Protocol Families for Service Filters
fragment-flags ***flags***	Hexadecimal values or text aliases for the three-bit IP fragmentation flags field in the IP header.	**family inet**	**family inet**
fragment-offset ***value***	Hexadecimal values or text aliases for the 13-bit fragment offset field in the IP header.	**family inet**	**family inet**
tcp-flags ***value***†	Hexadecimal values or text aliases for the low-order 6 bits of the 8-bit TCP flags field in the TCP header.	**family inet** **family inet6** **family vpls** **family bridge** **family ethernet-switching** (only for EX Series switches)	**family inet** **family inet6**

† Junos OS software does not automatically check the first fragment bit when matching TCP flags for IPv4 traffic. To check the first fragment bit for IPv4 traffic only, use the **first-fragment** match condition.

Match Conditions for Common Bit-Field Values or Combinations

Table 18 on page 325 describes firewall filter match conditions that are based on whether certain commonly used values or *combinations* of bit fields in a packet are set or not set.

You can use text synonyms to specify some common bit-field matches. In the previous example, you can specify **tcp-initial** as the same match condition.

NOTE:

Some of the numeric range and bit-field match conditions allow you to specify a text synonym. For a complete list of synonyms:

- If you are using the J-Web interface, select the synonym from the appropriate list.
- If you are using the CLI, type a question mark (?) after the **from** statement.

Table 18: Bit-Field Match Conditions for Common Combinations

Match Condition	Description	Protocol Families for Standard Firewall Filters	Protocol Families for Service Filters
first-fragment	Text alias for the bit-field match condition **fragment-offset 0**, which indicates the first fragment of a fragmented packet.	**family inet**	**family inet**
is-fragment	Text alias for the bit-field match condition **fragment-offset 0 except**, which indicates a trailing fragment of a fragmented packet.	**family inet**	**family inet**

Table 18: Bit-Field Match Conditions for Common Combinations *(continued)*

Match Condition	Description	Protocol Families for Standard Firewall Filters	Protocol Families for Service Filters
tcp-established	Alias for the bit-field match condition **tcp-flags "(ack | rst)"**, which indicates an established TCP session, but not the first packet of a TCP connection.	**family inet** **family inet6**	—
tcp-initial	Alias for the bit-field match condition **tcp-flags "(!ack & syn)"**, which indicates the first packet of a TCP connection, but not an established TCP session.	**family inet** **family inet6**	—

Logical Operators for Bit-Field Values

Table 19 on page 326 lists the logical operators you can apply to *single* bit-field values when specifying stateless firewall filter match conditions. The operators are listed in order, from highest precedence to lowest precedence. Operations are left-associative, meaning that the operations are processed from left to right.

Table 19: Bit-Field Logical Operators

Precedence Order	Bit-Field Logical Operator	Description
1	**(***complex-match-condition***)**	Grouping—The complex match condition is evaluated before any operators outside the parentheses are applied.
2	**!** ***match-condition***	Negation—A match occurs if the match condition is false.
3	***match-condition-1*** **,** ***match-condition-2***	Logical AND—A match occurs if both match conditions are true.
4	***match-condition-1*** **|** ***match-condition-2*** or ***match-condition-1*** **,** ***match-condition-2***	Logical OR—A match occurs if either match condition is true.

Matching on a Single Bit-Field Value or Text Alias

For the **fragment-flags** and **tcp-flags** bit-match conditions, you can specify firewall filter match conditions based on whether a particular bit in the packet field is set or not set.

- Numeric value to specify a single bit—You can specify a single bit-field match condition by using a numeric value that has one bit set. Depending on the match condition, you can specify a decimal value, a binary value, or a hexadecimal value. To specify a binary value, specify the number with the prefix **b**. To specify a hexadecimal value, specify the number with the prefix **0x**.

  In the following example, a match occurs if the **RST** bit in the TCP flags field is set:

  ```
  [edit firewall family inet filter filter_tcp_rst_number term term1 from]
  user@host# set tcp-flags 0x04
  ```

- Text alias to specify a single bit—You generally specify a single bit-field match condition by using a text alias enclosed in double-quotation marks (“ ”).

  In the following example, a match occurs if the **RST** bit in the TCP flags field is set:

  ```
  [edit firewall family inet filter filter_tcp_rst_alias term term1 from]
  user@host# set tcp-flags “rst”
  ```

Matching on Multiple Bit-Field Values or Text Aliases

You can specify a firewall filter match condition based on whether a particular set of bits in a packet field are set.

- Numeric values to specify multiple set bits—When you specify a numeric value whose binary representation has more than one set bit, the value is treated as a logical AND of the set bits.

  In the following example, the two match conditions are the same. A match occurs if either bit **0x01** or **0x02** is not set:

  ```
  [edit firewall family inet filter reset_or_not_initial_packet term term5 from]
  user@host# set tcp-flags “!0x3”
  user@host# set tcp-flags “!(0x01 & 0x02)”
  ```

- Text aliases that specify common bit-field matches—You can use text aliases to specify some common bit-field matches. You specify these matches as a single keyword.

  In the following example, the **tcp-established** condition, which is an alias for **“(ack | rst)”**, specifies that a match occurs on TCP packets other than the first packet of a connection:

  ```
  [edit firewall family inet filter reset_or_not_initial_packet term term6 from]
  user@host# set tcp-established
  ```

Matching on a Negated Bit-Field Value

To negate a match, precede the value with an exclamation point.

In the following example, a match occurs if the **RST** bit in the TCP flags field is *not* set:

```
[edit firewall family inet filter filter_tcp_rst term term1 from]
user@host# set tcp-flags “!rst”
```

Matching on the Logical OR of Two Bit-Field Values

You can use the *logical OR operator* (**|** or **,**) to specify that a match occurs if a bit field matches either of two bit-field values specified.

In the following example, a match occurs if the packet is *not* the initial packet in a TCP session:

```
[edit firewall family inet filter not_initial_packet term term3 from]
user@host# set tcp-flags "!syn | ack"
```

In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. In a packet that is not the initial packet in a TCP session, either the SYN flag is not set or the ACK flag is set.

Matching on the Logical AND of Two Bit-Field Values

You can use the *logical AND operator* (**&** or **+**) to specify that a match occurs if a bit field matches both of two bit-field values specified.

In the following example, a match occurs if the packet is the initial packet in a TCP session:

```
[edit firewall family inet filter initial_packet term term2 from]
user@host# set tcp-flags “syn & !ack”
```

In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. In a packet that is an initial packet in a TCP session, the SYN flag is set and the ACK flag is not set.

Grouping Bit-Field Match Conditions

You can use the *logical grouping notation* to specify that the complex match condition inside the parentheses is evaluated before any operators outside the parentheses are applied.

In the following example, a match occurs if the packet is a TCP reset or if the packet is not the initial packet in the TCP session:

```
[edit firewall family inet filter reset_or_not_initial_packet term term4 from]
user@host# set tcp-flags “!(syn & !ack) | rst”
```

In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. In a packet that is *not* the initial packet in a TCP session, the SYN flag is not set and the ACK field is set.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Match Conditions Based on Numbers or Text Aliases on page 323
- Firewall Filter Match Conditions Based on Address Fields on page 328
- Firewall Filter Match Conditions Based on Address Classes on page 336

Firewall Filter Match Conditions Based on Address Fields

You can configure firewall filter match conditions that evaluate packet address fields—IPv4 source and destination addresses, IPv6 source and destination addresses, or media access control (MAC) source and destination addresses—against specified addresses or prefix values.

- Implied Match on the ’0/0 except’ Address for Firewall Filter Match Conditions Based on Address Fields on page 329
- Matching an Address Field to a Subnet Mask or Prefix on page 329
- Matching an Address Field to an Excluded Value on page 330
- Matching Either IP Address Field to a Single Value on page 333
- Matching an Address Field to Noncontiguous Prefixes on page 334
- Matching an Address Field to a Prefix List on page 335

Implied Match on the '0/0 except' Address for Firewall Filter Match Conditions Based on Address Fields

Every firewall filter match condition based on a set of addresses or address prefixes is associated with an implicit match on the address **0.0.0.0/0 except** (for IPv4 or VPLS traffic) or **0:0:0:0:0:0:0:0/0 except** (for IPv6 traffic). As a result, any packet whose specified address field does not match any of the specified addresses or address prefixes fails to match the entire term.

Matching an Address Field to a Subnet Mask or Prefix

You can specify a single match condition to match a source address or destination address that falls within a specified address prefix.

IPv4 Subnet Mask Notation

For an IPv4 address, you can specify a subnet mask value rather than a prefix length. For example:

```
[edit firewall family inet filter filter_on_dst_addr term term3 from]
user@host# set address 10.0.0.10/255.0.0.255
```

Prefix Notation

To specify the address prefix, use the notation ***prefix/prefix-length***. In the following example, a match occurs if a destination address matches the prefix **10.0.0.0/8**:

```
[edit firewall family inet filter filter_on_dst_addr term term1 from]
user@host# set destination-address 10.0.0.0/8
```

Default Prefix Length for IPv4 Addresses

If you do not specify ***/prefix-length*** for an IPv4 address, the prefix length defaults to **/32**. The following example illustrates the default prefix value:

```
[edit firewall family inet filter filter_on_dst_addr term term2 from]
user@host# set destination-address 10
user@host# show
destination-address {
  10.0.0.0/32;
}
```

Default Prefix Length for IPv6 Addresses

If you do not specify ***/prefix-length*** for an IPv6 address, the prefix length defaults to **/128**. The following example illustrates the default prefix value:

```
[edit firewall family inet6 filter filter_on_dst_addr term term1 from]
user@host# set destination-address ::10
user@host# show
destination-address {
  ::10/128;
}
```

Default Prefix Length for MAC Addresses

If you do not specify */prefix-length* for a media access control (MAC) address of a VPLS, Layer 2 CCC, or Layer 2 bridging packet, the prefix length defaults to **/48**. The following example illustrates the default prefix value:

```
[edit firewall family vpls filter filter_on_dst_mac_addr term term1 from]
user@host# set destination-mac-address 01:00:0c:cc:cc:cd
user@host# show
destination-address {
  01:00:0c:cc:cc:cd/48;
}
```

Matching an Address Field to an Excluded Value

For the address-field match conditions, you can include the **except** keyword to specify that a match occurs for an address field that does not match the specified address or prefix.

Excluding IP Addresses in IPv4 or IPv6 Traffic

For the following IPv4 and IPv6 match conditions, you can include the **except** keyword to specify that a match occurs for an IP address field that does not match the specified IP address or prefix:

- **address** *address* **except**—A match occurs if either the source IP address or the destination IP address does not match the specified address or prefix.
- **source-address** *address* **except**—A match occurs if the source IP address does not match the specified address or prefix.
- **destination-address** *address* **except**—A match occurs if the destination IP address does not match the specified address or prefix.

In the following example, a match occurs for any IPv4 destination addresses that fall under the **192.168.10.0/8** prefix, except for addresses that fall under **192.168.0.0/16**. All other addresses implicitly do not match this condition.

```
[edit firewall family inet filter filter_on_dst_addr term term1 from]
user@host# set 192.168.0.0/16 except
user@host# set 192.168.10.0/8
user@host# show
destination-address {
  192.168.0.0/16 except;
  192.168.10.0/8;
}
```

In the following example, a match occurs for any IPv4 destination address that does not fall within the prefix **10.1.1.0/24**:

```
[edit firewall family inet filter filter_on_dst_addr term term24 from]
user@host# set destination-address 0.0.0.0/0
user@host# set destination-address 10.1.1.0/24 except
user@host# show
destination-address {
  0.0.0.0/0;
```

```
    10.1.1.0/24 except;
}
```

Excluding IP Addresses in VPLS or Layer 2 Bridging Traffic

For the following VPLS and Layer 2 bridging match conditions on MX Series routers and EX Series switches only, you can include the **except** keyword to specify that a match occurs for an IP address field that does not match the specified IP address or prefix:

- **ip-address *address* except**—A match occurs if either the source IP address or the destination IP address does not match the specified address or prefix.
- **source-ip-address *address* except**—A match occurs if the source IP address does not match the specified address or prefix.
- **destination-ip-address *address* except**—A match occurs if the destination IP address does not match the specified address or prefix.

In the following example for filtering VPLS traffic on an MX Series router and on an EX Series switch, a match occurs if the source IP address falls within the exception range of **55.0.1.0/255.0.255.0** and the destination IP address matches **55.0.0.0/8**:

```
[edit]
firewall {
   family vpls {
      filter fvpls {
         term 1 {
            from {
               ip-address {
                  55.0.0.0/8;
                  55.0.1.0/255.0.255.0 except;
               }
            }
            then {
               count from-55/8;
               discard;
            }
         }
      }
   }
}
```

Excluding MAC Addresses in VPLS or Layer 2 Bridging Traffic

For the following VPLS or Layer 2 bridging traffic match conditions, you can include the **except** keyword to specify that a match occurs for a MAC address field that does not match the specified MAC address or prefix:

- **source-mac-address *address* except**—A match occurs if the source MAC address does not match the specified address or prefix.
- **destination-mac-address *address* except**—A match occurs if either the destination MAC address does not match the specified address or prefix.

Excluding All Addresses Requires an Explicit Match on the '0/0' Address

If you specify a firewall filter match condition that consists of one or more address-*exception* match conditions (address match conditions that use the **except** keyword) but no *matchable* address match conditions, packets that do not match any of the configured prefixes fails the overall match operation. To configure a firewall filter term of address-exception match conditions to match any address that is not in the prefix list, include an explicit match of **0/0** so that the term contain a matchable address.

For the following example firewall filter for IPv4 traffic, the **from-trusted-addresses** term fails to discard matching traffic, and the **INTRUDERS-COUNT** counter is missing from the output of the **show firewall** operational mode command:

```
[edit]
user@host# show policy-options
prefix-list TRUSTED-ADDRESSES {
   10.2.1.0/24;
   192.168.122.0/24;
}

[edit firewall family inet filter protect-RE]
user@host# show
term from-trusted-addresses {
   from {
      source-prefix-list {
         TRUSTED-ADDRESSES except;
      }
      protocol icmp;
   }
   then {
      count INTRUDERS-COUNT;
      discard;
   }
}
   term other-icmp {
      from {
         protocol icmp;
      }
      then {
         count VALID-COUNT;
         accept;
      }
   }
   term all {
      then accept;
   }
```

```
[edit]
user@host# run show firewall
Filter: protect-RE
Counters:
Name                                                Bytes              Packets
VALID-COUNT                                          2770                   70
Filter: __default_bpdu_filter__
```

To cause a filter term of address-exception match conditions to match any address that is not in the prefix list, include an explicit match of **0/0** in the set of match conditions:

```
[edit firewall family inet filter protect-RE]
user@host# show term from-trusted-addresses
from {
  source-address {
    0.0.0.0/0;
  }
  source-prefix-list {
    TRUSTED-ADDRESSES except;
  }
  protocol icmp;
}
```

With the addition of the **0.0.0.0/0** source prefix address to the match condition, the **from-trusted-addresses** term discards matching traffic, and the INTRUDERS-COUNT counter displays in the output of the **show firewall** operational mode command:

```
[edit]
user@host# run show firewall
Filter: protect-RE
Counters:
Name                                                Bytes              Packets
VALID-COUNT                                          2770                   70
INTRUDERS-COUNT                                       420                    5
Filter: __default_bpdu_filter__
```

Matching Either IP Address Field to a Single Value

For IPv4 and IPv6 traffic and for VPLS and Layer 2 bridging traffic only on MX Series routers and on EX Series switches, you can use a single match condition to match a single address or prefix value to either the source or destination IP address field.

Matching Either IP Address Field in IPv4 or IPv6 Traffic

For IPv4 or IPv6 traffic, you can use a single match condition to specify the same address or prefix value as the match for either the source or destination IP address field. Instead of creating separate filter terms that specify the same address for the **source-address** and **destination-address** match conditions, you use only the **address** match condition. A match occurs if *either* the source IP address *or* the destination IP address matches the specified address or prefix.

If you use the **except** keyword with the **address** match condition, a match occurs if *both* the source IP address and the destination IP address match the specified value *before* the exception applies.

In a firewall filter term that specifies either the **source-address** or the **destination-address** match condition, you cannot also specify the **address** match condition.

Matching Either IP Address Field in VPLS or Layer 2 Bridging Traffic

For VPLS or Layer 2 bridging traffic on MX Series routers and EX Series switches only, you can use a single match condition to specify the same address or prefix value as the match for either the source or destination IP address field. Instead of creating separate

filter terms that specify the same address for the **source-ip-address** and **destination-ip-address** match conditions, you use only the **ip-address** match condition. A match occurs if *either* the source IP address *or* the destination IP address matches the specified address or prefix.

If you use the **except** keyword with the **ip-address** match condition, a match occurs if *both* the source IP address and the destination IP address match the specified value *before* the exception applies.

In a firewall filter term that specifies either the **source-ip-address** or the **destination-ip-address** match condition, you cannot also specify the **ip-address** match condition.

Matching an Address Field to Noncontiguous Prefixes

For IPv4 traffic only, specify a single match condition to match the IP source or destination address field to any prefix specified. The prefixes do not need to be contiguous. That is, the prefixes under the **source-address** or **destination-address** match condition do not need to be adjacent or neighboring to one another.

In the following example, a match occurs if a destination address matches either the **10.0.0.0/8** prefix or the **192.168.0.0/32** prefix:

```
[edit firewall family inet filter filter_on_dst_addr term term5 from]
user@host# set destination-address 10.0.0.0/8
user@host# set destination-address 192.168.0.0/32
user@host# show
destination-address {
   destination-address 10.0.0.0/8;
   destination-address 192.168.0.0/32;
}
```

The order in which you specify the prefixes within the match condition is not significant. Packets are evaluated against all the prefixes in the match condition to determine whether a match occurs. If prefixes overlap, longest-match rules are used to determine whether a match occurs. A match condition of noncontiguous prefixes includes an implicit **0/0 except** statement, which means that any prefix that does not match any prefix included in the match condition is explicitly considered not to match.

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.

Consider the following example:

```
[edit firewall family inet filter filter_on_src_addr term term1 from]
source-address {
   172.16.0.0/10;
   172.16.2.0/24 except;
   192.168.1.0;
   192.168.1.192/26 except;
   192.168.1.254;
   172.16.3.0/24; # ignored
```

```
    10.2.2.2 except; # ignored
}
```

Within the **source-address** match condition, two addresses are ignored. The **172.16.3.0/16** value is ignored because it falls under the address **172.16.0.0/10**, which is the same type. The **10.2.2.2 except** value is ignored because it is subsumed by the implicit **0.0.0.0/0 except** match value.

Suppose the following source IP address are evaluated by this firewall filter:

- Source IP address **172.16.1.2**—This address matches the **172.16.0.0/10** prefix, and thus the action in the **then** statement is taken.
- Source IP address **172.16.2.2**—This address matches the **172.16.2.0/24** prefix. Because this prefix is negated (that is, includes the **except** keyword), 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.
- Source IP address **10.1.2.3**—This address 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 of prefixes configured under the **source-address** match condition, and is considered to be a mismatch.

  The **172.16.3.0/24** 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 of prefixes configured under the **source-address** match condition.

BEST PRACTICE: 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 might be processed by the latter term before they are evaluated by any intervening terms. As a result, packets that should be rejected by the intervening terms might be accepted instead, or packets that should be accepted might be rejected instead.

To prevent this from occurring, we recommend that 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.

Matching an Address Field to a Prefix List

You can define a list of IPv4 or IPv6 address prefixes for use in a routing policy statement or in a stateless firewall filter match condition that evaluates packet address fields.

To define a list of IPv4 or IPv6 address prefixes, include the **prefix-list** ***prefix-list*** statement.

```
prefix-list name {
```

```
    ip-addresses;
    apply-path path;
}
```

You can include the statement at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems *logical-system-name* policy-options]**

After you have defined a prefix list, you can use it when specifying a firewall filter match condition based on an IPv4 or IPv6 address prefix.

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

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Match Conditions Based on Numbers or Text Aliases on page 323
- Firewall Filter Match Conditions Based on Bit-Field Values on page 324
- Firewall Filter Match Conditions Based on Address Classes on page 336

Firewall Filter Match Conditions Based on Address Classes

For IPv4 and IPv6 traffic only, you can use class-based firewall filter conditions to match packet fields based on source class or destination class.

- Source-Class Usage on page 336
- Destination-Class Usage on page 337
- Guidelines for Applying SCU or DCU Firewall Filters to Output Interfaces on page 337

Source-Class Usage

A *source class* is a set of source prefixes grouped together and given a class name. To configure a firewall filter term that matches an IP source address field to one or more source classes, use the **source-class *class-name*** match condition under the **[edit firewall family (inet | inet6) filter *filter-name* term *term-name* from]** hierarchy level.

Source-class usage (SCU) enables you to monitor the amount of traffic originating from a specific prefix. With this feature, usage can be tracked and customers can be billed for the traffic they receive.

Destination-Class Usage

A *destination class* is a set of destination prefixes grouped together and given a class name. To configure a firewall filter term that matches an IP destination address field to one or more destination classes, use the **destination-class *class-name*** match condition at the **[edit firewall family (inet | inet6) filter *filter-name* term *term-name* from]** hierarchy level.

Destination-class usage (DCU) enables you can track how much traffic is sent to a specific prefix in the core of the network originating from one of the specified interfaces.

Note, however, that DCU limits your ability to keep track of traffic moving in the reverse direction. It can account for all traffic that arrives on a core interface and heads toward a specific customer, but it cannot count traffic that arrives on a core interface from a specific prefix.

Guidelines for Applying SCU or DCU Firewall Filters to Output Interfaces

When applying a SCU or DCU firewall filter to an interface, keep the following guidelines in mind:

- Output interfaces—Class-based firewall filter match conditions work only for firewall filters that you apply to output interfaces. This is because the SCU and DCU are determined after route lookup occurs.
- Input interfaces—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.
- Output interfaces for tunnel traffic—SCU and DCU are not supported on the interfaces you configure as the output interface for tunnel traffic for transit packets exiting the router (or switch) through the tunnel.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Match Conditions for IPv4 Traffic on page 341
- Standard Firewall Filter Match Conditions for IPv6 Traffic on page 350
- *Source Class Usage Feature Guide*
- Firewall Filter Match Conditions Based on Numbers or Text Aliases on page 323
- Firewall Filter Match Conditions Based on Bit-Field Values on page 324
- Firewall Filter Match Conditions Based on Address Fields on page 328

CHAPTER 27

Firewall Filter Match Conditions and Actions

Firewall Filter Match Conditions for Protocol-Independent Traffic

You can configure a firewall filter with match conditions for protocol-independent traffic (**family any**).

To apply a protocol-indepdendent firewall filter to a logical interface, configure the **filter** statement under the logical unit.

NOTE:

On MX Series routers, attach a protocol-independent firewall filter to a logical interface by configuring the filter statement *directly* under the logical unit:

- **[edit intefaces *name* unit *number* filter]**
- **[edit logical-systems *name* intefaces *name* unit *number* filter]**

On all other supported devices, attach a protocol-independent firewall filter to a logical interface by configuring the filter statement under the protocol family (**family any**):

- **[edit intefaces *name* unit *number* family any filter]**
- **[edit logical-systems *name* intefaces *name* unit *number* family any filter]**

Table 20 on page 340 describes the ***match-conditions*** you can configure at the **[edit firewall family any filter *filter-name* term *term-name* from]** hierarchy level.

Table 20: Firewall Filter Match Conditions for Protocol-Independent Traffic

Match Condition	Description
forwarding-class *class*	Match the forwarding class of the packet. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**. For information about forwarding classes and router-internal output queues, see *Overview of Forwarding Classes*. NOTE: On T4000 Type 5 FPCs, a filter attached at the Layer 2 application point (that is, at the logical interface level) is unable to match with the forwarding class of a packet that is set by a Layer 3 classifier such as DSCP, DSCP V6, **inet-precedence**, and **mpls-exp**.
forwarding-class-except *class*	Do not match on the forwarding class. For details, see the **forwarding-class** match condition.
interface *interface-name*	Match the interface on which the packet was received. NOTE: If you configure this match condition with an interface that does not exist, the term does not match any packet.
interface-set *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the **interface-set** statement at the **[edit firewall]** hierarchy level. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.

Table 20: Firewall Filter Match Conditions for Protocol-Independent Traffic *(continued)*

Match Condition	Description
loss-priority *level*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. NOTE: This match condition is not supported on PTX series packet transport routers. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*.
loss-priority-except *level*	Do not match the PLP level. For details, see the **loss-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
packet-length *bytes*	Match the 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.
packet-length-except *bytes*	Do not match on the received packet length, in bytes. For details, see the **packet-length** match type.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Match Conditions for IPv4 Traffic

You can configure a firewall filter with match conditions for Internet Protocol version 4 (IPv4) traffic (**family inet**). Table 21 on page 341 describes the ***match-conditions*** you can configure at the **[edit firewall family inet filter *filter-name* term *term-name* from]** hierarchy level.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic

Match Condition	Description
address *address* **[except]**	Match the IPv4 source or destination address field unless the **except** option is included. If the option is included, do not match the IPv4 source or destination address field.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
ah-spi *spi-value*	(M Series routers, except M120 and M320) Match the IPsec authentication header (AH) security parameter index (SPI) value. NOTE: This match condition is not supported on PTX series packet transport routers.
ah-spi-except *spi-value*	(M Series routers, except M120 and M320) Do not match the IPsec AH SPI value. NOTE: This match condition is not supported on PTX series packet transport routers.
apply-groups	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
apply-groups-except	Specify which groups not to inherit configuration data from. You can specify more than one group name.
destination-address *address* **[except]**	Match the IPv4 destination address field unless the **except** option is included. If the option is included, do not match the IPv4 destination address field.. You cannot specify both the **address** and **destination-address** match conditions in the same term.
destination-class *class-names*	Match one or more specified destination class names (sets of destination prefixes grouped together and given a class name). For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 336. NOTE: This match condition is not supported on PTX series packet transport routers.
destination-class-except *class-names*	Do not match one or more specified destination class names. For details, see the **destination-class** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
destination-port *number*	Match the UDP or TCP destination port field. You cannot specify both the **port** and **destination-port** match conditions in the same term. If you configure this match condition, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. 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), **ldp** (646), **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** (49), **tacacs-ds** (65), **talk** (517), **telnet** (23), **tftp** (69), **timed** (525), **who** (513), or **xdmcp** (177).
destination-port-except *number*	Do not match the UDP or TCP destination port field. For details, see the **destination-port** match condition.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
destination-prefix-list *name* **[except]**	Match destination prefixes in the specified list unless the **except** option is included. If the option is included, do not match the destination prefixes in the specified list. Specify the name of a prefix list defined at the **[edit policy-options prefix-list** *prefix-list-name*] hierarchy level.

dscp *number*	Match the Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see *BA Classifier Overview*.
	
	You can specify a numeric value from **0** through **63**. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: **ef** (46).
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	  ◦ **af11** (10), **af12** (12), **af13** (14)
	  ◦ **af21** (18), **af22** (20), **af23** (22)
	  ◦ **af31** (26), **af32** (28), **af33** (30)
	  ◦ **af41** (34), **af42** (36), **af43** (38)

dscp-except *number*	Do not match on the DSCP number. For more information, see the **dscp** match condition.
esp-spi *spi-value*	Match the IPsec encapsulating security payload (ESP) SPI value. Match on this specific SPI value. You can specify the ESP SPI value in hexadecimal, binary, or decimal form. NOTE: This match condition is not supported on PTX series packet transport routers.
esp-spi-except *spi-value*	Match the IPsec ESP SPI value. Do not match on this specific SPI value. NOTE: This match condition is not supported on PTX series packet transport routers.
first-fragment	Match if the packet is the first fragment of a fragmented packet. Do not match if the packet is a trailing fragment of a fragmented packet. The first fragment of a fragmented packet has a fragment offset value of **0**. This match condition is an alias for the bit-field match condition **fragment-offset 0** match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions: **first-fragment** and **is-fragment**.
forwarding-class *class*	Match the forwarding class of the packet. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**. For information about forwarding classes and router-internal output queues, see *Overview of Forwarding Classes*.
forwarding-class-except *class*	Do not match the forwarding class of the packet. For details, see the **forwarding-class** match condition.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
fragment-flags *number*	(Ingress only) Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): **dont-fragment** (0x4), **more-fragments** (0x2), or **reserved** (0x8).
fragment-offset *value*	Match the 13-bit fragment offset field in the IP header. The value is the offset, in 8-byte units, in the overall datagram message to the data fragment. Specify a numeric value, a range of values, or a set of values. An offset value of **0** indicates the first fragment of a fragmented packet. The **first-fragment** match condition is an alias for the **fragment-offset 0** match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions (**first-fragment** and **is-fragment**).
fragment-offset-except *number*	Do not match the 13-bit fragment offset field.
icmp-code *number*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the **protocol icmp** match condition in the same term. If you configure this match condition, you must also configure the **icmp-type** *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: **ip-header-bad** (0), **required-option-missing** (1) • redirect: **redirect-for-host** (1), **redirect-for-network** (0), **redirect-for-tos-and-host** (3), **redirect-for-tos-and-net** (2) • time-exceeded: **ttl-eq-zero-during-reassembly** (1), **ttl-eq-zero-during-transit** (0) • 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)
icmp-code-except *message-code*	Do not match the ICMP message code field. For details, see the **icmp-code** match condition.
icmp-type *number*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the **protocol icmp** match condition in the same term. 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).

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
icmp-type-except *message-type*	Do not match the ICMP message type field. For details, see the **icmp-type** match condition.
interface *interface-name*	Match the interface on which the packet was received. NOTE: If you configure this match condition with an interface that does not exist, the term does not match any packet.
interface-group *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from 0 through 255. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the **[interfaces** *interface-name* **unit** *number* **family** *family* **filter group]** hierarchy level. NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63.
interface-group-except *group-number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the **interface-group** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
interface-set *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the **interface-set** statement at the **[edit firewall]** hierarchy level. NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
ip-options ***values***	Match the 8-bit IP option field, if present, to the specified value or list of values. In place of a numeric value, you can specify one of the following text synonyms (the option values are also listed): **loose-source-route** (131), **record-route** (7), **router-alert** (148), **security** (130), **stream-id** (136),**strict-source-route** (137), or **timestamp** (68). To match *any* value for the IP option, use the text synonym **any**. To match on *multiple* values, specify the list of values within square brackets ('**[**' and '**]**'). To match a *range* of values, use the value specification **[** ***value1-value2*** **]**. For example, the match condition **ip-options [0-147]** matches on an IP options field that contains the **loose-source-route**, **record-route**, or **security** values, or any other value from 0 through 147. However, this match condition does not match on an IP options field that contains only the **router-alert** value (148). For most interfaces, a filter term that specifies an **ip-option** match on one or more *specific* IP option values (a value other than **any**) causes packets to be sent to the Routing Engine so that the kernel can parse the IP option field in the packet header. • For a firewall filter term that specifies an **ip-option** match on one or more specific IP option values, you cannot specify the **count**, **log**, or **syslog** nonterminating actions *unless* you also specify the **discard** terminating action in the same term. This behavior prevents double-counting of packets for a filter applied to a transit interface on the router. • Packets processed on the kernel might be dropped in case of a system bottleneck. To ensure that matched packets are instead sent to the Packet Forwarding Engine (where packet processing is implemented in hardware), use the **ip-options any** match condition. The 10-Gigabit Ethernet Modular Port Concentrator (MPC), 100-Gigabit Ethernet MPC, 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers are capable of parsing the IP option field of the IPv4 packet header. For interfaces configured on those MPCs, *all* packets that are matched using the **ip-options** match condition are sent to the Packet Forwarding Engine for processing. NOTE: On M and T series routers, firewall filters cannot count **ip-options** packets on a per option type and per interface basis. A limited work around is to use the **show pfe statistics ip options** command to see **ip-options** statistics on a per PFE basis. See *show pfe statistics ip* for sample output.
ip-options-except ***values***	Do not match the IP option field to the specified value or list of values. For details about specifying the ***values***, see the **ip-options** match condition.
is-fragment	Match if the packet is a trailing fragment of a fragmented packet. Do not match the first fragment of a fragmented packet. NOTE: To match both first and trailing fragments, you can use two terms that specify different match conditions (**first-fragment** and **is-fragment**).

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
loss-priority *level*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. NOTE: This match condition is not supported on PTX series packet transport routers. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *BA Classifier Overview*.
loss-priority-except *level*	Do not match the PLP level. For details, see the **loss-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
packet-length *bytes*	Match the 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.
packet-length-except *bytes*	Do not match the length of the received packet, in bytes. For details, see the **packet-length** match type.
port *number*	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the **destination-port** match condition or the **source-port** match condition in the same term. If you configure this match condition, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed under **destination-port**.
port-except *number*	Do not match the UDP or TCP source or destination port field. For details, see the **port** match condition.
precedence *ip-precedence-value*	Match the 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). You can specify precedence in hexadecimal, binary, or decimal form.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
precedence-except *ip-precedence-value*	Do not match the 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). You can specify precedence in hexadecimal, binary, or decimal form.
prefix-list *name* **[except]**	Match the prefixes of the source or destination address fields to the prefixes in the specified list unless the **except** option is included. If the option is included, do not match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the **[edit policy-options prefix-list** *prefix-list-name***]** hierarchy level.
protocol *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **ospf** (89), **pim** (103), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).
protocol-except *number*	Do not match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **ospf** (89), **pim** (103), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).

rat-type *tech-type-value*	Match the radio-access technology (RAT) type specified in the 8-bit Tech-Type field of Proxy Mobile IPv4 (PMIPv4) access technology type extension. The technology type specifies the access technology through which the mobile device is connected to the access network.
	
	Specify a single value, a range of values, or a set of values. You can specify a technology type as a numeric value from 0 through 255 or as a system keyword.
	
	• The following numeric values are examples of well-known technology types:
	  ◦ Numeric value 1 matches IEEE 802.3.
	  ◦ Numeric value 2 matches IEEE 802.11a/b/g.
	  ◦ Numeric value 3 matches IEEE 802.16e
	  ◦ Numeric value 4 matches IEEE 802.16m.
	• Text string **eutran** matches 4G.
	• Text string **geran** matches 2G.
	• Text string **utran** matches 3G.

rat-type-except *tech-type-value*	Do not match the RAT Type.
service-filter-hit	Match a packet received from a filter where a **service-filter-hit** action was applied. NOTE: This match condition is not supported on PTX series packet transport routers.
source-address *address* **[except]**	Match the IPv4 address of the source node sending the packet unless the **except** option is included. If the option is included, do not match the IPv4 address of the source node sending the packet. You cannot specify both the **address** and **source-address** match conditions in the same term.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
source-class *class-names*	Match one or more specified source class names (sets of source prefixes grouped together and given a class name). For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 336. NOTE: This match condition is not supported on PTX series packet transport routers.
source-class-except *class-names*	Do not match one or more specified source class names. For details, see the **source-class** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
source-port *number*	Match the UDP or TCP source port field. You cannot specify the **port** and **source-port** match conditions in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the **destination-port** *number* match condition.
source-port-except *number*	Do not match the UDP or TCP source port field. For details, see the **source-port** match condition.
source-prefix-list *name* **[except]**	Match source prefixes in the specified list unless the **except** option is included. If the option is included, do not match the source prefixes in the specified list. Specify the name of a prefix list defined at the **[edit policy-options prefix-list** *prefix-list-name***]** hierarchy level.
tcp-established	Match TCP packets of an established TCP session (packets other than the first packet of a connection). This is an alias for **tcp-flags "(ack | rst)"**. This match condition does not implicitly check that the protocol is TCP. To check this, specify the **protocol tcp** match condition.

Table 21: Firewall Filter Match Conditions for IPv4 Traffic *(continued)*

Match Condition	Description
tcp-flags ***value***	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • **fin** (0x01) • **syn** (0x02) • **rst** (0x04) • **push** (0x08) • **ack** (0x10) • **urgent** (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the **tcp-established** and **tcp-initial** match conditions. If you configure this match condition, we recommend that you also configure the **protocol tcp** match statement in the same term to specify that the TCP protocol is being used on the port. For IPv4 traffic only, this match condition does not implicitly check whether the datagram contains the first fragment of a fragmented packet. To check for this condition for IPv4 traffic only, use the **first-fragment** match condition.
tcp-initial	Match the initial packet of a TCP connection. This is an alias for **tcp-flags "(!ack & syn)"**. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the **protocol tcp** match condition in the same term.
ttl ***number***	Match the IPv4 time-to-live number. Specify a TTL value or a range of TTL values. For ***number***, you can specify one or more values from **0** through **255**. This match condition is supported only on M120, M320, MX Series, and T Series routers.
ttl-except ***number***	Do not match on the IPv4 TTL number. For details, see the **ttl** match condition.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Standard Firewall Filter Match Conditions for IPv6 Traffic

You can configure a firewall filter with match conditions for Internet Protocol version 6 (IPv6) traffic (**family inet6**). Table 22 on page 351 describes the ***match-conditions*** you can configure at the **[edit firewall family inet6 filter** ***filter-name*** **term** ***term-name*** **from]** hierarchy level.

Table 22: Firewall Filter Match Conditions for IPv6 Traffic

Match Condition	Description
address *address* **[except]**	Match the IPv6 source or destination address field unless the **except** option is included. If the option is included, do not match the IPv6 source or destination address field.
apply-groups	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
apply-groups-except	Specify which groups not to inherit configuration data from. You can specify more than one group name.
destination-address *address* **[except]**	Match the IPv6 destination address field unless the **except** option is included. If the option is included, do not match the IPv6 destination address field. You cannot specify both the **address** and **destination-address** match conditions in the same term.
destination-class *class-names*	Match one or more specified destination class names (sets of destination prefixes grouped together and given a class name). NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 336.
destination-class-except *class-names*	Do not match one or more specified destination class names. For details, see the **destination-class** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
destination-port *number*	Match the UDP or TCP destination port field. You cannot specify both the **port** and **destination-port** match conditions in the same term. If you configure this match condition, we recommend that you also configure the **next-header udp** or **next-header tcp** match condition in the same term to specify which protocol is being used on the port. 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), **ldp** (646), **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** (49), **tacacs-ds** (65), **talk** (517), **telnet** (23), **tftp** (69), **timed** (525), **who** (513), or **xdmcp** (177).
destination-port-except *number*	Do not match the UDP or TCP destination port field. For details, see the **destination-port** match condition.
destination-prefix-list *prefix-list-name* **[except]**	Match the IPv6 destination prefix to the specified list unless the **except** option is included. If the option is included, do not match the IPv6 destination prefix to the specified list. The prefix list is defined at the **[edit policy-options prefix-list** *prefix-list-name***]** hierarchy level.

Table 22: Firewall Filter Match Conditions for IPv6 Traffic *(continued)*

Match Condition	Description
forwarding-class *class*	Match the forwarding class of the packet. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**. For information about forwarding classes and router-internal output queues, see *Overview of Forwarding Classes*.
forwarding-class-except *class*	Do not match the forwarding class of the packet. For details, see the **forwarding-class** match condition.
hop-limit *hop-limit*	Match the hop limit to the specified hop limit or set of hop limits. For ***hop-limit***, specify a single value or a range of values from 0 through 255.. Supported on interfaces hosted on MICs or MPCs in MX Series routers only.
hop-limit-except *hop-limit*	Do not match the hop limit to the specified hop limit or set of hop limits. For details, see the **hop-limit** match condition. Supported on interfaces hosted on MICs or MPCs in MX Series routers only.
icmp-code *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the **next-header icmp** or **next-header icmp6** match condition in the same term. If you configure this match condition, you must also configure the **icmp-type** *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: **ip6-header-bad** (0), **unrecognized-next-header** (1), **unrecognized-option** (2) • time-exceeded: **ttl-eq-zero-during-reassembly** (1), **ttl-eq-zero-during-transit** (0) • destination-unreachable: **administratively-prohibited** (1), **address-unreachable** (3), **no-route-to-destination** (0), **port-unreachable** (4)
icmp-code-except *message-code*	Do not match the ICMP message code field. For details, see the **icmp-code** match condition.
icmp-type *message-type*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the **next-header icmp** or **next-header icmp6** match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **destination-unreachable** (1), **echo-reply** (129), **echo-request** (128), **membership-query** (130), **membership-report** (131), **membership-termination** (132), **neighbor-advertisement** (136), **neighbor-solicit** (135), **node-information-reply** (140), **node-information-request** (139), **packet-too-big** (2), **parameter-problem** (4), **redirect** (137), **router-advertisement** (134), **router-renumbering** (138), **router-solicit** (133), or **time-exceeded** (3).

Table 22: Firewall Filter Match Conditions for IPv6 Traffic *(continued)*

Match Condition	Description
icmp-type-except *message-type*	Do not match the ICMP message type field. For details, see the **icmp-type** match condition.
interface *interface-name*	Match the interface on which the packet was received. NOTE: If you configure this match condition with an interface that does not exist, the term does not match any packet.
interface-group *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from **0** through **255**. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the **[interfaces** *interface-name* **unit** *number* **family** *family* **filter group]** hierarchy level. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63.
interface-group-except *group-number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the **interface-group** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
interface-set *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the **interface-set** statement at the **[edit firewall]** hierarchy level. NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.

Table 22: Firewall Filter Match Conditions for IPv6 Traffic *(continued)*

Match Condition	Description
ip-options ***values***	Match the 8-bit IP option field, if present, to the specified value or list of values. In place of a numeric value, you can specify one of the following text synonyms (the option values are also listed): **loose-source-route** (131), **record-route** (7), **router-alert** (148), **security** (130), **stream-id** (136),**strict-source-route** (137), or **timestamp** (68). To match *any* value for the IP option, use the text synonym **any**. To match on *multiple* values, specify the list of values within square brackets ('**[**' and '**]**'). To match a *range* of values, use the value specification **[** ***value1-value2*** **]**. For example, the match condition **ip-options [0-147]** matches on an IP options field that contains the **loose-source-route**, **record-route**, or **security** values, or any other value from 0 through 147. However, this match condition does not match on an IP options field that contains only the **router-alert** value (148). For most interfaces, a filter term that specifies an **ip-option** match on one or more *specific* IP option values (a value other than **any**) causes packets to be sent to the Routing Engine so that the kernel can parse the IP option field in the packet header. • For a firewall filter term that specifies an **ip-option** match on one or more specific IP option values, you cannot specify the **count**, **log**, or **syslog** nonterminating actions *unless* you also specify the **discard** terminating action in the same term. This behavior prevents double-counting of packets for a filter applied to a transit interface on the router. • Packets processed on the kernel might be dropped in case of a system bottleneck. To ensure that matched packets are instead sent to the Packet Forwarding Engine (where packet processing is implemented in hardware), use the **ip-options any** match condition. The 10-Gigabit Ethernet Modular Port Concentrator (MPC), 100-Gigabit Ethernet MPC, 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers are capable of parsing the IP option field of the IPv4 packet header. For interfaces configured on those MPCs, *all* packets that are matched using the **ip-options** match condition are sent to the Packet Forwarding Engine for processing.
ip-options-except ***values***	Do not match the IP option field to the specified value or list of values. For details about specifying the ***values***, see the **ip-options** match condition.
loss-priority ***level***	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, T Series routers and EX Series switches with Enhanced II Flexible PIC Concentrators (FPCs), you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. NOTE: This match condition is not supported on PTX series packet transport routers. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*.

Table 22: Firewall Filter Match Conditions for IPv6 Traffic *(continued)*

Match Condition	Description
loss-priority-except *level*	Do not match the PLP level. For details, see the **loss-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
next-header *header-type*	Match the 8-bit Next Header field that identifies the type of header between the IPv6 header and payload. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **ah** (51), **dstops** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **no-next-header** (59), **ospf** (89), **pim** (103), **routing** (43), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112). NOTE: **next-header icmp6** and **next-header icmpv6** match conditions perform the same function. **next-header icmp6** is the preferred option. **next-header icmpv6** is hidden in the Junos OS CLI.
next-header-except *header-type*	Do not match the 8-bit Next Header field that identifies the type of header between the IPv6 header and payload. For details, see the **next-header** match type.
packet-length *bytes*	Match the 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.
packet-length-except *bytes*	Do not match the length of the received packet, in bytes. For details, see the **packet-length** match type.
payload-protocol *protocol-type*	Match the payload protocol type. In place of the ***protocol-type*** numeric value, you can specify one of the following text synonyms (the field values are also listed): specify one or a set of of the following: **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58, **igmp** (2), **ipip** (4), **ipv6** (41), **no-next-header**, **ospf** (89), **pim** (103), **routing**, **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).
payload-protocol-except *protocol-type*	Do not match the payload protocol type. For details, see the **payload-protocol** match type.
port *number*	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the **destination-port** match condition or the **source-port** match condition in the same term. If you configure this match condition, we recommend that you also configure the **next-header udp** or **next-header tcp** match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed under **destination-port**.
port-except *number*	Do not match the UDP or TCP source or destination port field. For details, see the **port** match condition.

Table 22: Firewall Filter Match Conditions for IPv6 Traffic *(continued)*

Match Condition	Description
prefix-list *prefix-list-name* **[except]**	Match the prefixes of the source or destination address fields to the prefixes in the specified list unless the **except** option is included. If the option is included, do not match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the **[edit policy-options prefix-list** *prefix-list-name***]** hierarchy level.
service-filter-hit	Match a packet received from a filter where a **service-filter-hit** action was applied. NOTE: This match condition is not supported on PTX series packet transport routers.
source-address *address* **[except]**	Match the IPv6 address of the source node sending the packet unless the **except** option is included. If the option is included, do not match the IPv6 address of the source node sending the packet. You cannot specify both the **address** and **source-address** match conditions in the same term.
source-class *class-names*	Match one or more specified source class names (sets of source prefixes grouped together and given a class name). NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Firewall Filter Match Conditions Based on Address Classes" on page 336.
source-class-except *class-names*	Do not match one or more specified source class names. For details, see the **source-class** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
source-port *number*	Match the UDP or TCP source port field. You cannot specify the **port** and **source-port** match conditions in the same term. If you configure this match condition, we recommend that you also configure the **next-header udp** or **next-header tcp** match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the **destination-port** *number* match condition.
source-port-except *number*	Do not match the UDP or TCP source port field. For details, see the **source-port** match condition.
source-prefix-list *name* **[except]**	Match the IPv6 address prefix of the packet source field unless the **except** option is included. If the option is included, do not match the IPv6 address prefix of the packet source field. Specify a prefix list name defined at the **[edit policy-options prefix-list** *prefix-list-name***]** hierarchy level.
tcp-established	Match TCP packets other than the first packet of a connection. This is a text synonym for **tcp-flags "(ack | rst)"** (**0x14**). NOTE: This condition does not implicitly check that the protocol is TCP. To check this, specify the **protocol tcp** match condition. If you configure this match condition, we recommend that you also configure the **next-header tcp** match condition in the same term.

Table 22: Firewall Filter Match Conditions for IPv6 Traffic *(continued)*

Match Condition	Description
tcp-flags *flags*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • **fin** (0x01) • **syn** (0x02) • **rst** (0x04) • **push** (0x08) • **ack** (0x10) • **urgent** (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the **tcp-established** and **tcp-initial** match conditions. If you configure this match condition, we recommend that you also configure the **next-header tcp** match condition in the same term to specify that the TCP protocol is being used on the port.
tcp-initial	Match the initial packet of a TCP connection. This is a text synonym for **tcp-flags "(!ack & syn)"**. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the **next-header tcp** match condition in the same term.

traffic-class *number*	Match the 8-bit field that specifies the class-of-service (CoS) priority of the packet.
	
	This field was previously used as the type-of-service (ToS) field in IPv4.
	
	You can specify a numeric value from **0** through **63**. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: **ef** (46).
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	  • **af11** (10), **af12** (12), **af13** (14)
	  • **af21** (18), **af22** (20), **af23** (22)
	  • **af31** (26), **af32** (28), **af33** (30)
	  • **af41** (34), **af42** (36), **af43** (38)

traffic-class-except *number*	Do not match the 8-bit field that specifies the CoS priority of the packet. For details, see the **traffic-class** match description.

NOTE: If you specify an IPv6 address in a match condition (the **address**, **destination-address**, or **source-address** match conditions), use the syntax for text representations described in RFC 2373, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see *IPv6 Overview* and *Supported IPv6 Standards*.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Match Conditions for MPLS Traffic

You can configure a firewall filter with match conditions for MPLS traffic (**family mpls**).

NOTE: The **input-list** *filter-names* and **output-list** *filter-names* statements for firewall filters for the **mpls** protocol family are supported on all interfaces with the exception of management interfaces and internal Ethernet interfaces (**fxp** or **em0**), loopback interfaces (**lo0**), and USB modem interfaces (**umd**).

Table 23 on page 358 describes the ***match-conditions*** you can configure at the **[edit firewall family mpls filter** *filter-name* **term** *term-name* **from]** hierarchy level.

Table 23: Firewall Filter Match Conditions for MPLS Traffic

Match Condition	Description
apply-groups	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
apply-groups-except	Specify which groups not to inherit configuration data from. You can specify more than one group name.
exp *number*	Experimental (EXP) bit number or range of bit numbers in the MPLS header. For ***number***, you can specify one or more values from **0** through **7** in decimal, binary, or hexadecimal format. NOTE: This match condition is not supported on PTX series packet transport routers.
exp-except *number*	Do not match on the EXP bit number or range of bit numbers in the MPLS header. For ***number***, you can specify one or more values from **0** through **7**. NOTE: This match condition is not supported on PTX series packet transport routers.
forwarding-class *class*	Forwarding class. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.
forwarding-class-except *class*	Do not match on the forwarding class. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.

Table 23: Firewall Filter Match Conditions for MPLS Traffic *(continued)*

Match Condition	Description
interface *interface-name*	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received. NOTE: If you configure this match condition with an interface that does not exist, the term does not match any packet.
interface-set *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the **interface-set** statement at the **[edit firewall]** hierarchy level. NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.
ip-version *number*	(Interfaces on Enhanced Scaling flexible PIC concentrators [FPCs] on supported T Series routers only) Inner IP version. To match MPLS-tagged IPv4 packets, match on the text synonym **ipv4**. NOTE: This match condition is not supported on PTX series packet transport routers.
loss-priority *level*	Match the packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*.
loss-priority-except *level*	Do not match the PLP level. For details, see the **loss-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Match Conditions for MPLS-Tagged IPv4 or IPv6 Traffic

This topic covers the following information:

Matching on IPv4 or IPv6 Packet Header Address or Port Fields in MPLS Flows

To support network-based service in a core network, you can configure a firewall filter that matches Internet Protocol version 4 (IPv4) or version 6 (IPv6) packet header fields in MPLS traffic (**family mpls**). The firewall filter can match IPv4 or IPv6 packets as an inner payload of an MPLS packet that has a single MPLS label or up to five MPLS labels stacked together. You can configure match conditions based on IPv4 addresses and IPv4 port numbers or IPv6 addresses and IPv6 port numbers in the header.

Firewall filters based on MPLS-tagged IPv4 headers are supported for interfaces on Enhanced Scaling flexible PIC concentrators (FPCs) on T320, T640, T1600, TX Matrix, and TX Matrix Plus routers only. However, the firewall filters based on MPLS-tagged IPv6 headers are supported for interfaces on the Type 5 FPC on T4000 Core Routers only. The feature is not supported for the router loopback interface (**lo0**), the router management interface (**fxp0** or **em0**), or USB modem interfaces (**umd**).

To configure a firewall filter term that matches an address or port fields in the Layer 4 header of packets in an MPLS flow, you use the **ip-version ipv4** match condition to specify that the term is to match packets based on inner IP fields:

- To match an MPLS-tagged IPv4 packet on the source or destination address field in the IPv4 header, specify the match condition at the **[edit firewall family mpls filter *filter-name* term *term-name* from ip-version ipv4]** hierarchy level.
- To match an MPLS-tagged IPv4 packet on the source or destination port field in the Layer 4 header, specify the match condition at the **[edit firewall family mpls filter *filter-name* term *term-name* from ip-version ipv4 protocol (udp | tcp)]** hierarchy level.

To configure a firewall filter term that matches an address or port fields in the IPv6 header of packets in an MPLS flow, you use the **ip-version ipv6** match condition to specify that the term is to match packets based on inner IP fields:

- To match an MPLS-tagged IPv6 packet on the source or destination address field in the IPv6 header, specify the match condition at the **[edit firewall family mpls filter *filter-name* term *term-name* from ip-version ipv6]** hierarchy level.
- To match an MPLS-tagged IPv6 packet on the source or destination port field in the Layer 4 header, specify the match condition at the **[edit firewall family mpls filter *filter-name* term *term-name* from ip-version ipv6 protocol (udp | tcp)]** hierarchy level.

IP Address Match Conditions for MPLS Traffic

Table 24 on page 361 describes the IP address-specific match conditions you can configure at the **[edit firewall family mpls filter *filter-name* term *term-name* from ip-version *ip-version*]** hierarchy level.

Table 24: IP Address-Specific Firewall Filter Match Conditions for MPLS Traffic

Match Condition	Description
destination-address *address*	Match the address of the destination node to receive the packet.
destination-address *address* **except**	Do not match the address of the destination node to receive the packet.
protocol *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **ospf** (89), **pim** (103), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).
source-address *address*	Match the address of the source node sending the packet.
source-address *address* **except**	Do not match the address of the source node sending the packet.

IP Port Match Conditions for MPLS Traffic

Table 25 on page 362 describes the IP port-specific ***match-conditions*** you can configure at the **[edit firewall family mpls filter *filter-name* term *term-name* from ip-version *ip-version* protocol (udp | tcp)]** hierarchy level.

Table 25: IP Port-Specific Firewall Filter Match Conditions for MPLS Traffic

Match Condition	Description
destination-port *number*	Match on the UDP or TCP destination port field. 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), **ldp** (646), **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** (49), **tacacs-ds** (65), **talk** (517), **telnet** (23), **tftp** (69), **timed** (525), **who** (513), or **xdmcp** (177).
destination-port-except *number*	Do not match on the UDP or TCP destination port field. In place of the numeric value, you can specify one of the text synonyms listed with the **destination-port** match condition.
source-port *number*	Match on the TCP or UDP source port field. In place of the numeric field, you can specify one of the text synonyms listed under **destination-port**.
source-port-except *number*	Do not match on the TCP or UDP source port field.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Match Conditions for VPLS Traffic

In the **from** statement in the VPLS 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. You can also specify multiple source addresses 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. For example, the negated match condition for **forwarding-class** is **forwarding-class-except**. 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.

You can configure a firewall filter with match conditions for Virtual Private LAN Service (VPLS) traffic (**family vpls**). Table 26 on page 363 describes the ***match-conditions*** you can configure at the **[edit firewall family vpls filter *filter-name* term *term-name* from]** hierarchy level.

NOTE: Not all match conditions for VPLS traffic are supported on all routing platforms or switching platforms. A number of match conditions for VPLS traffic are supported only on MX Series 3D Universal Edge Routers.

In the VPLS documentation, the word *router* in terms such as *PE router* is used to refer to any device that provides routing functions.

Table 26: Firewall Filter Match Conditions for VPLS Traffic

Match Condition	Description
destination-mac-address *address*	Match the destination media access control (MAC) address of a VPLS packet.
destination-port *number*	(MX Series routers and EX Series switches only) Match the UDP or TCP destination port field. You cannot specify both the **port** and **destination-port** match conditions in the same term. 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), **ldp** (646), **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** (49), **tacacs-ds** (65), **talk** (517), **telnet** (23), **tftp** (69), **timed** (525), **who** (513), or **xdmcp** (177).
destination-port-except *number*	(MX Series routers and EX Series switches only) Do not match on the TCP or UDP destination port field. You cannot specify both the **port** and **destination-port** match conditions in the same term.
destination-prefix-list *name*	(MX Series routers and EX Series switches only) Match destination prefixes in the specified list. Specify the name of a prefix list defined at the **[edit policy-options prefix-list** *prefix-list-name*] hierarchy level. NOTE: VPLS prefix lists support only IPv4 addresses. IPv6 addresses included in a VPLS prefix list will be discarded.
destination-prefix-list *name* **except**	(MX Series routers and EX Series switches only) Do not match destination prefixes in the specified list. For more information, see the **destination-prefix-list** match condition.

Table 26: Firewall Filter Match Conditions for VPLS Traffic *(continued)*

Match Condition	Description
dscp *number*	(MX Series routers and EX Series switches only) Match the Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see the *BA Classifier Overview*. You can specify a numeric value from **0** through **63**. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: **ef** (46). • RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: **af11** (10), **af12** (12), **af13** (14), **af21** (18), **af22** (20), **af23** (22), **af31** (26), **af32** (28), **af33** (30), **af41** (34), **af42** (36), **af43** (38)
dscp-except *number*	(MX Series routers and EX Series switches only) Do not match on the DSCP. For details, see the **dscp** match condition.
ether-type *values*	Match the 2-octet IEEE 802.3 Length/EtherType field to the specified value or list of values. You can specify decimal or hexadecimal values from 0 through 65535 (0xFFFF). A value from 0 through 1500 (0x05DC) specifies the length of an Ethernet Version 1 frame. A value from 1536 (0x0600) through 65535 specifies the EtherType (nature of the MAC client protocol) of an Ethernet Version 2 frame. In place of the numeric value, you can specify one of the following text synonyms (the hexadecimal values are also listed): **aarp** (0x80F3), **appletalk** (0x809B), **arp** (0x0806), **ipv4** (0x0800), **ipv6** (0x86DD), **mpls-multicast** (0x8848), **mpls-unicast** (0x8847), **oam** (0x8902), **ppp** (0x880B), **pppoe-discovery** (0x8863), **pppoe-session** (0x8864), or **sna** (0x80D5).
ether-type-except *values*	Do not match the 2-octet Length/EtherType field to the specified value or list of values. For details about specifying the ***values***, see the **ether-type** match condition.
forwarding-class *class*	Match the forwarding class. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.
forwarding-class-except *class*	Do not match the forwarding class. For details, see the **forwarding-class** match condition.

Table 26: Firewall Filter Match Conditions for VPLS Traffic *(continued)*

Match Condition	Description
icmp-code *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the **next-header icmp** or **next-header icmp6** match condition in the same term. If you configure this match condition, you must also configure the **icmp-type** *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: **ip6-header-bad** (0), **unrecognized-next-header** (1), **unrecognized-option** (2) • time-exceeded: **ttl-eq-zero-during-reassembly** (1), **ttl-eq-zero-during-transit** (0) • destination-unreachable: **address-unreachable** (3), **administratively-prohibited** (1), **no-route-to-destination** (0), **port-unreachable** (4)
icmp-code-except *message-code*	Do not match the ICMP message code field. For details, see the **icmp-code** match condition.
icmp-code *number*	(MX Series routers and EX Series switches only) Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the **ip-protocol icmp** or **ip-protocol icmp6** match condition in the same term. If you configure this match condition, you must also configure the **icmp-type** *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: **ip6-header-bad** (0), **unrecognized-next-header** (1), **unrecognized-option** (2) • time-exceeded: **ttl-eq-zero-during-reassembly** (1), **ttl-eq-zero-during-transit** (0) • destination-unreachable: **address-unreachable** (3), **administratively-prohibited** (1), **no-route-to-destination** (0), **port-unreachable** (4)
icmp-code-except *number*	(MX Series routers and EX Series switches only) Do not match on the ICMP code field. For details, see the **icmp-code** match condition.
icmp-type *number*	(MX Series routers and EX Series switches only) Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the **ip-protocol icmp**, **ip-protocol icmp6**, or **ip-protocol icmpv6** match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **destination-unreachable** (1), **echo-reply** (129), **echo-request** (128), **membership-query** (130), **membership-report** (131), **membership-termination** (132), **neighbor-advertisement** (136), **neighbor-solicit** (135), **node-information-reply** (140), **node-information-request** (139), **packet-too-big** (2), **parameter-problem** (4), **redirect** (137), **router-advertisement** (134), **router-renumbering** (138), **router-solicit** (133), or **time-exceeded** (3).

Table 26: Firewall Filter Match Conditions for VPLS Traffic *(continued)*

Match Condition	Description
icmp-type-except *number*	(MX Series routers and EX Series switches only) Do not match the ICMP message type field. For details, see the **icmp-type** match condition.
interface *interface-name*	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received. NOTE: If you configure this match condition with an interface that does not exist, the term does not match any packet.
interface-group *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For *group-number*, specify a single value or a range of values from **0** through **255**. To assign a logical interface to an interface group *group-number*, specify the *group-number* at the **[interfaces** *interface-name* **unit** *number* **family** *family* **filter group]** hierarchy level. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63. NOTE: This match condition is not supported on T4000 Type 5 FPCs.
interface-group-except *group-name*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the **interface-group** match condition. NOTE: This match condition is not supported on T4000 Type 5 FPCs.
interface-set *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the **interface-set** statement at the **[edit firewall]** hierarchy level. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.
ip-address *address*	(MX Series routers and EX Series switches only) 32-bit address that supports the standard syntax for IPv4 addresses.
ip-destination-address *address*	(MX Series routers and EX Series switches only) 32-bit address that is the final destination node address for the packet.
ip-precedence *ip-precedence-field*	(MX Series routers and EX Series switches only) 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).
ip-precedence-except *ip-precedence-field*	(MX Series routers and EX Series switches only) Do not match on the IP precedence field.
ip-protocol *number*	(MX Series routers and EX Series switches only) IP protocol field.
ip-protocol-except *number*	(MX Series routers and EX Series switches only) Do not match on the IP protocol field.
ip-source-address *address*	(MX Series routers and EX Series switches only) IP address of the source node sending the packet.

Table 26: Firewall Filter Match Conditions for VPLS Traffic *(continued)*

Match Condition	Description
learn-vlan-1p-priority *number*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from **0** through **7**. Compare with the **user-vlan-1p-priority** match condition.
learn-vlan-1p-priority-except *number*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p learned VLAN priority bits. For details, see the **learn-vlan-1p-priority** match condition.
learn-vlan-dei	(MX Series routers and EX Series switches only) Match the user VLAN ID drop eligability indicator (DEI) bit.
learn-vlan-dei-except	(MX Series routers and EX Series switches only) Do not match the user VLAN ID DEI bit.
learn-vlan-id *number*	(MX Series routers and EX Series switches only) VLAN identifier used for MAC learning.
learn-vlan-id-except *number*	(MX Series routers and EX Series switches only) Do not match on the VLAN identifier used for MAC learning.
loss-priority *level*	Packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs) and EX Series switches, you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. For information about the **tri-color** statement and about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*.
loss-priority-except *level*	Do not match on the packet loss priority level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *BA Classifier Overview*.
port *number*	(MX Series routers and EX Series switches only) TCP or UDP source or destination port. You cannot specify both the **port** match condition and either the **destination-port** or **source-port** match condition in the same term.
port-except *number*	(MX Series routers and EX Series switches only) Do not match on the TCP or UDP source or destination port. You cannot specify both the **port** match condition and either the **destination-port** or **source-port** match condition in the same term.

Table 26: Firewall Filter Match Conditions for VPLS Traffic *(continued)*

Match Condition	Description
prefix-list *name*	(MX Series routers and EX Series switches only) Match the destination or source prefixes in the specified list. Specify the name of a prefix list defined at the **[edit policy-options prefix-list** *prefix-list-name*] hierarchy level. NOTE: VPLS prefix lists support only IPV4 addresses. IPV6 addresses included in a VPLS prefix list will be discarded.
prefix-list *name* **except**	(MX Series routers and EX Series switches only) Do not match the destination or source prefixes in the specified list. For more information, see the **destination-prefix-list** match condition.
source-mac-address *address*	Source MAC address of a VPLS packet.
source-port *number*	(MX Series routers and EX Series switches only) TCP or UDP source port field. You cannot specify the **port** and **source-port** match conditions in the same term.
source-port-except *number*	(MX Series routers and EX Series switches only) Do not match on the TCP or UDP source port field. You cannot specify the **port** and **source-port** match conditions in the same term.
source-prefix-list *name*	(MX Series routers and EX Series switches only) Match the source prefixes in the specified prefix list. Specify a prefix list name defined at the **[edit policy-options prefix-list** *prefix-list-name***]** hierarchy level. NOTE: VPLS prefix lists support only IPV4 addresses. IPV6 addresses included in a VPLS prefix list will be discarded.
source-prefix-list *name* **except**	(MX Series routers and EX Series switches only) Do not match the source prefixes in the specified prefix list. For more information, see the **source-prefix-list** match condition.
tcp-flags *flags*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • **fin** (0x01) • **syn** (0x02) • **rst** (0x04) • **push** (0x08) • **ack** (0x10) • **urgent** (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. If you configure this match condition for IPv6 traffic, we recommend that you also configure the **next-header tcp** match condition in the same term to specify that the TCP protocol is being used on the port.
traffic-type *type-name*	(MX Series routers and EX Series switches only) Traffic type. Specify **broadcast**, **multicast**, **unknown-unicast**, or **known-unicast**.

Table 26: Firewall Filter Match Conditions for VPLS Traffic *(continued)*

Match Condition	Description
traffic-type-except *type-name*	(MX Series routers and EX Series switches only) Do not match on the traffic type. Specify **broadcast**, **multicast**, **unknown-unicast**, or **known-unicast**.
user-vlan-1p-priority *number*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p user priority bits in the customer VLAN tag (the inner tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from **0** through **7**. Compare with the **learn-vlan-1p-priority** match condition.
user-vlan-1p-priority-except *number*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p user priority bits. For details, see the **user-vlan-1p-priority** match condition.
user-vlan-id *number*	(MX Series routers and EX Series switches only) Match the first VLAN identifier that is part of the payload.
user-vlan-id-except *number*	(MX Series routers and EX Series switches only) Do not match on the first VLAN identifier that is part of the payload.
vlan-ether-type *value*	VLAN Ethernet type field of a VPLS packet.
vlan-ether-type-except *value*	Do not match on the VLAN Ethernet type field of a VPLS packet.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Match Conditions for Layer 2 CCC Traffic

You can configure a firewall filter with match conditions for Layer 2 circuit cross-connect (CCC) traffic (**family ccc**).

The following restrictions apply to firewall filters for Layer 2 CCC traffic:

- The **input-list** ***filter-names*** and **output-list** ***filter-names*** statements for firewall filters for the **ccc** protocol family are supported on all interfaces with the exception of management interfaces and internal Ethernet interfaces (**fxp** or **em0**), loopback interfaces (**lo0**), and USB modem interfaces (**umd**).
- Only on MX Series routers and EX Series switches, you cannot apply a Layer 2 CCC stateless firewall filter (a firewall filter configured at the **[edit firewall filter family ccc]** hierarchy level) as an output filter. On MX Series routers and EX Series switches, firewall filters configured for the **family ccc** statement can be applied only as input filters.

Table 27 on page 370 describes the ***match-conditions*** you can configure at the **[edit firewall family ccc filter** ***filter-name*** **term** ***term-name*** **from]** hierarchy level.

Table 27: Firewall Filter Match Conditions for Layer 2 CCC Traffic

Match Condition	Description
apply-groups	Specify which groups to inherit configuration data from. You can specify more than one group name. You must list them in order of inheritance priority. The configuration data in the first group takes priority over the data in subsequent groups.
apply-groups-except	Specify which groups not to inherit configuration data from. You can specify more than one group name.
destination-mac-address *address*	(MX Series routers and EX Series switches only) Match the destination media access control (MAC) address of a virtual private LAN service (VPLS) packet. To have packets correctly evaluated by this match condition when applied to egress traffic flowing over a CCC circuit from a logical interface on an I-chip DPC in a Layer 2 virtual private network (VPN) routing instance, you must make a configuration change to the Layer 2 VPN routing instance. You must explicitly disable the use of a control word for traffic flowing out over a Layer 2 circuit. The use of a control word is enabled by default for Layer 2 VPN routing instances to support the emulated virtual circuit (VC) encapsulation for Layer 2 circuits. To explicitly disable the use of a control word for Layer 2 VPNs, include the **no-control-word** statement at either of the following hierarchy levels: • **[edit routing-instances** *routing-instance-name* **protocols l2vpn]** • **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name* **protocols l2vpn]** NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see *Disabling the Control Word for Layer 2 VPNs*.
forwarding-class *class*	Forwarding class. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.
forwarding-class-except *class*	Do not match on the forwarding class. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.
interface-group *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For ***group-number***, specify a single value or a range of values from **0** through **255**. To assign a logical interface to an interface group ***group-number***, specify the ***group-number*** at the **[interfaces** *interface-name* **unit** *number* **family** *family* **filter group]** hierarchy level. NOTE: This match condition is not supported on PTX series packet transport routers. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63.
interface-group-except *number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the **interface-group** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.

Table 27: Firewall Filter Match Conditions for Layer 2 CCC Traffic *(continued)*

Match Condition	Description
learn-vlan-1p-priority *number*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from **0** through **7**. Compare with the **user-vlan-1p-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
learn-vlan-1p-priority-except *number*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p learned VLAN priority bits. For details, see the **learn-vlan-1p-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
loss-priority *level*	Packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*.
loss-priority-except *level*	Do not match on the packet loss priority level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. NOTE: This match condition is not supported on PTX series packet transport routers. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *BA Classifier Overview*.
user-vlan-1p-priority *number*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p user priority bits in the customer VLAN tag (the inner tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from **0** through **7**. Compare with the **learn-vlan-1p-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.
user-vlan-1p-priority-except *number*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p user priority bits. For details, see the **user-vlan-1p-priority** match condition. NOTE: This match condition is not supported on PTX series packet transport routers.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384

- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Match Conditions for Layer 2 Bridging Traffic

Only on MX Series routers and EX Series switches, you can configure a standard stateless firewall filter with match conditions for Layer 2 bridging traffic (**family bridge**). Table 28 on page 372 describes the ***match-conditions*** you can configure at the **[edit firewall family bridge filter *filter-name* term *term-name* from]** hierarchy level.

Table 28: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series 3D Universal Edge Routers and EX Series switches Only)

Match Condition	Description
destination-mac-address *address*	Destination media access control (MAC) address of a Layer 2 packet in a bridging environment.
destination-port *number*	TCP or UDP destination port field. You cannot specify both the **port** and **destination-port** match conditions in the same term.
destination-port-except	Do not match the TCP/UDP destination port.
destination-prefix-list	Match the IP destination prefixes in a named list.
dscp *number*	Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see *BA Classifier Overview*. You can specify a numeric value from **0** through **63**. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): • RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: **ef** (46). • RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points: **af11** (10), **af12** (12), **af13** (14), **af21** (18), **af22** (20), **af23** (22), **af31** (26), **af32** (28), **af33** (30), **af41** (34), **af42** (36), **af43** (38)
dscp-except *number*	Do not match on the DSCP number. For more information, see the **dscp-except** match condition.

Table 28: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series 3D Universal Edge Routers and EX Series switches Only) *(continued)*

Match Condition	Description
ether-type *value*	Match the 2-octet IEEE 802.3 Length/EtherType field to the specified value or list of values. You can specify decimal or hexadecimal values from 0 through 65535 (0xFFFF). A value from 0 through 1500 (0x05DC) specifies the length of an Ethernet Version 1 frame. A value from 1536 (0x0600) through 65535 specifies the EtherType (nature of the MAC client protocol) of an Ethernet Version 2 frame. In place of the numeric value, you can specify one of the following text synonyms (the hexadecimal values are also listed): **aarp** (0x80F3), **appletalk** (0x809B), **arp** (0x0806), **ipv4** (0x0800), **ipv6** (0x86DD), **mpls-multicast** (0x8848), **mpls-unicast** (0x8847), **oam** (0x8902), **ppp** (0x880B), **pppoe-discovery** (0x8863), **pppoe-session** (0x8864), **sna** (0x80D5).
ether-type-except *value*	Do not match the 2-octet IEEE 802.3 Length/EtherType field to the specified value or list of values. For details about specifying the ***values***, see the **ether-type** match condition.
forwarding class *class*	Forwarding class. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.
forwarding-class-except *class*	Ethernet type field of a Layer 2 packet environment. Specify **assured-forwarding**, **best-effort**, **expedited-forwarding**, or **network-control**.
icmp-code *message-code*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the **ip-protocol icmp**, **ip-protocol icmp6**, or **ip-protocol icmpv6** match condition in the same term. If you configure this match condition, you must also configure the **icmp-type** *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: **ip6-header-bad** (0), **unrecognized-next-header** (1), **unrecognized-option** (2) • time-exceeded: **ttl-eq-zero-during-reassembly** (1), **ttl-eq-zero-during-transit** (0) • destination-unreachable: **address-unreachable** (3), **administratively-prohibited** (1), **no-route-to-destination** (0), **port-unreachable** (4)
icmp-code-except *message-code*	Do not match the ICMP message code field. For details, see the **icmp-code** match condition.

Table 28: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series 3D Universal Edge Routers and EX Series switches Only) *(continued)*

Match Condition	Description
icmp-type *message-type*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the **ip-protocol icmp**, **ip-protocol icmp6**, or **ip-protocol icmpv6** match condition in the same term. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **destination-unreachable** (1), **echo-reply** (129), **echo-request** (128), **membership-query** (130), **membership-report** (131), **membership-termination** (132), **neighbor-advertisement** (136), **neighbor-solicit** (135), **node-information-reply** (140), **node-information-request** (139), **packet-too-big** (2), **parameter-problem** (4), **redirect** (137), **router-advertisement** (134), **router-renumbering** (138), **router-solicit** (133), or **time-exceeded** (3).
icmp-type-except *message-type*	Do not match the ICMP message type field. For details, see the **icmp-type** match condition.
interface *interface-name*	Interface on which the packet was received. You can configure a match condition that matches packets based on the interface on which they were received. NOTE: If you configure this match condition with an interface that does not exist, the term does not match any packet.
interface-group *group-number*	Match the logical interface on which the packet was received to the specified interface group or set of interface groups. For ***group-number***, specify a single value or a range of values from **0** through **255**. To assign a logical interface to an interface group ***group-number***, specify the ***group-number*** at the **[interfaces** *interface-name* **unit** *number* **family** *family* **filter group]** hierarchy level. For more information, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63.
interface-group-except *number*	Do not match the logical interface on which the packet was received to the specified interface group or set of interface groups. For details, see the **interface-group** match condition.
interface-set *interface-set-name*	Match the interface on which the packet was received to the specified interface set. To define an interface set, include the **interface-set** statement at the **[edit firewall]** hierarchy level. For more information, see "Filtering Packets Received on an Interface Set Overview" on page 63.
ip-address *address*	32-bit address that supports the standard syntax for IPv4 addresses.
ip-destination-address *address*	32-bit address that is the final destination node address for the packet.
ip-precedence *ip-precedence-field*	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).
ip-precedence-except *ip-precedence-field*	Do not match on the IP precedence field.

Table 28: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series 3D Universal Edge Routers and EX Series switches Only) *(continued)*

Match Condition	Description
ip-protocol *number*	IP protocol field.
ip-protocol-except	Do not match the IP protocol type.
ip-source-address *address*	IP address of the source node sending the packet.
isid *number*	(Supported with Provider Backbone Bridging [PBB]) Match internet service identifier.
isid-dei *number*	(Supported with PBB) Match the Internet service identifier drop eligibility indicator (DEI) bit.
isid-dei-except *number*	(Supported with PBB) Do not match the Internet service identifier DEI bit.
isid-priority-code-point *number*	(Supported with PBB) Match the Internet service identifier priority code point.
isid-priority-code-point-except *number*	(Supported with PBB) Do not match the Internet service identifier priority code point.
learn-vlan-1p-priority *value*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p learned VLAN priority bits in the provider VLAN tag (the only tag in a single-tag frame with 802.1Q VLAN tags or the outer tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from **0** through **7**. Compare with the **user-vlan-1p-priority** match condition.
learn-vlan-1p-priority-except *value*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p learned VLAN priority bits. For details, see the **learn-vlan-1p-priority** match condition.
learn-vlan-dei *number*	(Supported with bridging) Match user virtual LAN (VLAN) identifier DEI bit.
learn-vlan-dei-except *number*	(Supported with bridging) Do not match user VLAN identifier DEI bit.
learn-vlan-id *number*	VLAN identifier used for MAC learning.
learn-vlan-id-except *number*	Do not match on the VLAN identifier used for MAC learning.

Table 28: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series 3D Universal Edge Routers and EX Series switches Only) *(continued)*

Match Condition	Description
loss-priority *level*	Packet loss priority (PLP) level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers and EX Series switches. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*.
loss-priority-except *level*	Do not match on the packet loss priority level. Specify a single level or multiple levels: **low**, **medium-low**, **medium-high**, or **high**. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see the *BA Classifier Overview*.
port *number*	TCP or UDP source or destination port. You cannot specify both the **port** match condition and either the **destination-port** or **source-port** match conditions in the same term.
source-mac-address *address*	Source MAC address of a Layer 2 packet.
source-port *number*	TCP or UDP source port field. You cannot specify the **port** and **source-port** match conditions in the same term.
source-port-except	Do not match the TCP/UDP source port.
tcp-flags *flags*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • **fin** (0x01) • **syn** (0x02) • **rst** (0x04) • **push** (0x08) • **ack** (0x10) • **urgent** (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. Configuring the **tcp-flags** match condition requires that you configure the **next-header-tcp** match condition.
traffic-type *type*	Traffic type. Specify **broadcast**, **multicast**, **unknown-unicast**, or **known-unicast**.

Table 28: Standard Firewall Filter Match Conditions for Layer 2 Bridging (MX Series 3D Universal Edge Routers and EX Series switches Only) *(continued)*

Match Condition	Description
traffic-type-except *type*	Do not match on the traffic type.
user-vlan-1p-priority *value*	(MX Series routers and EX Series switches only) Match on the IEEE 802.1p user priority bits in the customer VLAN tag (the inner tag in a dual-tag frame with 802.1Q VLAN tags). Specify a single value or multiple values from **0** through **7**. Compare with the **learn-vlan-1p-priority** match condition.
user-vlan-1p-priority-except *value*	(MX Series routers and EX Series switches only) Do not match on the IEEE 802.1p user priority bits. For details, see the **user-vlan-1p-priority** match condition.
user-vlan-id *number*	(MX Series routers and EX Series switches only) Match the first VLAN identifier that is part of the payload.
user-vlan-id-except *number*	(MX Series routers and EX Series switches only) Do not match on the first VLAN identifier that is part of the payload.
vlan-ether-type *value*	VLAN Ethernet type field of a Layer 2 bridging packet.
vlan-ether-type-except *value*	Do not match on the VLAN Ethernet type field of a Layer 2 bridging packet.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384
- Firewall Filter Nonterminating Actions on page 377

Firewall Filter Nonterminating Actions

Firewall filters support different sets of nonterminating actions for each protocol family.

NOTE: You cannot configure the **next term** action with a *terminating* action in the same filter term. However, you can configure the **next term** action with another *nonterminating* action in the same filter term.

Table 29 on page 378 describes the nonterminating actions you can configure for a firewall filter term.

Table 29: Nonterminating Actions for Firewall Filters

Nonterminating Action	Description	Protocol Families
count *counter-name*	Count the packet in the named counter.	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)

Table 29: Nonterminating Actions for Firewall Filters *(continued)*

Nonterminating Action	Description	Protocol Families
dscp *value*	Set the IPv4 Differentiated Services code point (DSCP) bit. You can specify a numerical value from **0** through **63**. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix. The default DSCP value is best effort, that is, **be** or **0**. You can also specify one of the following text synonyms: • **af11**—Assured forwarding class 1, low drop precedence • **af12**—Assured forwarding class 1, medium drop precedence • **af13**—Assured forwarding class 1, high drop precedence • **af21**—Assured forwarding class 2, low drop precedence • **af22**—Assured forwarding class 2, medium drop precedence • **af23**—Assured forwarding class 2, high drop precedence • **af31**—Assured forwarding class 3, low drop precedence • **af32**—Assured forwarding class 3, medium drop precedence • **af33**—Assured forwarding class 3, high drop precedence • **af41**—Assured forwarding class 4, low drop precedence • **af42**—Assured forwarding class 4, medium drop precedence • **af43**—Assured forwarding class 4, high drop precedence • **be**—Best effort • **cs0**—Class selector 0 • **cs1**—Class selector 1 • **cs2**—Class selector 2 • **cs3**—Class selector 3 • **cs4**—Class selector 4 • **cs5**—Class selector 5 • **cs6**—Class selector 6 • **cs7**—Class selector 7 • **ef**—Expedited forwarding NOTE: This action is not supported on PTX Series Packet Transport Routers. NOTE: The actions **dscp 0** and **dscp be** are supported only on T320, T640, T1600, TX Matrix, TX Matrix Plus. and M320 routers and on the 10-Gigabit Ethernet Modular Port Concentrators (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Ethernet Queuing MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers (and EX Series switches). However, these actions are not supported on Enhanced III Flexible PIC Concentrators (FPCs) on M320 routers. NOTE: On T4000 routers, the **dscp 0** action is not supported during the interoperation between a T1600 Enhanced Scaling Type 4 FPC and a T4000 Type 5 FPC.	**family inet**

Table 29: Nonterminating Actions for Firewall Filters *(continued)*

Nonterminating Action	Description	Protocol Families
forwarding-class *class-name*	Classify the packet to the named forwarding class: • *forwarding-class-name* • **assured-forwarding** • **best-effort** • **expedited-forwarding** • **network-control**	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)
ipsec-sa *ipsec-sa*	Use the specified IPsec security association. NOTE: This action is not supported on MX Series routers and EX Series switches, Type 5 FPCs on T4000 routers, and PTX Series Packet Transport Routers.	**family inet**
load-balance *group-name*	Use the specified load-balancing group. NOTE: This action is not supported on MX Series routers, EX Series switches, or PTX Series Packet Transport Routers.	**family inet**
log	Log the packet header information in a buffer within the Packet Forwarding Engine. You can access this information by issuing the **show firewall log** command at the command-line interface (CLI).	• **family inet** • **family inet6**
logical-system *logical-system-name*	Direct packets to a specific logical system.	• **family inet** • **family inet6**
loss-priority (high | medium-high | medium-low | low)	Set the packet loss priority (PLP) level. You cannot also configure the **three-color-policer** nonterminating action for the same firewall filter term. These two nonterminating actions are mutually exclusive. Supported on M120 and M320 routers; M7i and M10i routers with the Enhanced CFEB (CFEB-E); and MX Series routers. For IP traffic on M320, MX Series, and T Series routers with Enhanced II Flexible PIC Concentrators (FPCs), and EX Series switches, you must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can only configure the **high** and **low** levels. This applies to all protocol families. For information about the **tri-color** statement and using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *BA Classifier Overview*.	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)
next-hop-group *group-name*	Use the specified next-hop group.	• **family any** • **family inet**

Table 29: Nonterminating Actions for Firewall Filters *(continued)*

Nonterminating Action	Description	Protocol Families
next-interface *interface-name*	(MX Series routers and EX Series switches) Direct packets to the specified outgoing interface.	• **family inet** • **family inet6**
next-ip *ip-address*	(MX Series routers and EX Series switches) Direct packets to the specified destination IPv4 address.	**family inet**
next-ip6 *ipv6-address*	(MX Series routers and EX Series switches) Direct packets to the specified destination IPv6 address.	**family inet6**
packet-mode	Updates a bit field in the packet key buffer, which specifies traffic that will bypass flow-based forwarding. Packets with the **packet-mode** action modifier follow the packet-based forwarding path and bypass flow-based forwarding completely. For more information about selective stateless packet-based services, see the *Junos OS Security Configuration Guide*.	**family any**
policer *policer-name*	Name of policer to use to rate-limit traffic.	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)
port-mirror *instance-name*	Port-mirror the packet based on the specified family. Supported on M120 routers, M320 routers configured with Enhanced III FPCs, MX Series routers, and PTX Series Packet Transport Routers only.	• **family any** • **family inet** • **family inet6** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)
port-mirror-instance *instance-name*	Port mirror a packet for an instance. This action is only supported on the MX series routers.	• **family any** • **family inet** • **family inet6** • **family vpls** • **family ccc** • **family bridge**

Table 29: Nonterminating Actions for Firewall Filters *(continued)*

Nonterminating Action	Description	Protocol Families
prefix-action *action-name*	Count or police packets based on the specified action name. NOTE: This action is not supported on PTX Series Packet Transport Routers.	**family inet**
routing-instance *routing-instance-name*	Direct packets to the specified routing instance.	• **family inet** • **family inet6**
sample	Sample the packet. NOTE: The Junos OS does not sample packets originating from the router or switch. If you configure a 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.	• **family inet** • **family inet6** • **family mpls**
service-accounting	Count the packet for service accounting. The count is applied to a specific named counter (**__junos-dyn-service-counter**) that RADIUS can obtain. NOTE: This action is not supported on T4000 Type 5 FPCs and PTX Series Packet Transport Routers.	• **family inet** • **family inet6**
service-filter-hit	(Only if the **service-filter-hit** flag is marked by a previous filter in the current type of chained filters) Direct the packet to the next type of filters. Indicate to subsequent filters in the chain that the packet was already processed. This action, coupled with the **service-filter-hit** match condition in receiving filters, helps to streamline filter processing. NOTE: This action is not supported on T4000 Type 5 FPCs and PTX Series Packet Transport Routers.	• **family inet** • **family inet6**
syslog	Log the packet to the system log file.	• **family inet** • **family inet6**
three-color-policer (single-rate | two-rate) *policer-name*	Police the packet using the specified single-rate or two-rate three-color-policer. You cannot also configure the **loss-priority** action for the same firewall filter term. These two actions are mutually exclusive.	• **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)

Table 29: Nonterminating Actions for Firewall Filters *(continued)*

Nonterminating Action	Description	Protocol Families
traffic-class *value*	Specify the traffic-class code point. You can specify a numerical value from **0** through **63**. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix. The default traffic-class value is best effort, that is, **be** or **0**. In place of the numeric value, you can specify one of the following text synonyms: • **af11**—Assured forwarding class 1, low drop precedence • **af12**—Assured forwarding class 1, medium drop precedence • **af13**—Assured forwarding class 1, high drop precedence • **af21**—Assured forwarding class 2, low drop precedence • **af22**—Assured forwarding class 2, medium drop precedence • **af23**—Assured forwarding class 2, high drop precedence • **af31**—Assured forwarding class 3, low drop precedence • **af32**—Assured forwarding class 3, medium drop precedence • **af33**—Assured forwarding class 3, high drop precedence • **af41**—Assured forwarding class 4, low drop precedence • **af42**—Assured forwarding class 4, medium drop precedence • **af43**—Assured forwarding class 4, high drop precedence • **be**—Best effort • **cs0**—Class selector 0 • **cs1**—Class selector 1 • **cs2**—Class selector 2 • **cs3**—Class selector 3 • **cs4**—Class selector 4 • **cs5**—Class selector 5 • **cs6**—Class selector 6 • **cs7**—Class selector 7 • **ef**—Expedited forwarding NOTE: The actions **traffic-class 0**and **traffic-class be** are supported only on T Series and M320 routers and on the 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Ethernet Queuing MPC, and 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers (and EX Series switches). However, these actions are not supported on Enhanced III Flexible PIC Concentrators (FPCs) on M320 routers.	**family inet6**

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Terminating Actions on page 384

Firewall Filter Terminating Actions

Firewall filters support a set of terminating actions for each protocol family. A filter-terminating action halts all evaluation of a firewall filter for a specific packet. The router performs the specified action, and no additional terms are examined.

NOTE: You cannot configure the **next term** action with a *terminating* action in the same filter term. However, you can configure the **next term** action with another *nonterminating* action in the same filter term.

Table 30 on page 384 describes the terminating actions you can specify in a firewall filter term.

Table 30: Terminating Actions for Firewall Filters

Terminating Action	Description	Protocols
accept	Accept the packet.	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)

Table 30: Terminating Actions for Firewall Filters *(continued)*

Terminating Action	Description	Protocols
decapsulate gre [routing-instance *instance-name*]	At a customer-facing interface on an MX Series router installed at the provider edge (PE) of an IPv4 transport network, enable decapsulation of generic routing encapsulation (GRE) packets transported through a filter-based GRE tunnel. You can configure a filter term that pairs this action with a match condition that includes a packet header match for the GRE protocol. For an IPv4 filter, include the **protocol gre** (or **protocol 47**) match condition. Attach the filter to the input of an Ethernet logical interface or aggregated Ethernet interface on a Modular Interface Card (MIC) or Modular Port Concentrator (MPC) in the router. If you commit a configuration that attaches a de-encapsulating filter to an interface that does not support filter-based GRE tunneling, the system writes a syslog warning message that the interface does not support the filter. When the interface receives a matched packet, processes that run on the Packet Forwarding Engine perform the following operations: • Remove the outer GRE header. • Forward the inner payload packet to its original destination by performing destination lookup. By default, the Packet Forwarding Engine uses the default routing instance to forward payload packets to the destination network. If the payload is MPLS, the Packet Forwarding Engine performs route lookup on the MPLS path routing table using the route label in the MPLS header. If you specify the **decapsulate** action with an optional routing instance name, the Packet Forwarding Engine performs route lookup on the routing-instance, and the instance must be configured. For more information, see "Understanding Filter-Based Tunneling Across IPv4 Networks" on page 65 and "Components of Filter-Based Tunneling Across IPv4 Networks" on page 69.	• **family inet**
discard	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are available for logging and sampling.	• **family any** • **family inet** • **family inet6** • **family mpls** • **family vpls** • **family ccc** • **family bridge** • **family ethernet-switching** (for EX Series switches only)

Table 30: Terminating Actions for Firewall Filters *(continued)*

Terminating Action	Description	Protocols
encapsulate *template-name*	At a customer-facing interface on an MX Series router installed at the provider edge (PE) of an IPv4 transport network, enable filter-based generic routing encapsulation (GRE) tunneling using the specified tunnel template. You can configure a filter term that pairs this action with the appropriate match conditions, and then attach the filter to the input of an Ethernet logical interface or aggregated Ethernet interface on a Modular Interface Card (MIC) or Modular Port Concentrator (MPC) in the router. If you commit a configuration that attaches an encapsulating filter to an interface that does not support filter-based GRE tunneling, the system writes a syslog warning message that the interface does not support the filter. When the interface receives a matched packet, processes that run on the Packet Forwarding Engine use information in the specified tunnel template to perform the following operations: 1. Attach a GRE header (with or without a tunnel key value, as specified in the tunnel template. 2. Attach a header for the IPv4 transport protocol. 3. Forward the resulting GRE packet from the tunnel source interface to the tunnel destination (the remote PE router). The specified tunnel template must be configured using the tunnel-end-point statement under the **[edit firewall]** or **[edit logical-systems** *logical-system-name* **firewall]** statement hierarchy. For more information, see "Understanding Filter-Based Tunneling Across IPv4 Networks" on page 65.	• **family inet** • **family inet6** • **family any** • **family mpls**
logical-system *logical-system-name*	Direct the packet to the specified logical system. NOTE: This action is not supported on PTX Series Packet Transport Routers.	• **family inet** • **family inet6**
reject *message-type*	Reject the packet and return an ICMPv4 or ICMPv6 message: • If no ***message-type*** is specified, a **destination unreachable** message is returned by default. • If **tcp-reset** is specified as the ***message-type***, **tcp-reset** is returned only if the packet is a TCP packet. Otherwise, the **administratively-prohibited** message, which has a value of 13, is returned. • If any other ***message-type*** is specified, that message is returned. NOTE: Rejected packets can be sampled or logged if you configure the **sample** or **syslog** action. The ***message-type*** can be one of the following values: **address-unreachable**, **administratively-prohibited**, **bad-host-tos**, **bad-network-tos**, **beyond-scope**, **fragmentation-needed**, **host-prohibited**, **host-unknown**, **host-unreachable**, **network-prohibited**, **network-unknown**, **network-unreachable**, **no-route**, **port-unreachable**, **precedence-cutoff**, **precedence-violation**, **protocol-unreachable**, **source-host-isolated**, **source-route-failed**, or **tcp-reset**.	• **family inet** • **family inet6**
routing-instance *instance-name*	Direct the packet to the specified routing instance. NOTE: This action is not supported on PTX Series Packet Transport Routers.	• **family inet** • **family inet6**

Table 30: Terminating Actions for Firewall Filters *(continued)*

Terminating Action	Description	Protocols
topology *topology-name*	Direct the packet to the specified topology. NOTE: This action is not supported on PTX Series Packet Transport Routers. Each routing instance (master or virtual-router) supports one default topology to which all forwarding classes are forwarded. For multitopology routing, you can configure a firewall filter on the ingress interface to match a specific forwarding class, such as expedited forwarding, with a specific topology. The traffic that matches the specified forwarding class is then added to the routing table for that topology.	• **family inet** • **family inet6**

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Firewall Filter Nonterminating Actions on page 377

CHAPTER 28

Firewall Filter Match Conditions and Actions for ACX Series Routers

Standard Firewall Filter Match Conditions and Actions on ACX Series Routers Overview

On ACX Series Universal Access Routers, you can configure firewall filters to filter packets and to perform an action on packets that match the filter. The match conditions specified to filter the packets are specific to the type of traffic being filtered.

NOTE: On ACX Series routers, the filter for the exiting traffic (egress filter) can be applied only for interface-specific instances of the firewall filter.

Table 31 on page 389 describes the types of traffic for which you can configure standard stateless firewall filters.

Table 31: Standard Firewall Filter Match Conditions by Protocol Family for ACX Series Routers

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
Protocol-independent	**[edit firewall family any filter *filter-name* term *term-name*]** No match conditions are supported for this traffic type on ACX Series routers.

Table 31: Standard Firewall Filter Match Conditions by Protocol Family for ACX Series Routers *(continued)*

Traffic Type	Hierarchy Level at Which Match Conditions Are Specified
IPv4	**[edit firewall family inet filter *filter-name* term *term-name*** For the complete list of match conditions, see "Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers" on page 391.
MPLS	**[edit firewall family mpls filter *filter-name* term *term-name*]** For the complete list of match conditions, see "Standard Firewall Filter Match Conditions for MPLS Traffic on ACX Series Routers" on page 394.
Layer 2 CCC	**[edit firewall family ccc filter *filter-name* term *term-name*]** No match conditions are supported for this traffic type on ACX Series routers.

Under the **then** statement for a standard stateless firewall filter term, you can specify the actions to be taken on a packet that matches the term.

Table 32 on page 390 summarizes the types of actions you can specify in a standard stateless firewall filter term.

Table 32: Standard Firewall Filter Action Categories for ACX Series Routers

Type of Action	Description	Comment
Terminating	Halts all evaluation of a firewall filter for a specific packet. The router performs the specified action, and no additional terms are used to examine the packet. You can specify only one *terminating action* in a standard firewall filter. You can, however, specify one terminating action with one or more *nonterminating actions* in a single term. For example, within a term, you can specify **accept** with **count** and **syslog**.	See "Standard Firewall Filter Terminating Actions on ACX Series Routers" on page 397.
Nonterminating	Performs other functions on a packet (such as incriminating a counter, logging information about the packet header, sampling the packet data, or sending information to a remote host using the system log functionality), but any additional terms are used to examine the packet.	See "Standard Firewall Filter Nonterminating Actions on ACX Series Routers" on page 394.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Interface-Specific Firewall Filter Instances Overview on page 61

Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers

On ACX Series routers, you can configure a standard stateless firewall filter with match conditions for IP version 4 (IPv4) traffic (**family inet**). Table 33 on page 391 describes the match conditions you can configure at the **[edit firewall family inet filter *filter-name* term *term-name* from]** hierarchy level.

Table 33: Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers

Match Condition	Description
destination-address *address*	Match the IPv4 destination address field. NOTE: On ACX Series routers, you can specify only one destination address. A list of IPv4 destination addresses is not supported.
destination-port *number*	Match the UDP or TCP destination port field. If you configure this match condition, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. NOTE: On ACX Series routers, you can specify only one destination port number. A list of port numbers is not supported. 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), **ldp** (646), **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** (49), **tacacs-ds** (65), **talk** (517), **telnet** (23), **tftp** (69), **timed** (525), **who** (513), or **xdmcp** (177).

dscp *number*	Match the Differentiated Services code point (DSCP). The DiffServ protocol uses the type-of-service (ToS) byte in the IP header. The most significant 6 bits of this byte form the DSCP. For more information, see *BA Classifier Overview*.
	
	You can specify a numeric value from 0 through 63. To specify the value in hexadecimal form, include 0x as a prefix. To specify the value in binary form, include b as a prefix.
	
	In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed):
	
	• RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, defines one code point: **ef** (46).
	• RFC 2597, *Assured Forwarding PHB Group*, defines 4 classes, with 3 drop precedences in each class, for a total of 12 code points:
	  • **af11** (10), **af12** (12), **af13** (14)
	  • **af21** (18), **af22** (20), **af23** (22)
	  • **af31** (26), **af32** (28), **af33** (30)
	  • **af41** (34), **af42** (36), **af43** (38)

Table 33: Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(continued)*

Match Condition	Description
fragment-flags *number*	(Ingress only) Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): **dont-fragment** (0x4), **more-fragments** (0x2), or **reserved** (0x8).
icmp-code *number*	Match the ICMP message code field. If you configure this match condition, we recommend that you also configure the **protocol icmp** match condition in the same term. If you configure this match condition, you must also configure the **icmp-type** *message-type* match condition in the same term. An ICMP message code provides more specific information than an ICMP message type, but the meaning of an ICMP message code is dependent on the associated ICMP message type. 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: • parameter-problem: **ip-header-bad** (0), **required-option-missing** (1) • redirect: **redirect-for-host** (1), **redirect-for-network** (0), **redirect-for-tos-and-host** (3), **redirect-for-tos-and-net** (2) • time-exceeded: **ttl-eq-zero-during-reassembly** (1), **ttl-eq-zero-during-transit** (0) • 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)
icmp-type *number*	Match the ICMP message type field. If you configure this match condition, we recommend that you also configure the **protocol icmp** match condition in the same term. 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).
ip-options *values*	Match the 8-bit IP option field, if present, to the specified value. ACX Series routers support only the **ip-options_any** match condition, which ensures that the packets are sent to the Packet Forwarding Engine for processing. NOTE: On ACX Series routers, you can specify only one IP option value. Configuring multiple values is not supported.

Table 33: Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(continued)*

Match Condition	Description
precedence *ip-precedence-field*	Match the 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). You can specify precedence in hexadecimal, binary, or decimal form.
protocol *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **ospf** (89), **pim** (103), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).
source-address *address*	Match the IPv4 address of the source node sending the packet.
source-port *number*	Match the UDP or TCP source port field. If you configure this match condition for IPv4 traffic, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the **destination-port** *number* match condition.
tcp-flags *value*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • **fin** (0x01) • **syn** (0x02) • **rst** (0x04) • **push** (0x08) • **ack** (0x10) • **urgent** (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the **tcp-initial** match conditions. If you configure this match condition, we recommend that you also configure the **protocol tcp** match statement in the same term to specify that the TCP protocol is being used on the port.
tcp-initial	Match the initial packet of a TCP connection. This is an alias for **tcp-flags "(!ack & syn)"**. This condition does not implicitly check that the protocol is TCP. If you configure this match condition, we recommend that you also configure the **protocol tcp** match condition in the same term.

Table 33: Standard Firewall Filter Match Conditions for IPv4 Traffic on ACX Series Routers *(continued)*

Match Condition	Description
ttl ***number***	Match the IPv4 time-to-live number. Specify a TTL value or a range of TTL values. For ***number***, you can specify one or more values from 2 through 255.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Standard Firewall Filter Match Conditions and Actions on ACX Series Routers Overview on page 389
- Standard Firewall Filter Terminating Actions on ACX Series Routers on page 397
- Standard Firewall Filter Nonterminating Actions on ACX Series Routers on page 394

Standard Firewall Filter Match Conditions for MPLS Traffic on ACX Series Routers

On ACX Series routers, you can configure a standard stateless firewall filter with match conditions for MPLS traffic (**family mpls**).

NOTE: The **input-list** ***filter-names*** and **output-list** ***filter-names*** statements for firewall filters for the **mpls** protocol family are supported on all interfaces with the exception of management interfaces and internal Ethernet interfaces (**fxp** or **em0**), loopback interfaces (**lo0**), and USB modem interfaces (**umd**).

Table 34 on page 394 describes the match conditions you can configure at the **[edit firewall family mpls filter** ***filter-name*** **term** ***term-name*** **from]** hierarchy level.

Table 34: Standard Firewall Filter Match Conditions for MPLS Traffic on ACX Series Routers

Match Condition	Description
exp ***number***	Experimental (EXP) bit number or range of bit numbers in the MPLS header. For ***number***, you can specify one or more values from 0 through 7 in decimal, binary, or hexadecimal format.

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Standard Firewall Filter Match Conditions and Actions on ACX Series Routers Overview on page 389
- Standard Firewall Filter Terminating Actions on ACX Series Routers on page 397
- Standard Firewall Filter Nonterminating Actions on ACX Series Routers on page 394

Standard Firewall Filter Nonterminating Actions on ACX Series Routers

Standard stateless firewall filters support different sets of nonterminating actions for each protocol family.

NOTE: ACX Series routers do not support the **next term** action.

Table 35 on page 395 describes the nonterminating actions you can configure for a standard firewall filter term.

Table 35: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers

Nonterminating Action	Description	Protocol Families
count *counter-name*	Count the packet in the named counter.	• **family any** • **family inet** • **family mpls** • **family ccc**
forwarding-class *class-name*	Classify the packet based on the specified forwarding class: • **assured-forwarding** • **best-effort** • **expedited-forwarding** • **network-control** NOTE: This action is supported on ingress only.	• **family inet** • **family any** • **family mpls** • **family ccc**
log	Log the packet header information in a buffer within the Packet Forwarding Engine. You can access this information by issuing the **show firewall log** command at the command-line interface (CLI). NOTE: This action is supported on ingress only.	**family inet**

Table 35: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers *(continued)*

Nonterminating Action	Description	Protocol Families
loss-priority (high | medium-high | low)	Set the packet loss priority (PLP) level. You cannot also configure the **three-color-policer** nonterminating action for the same firewall filter term. These two nonterminating actions are mutually exclusive. You must include the **tri-color** statement at the **[edit class-of-service]** hierarchy level to commit a PLP configuration with any of the four levels specified. If the **tri-color** statement is not enabled, you can configure only the **high** and **low** levels. This applies to all protocol families. For information about the **tri-color** statement, see *Configuring Tricolor Marking*. For information about using behavior aggregate (BA) classifiers to set the PLP level of incoming packets, see *Overview of Forwarding Classes*. NOTE: This action is supported on ingress only.	• **family any** • **family inet** • **family mpls** • **family ccc**
policer *policer-name*	Name of policer to use to rate-limit traffic.	• **family any** • **family inet** • **family mpls** • **family ccc**
port-mirror	Port-mirror the packet based on the specified family. NOTE: This action is supported on ingress only.	**family inet**
syslog	Log the packet to the system log file. NOTE: This action is supported on ingress only.	**family inet**

Table 35: Nonterminating Actions for Standard Firewall Filters on ACX Series Routers *(continued)*

Nonterminating Action	Description	Protocol Families
three-color-policer (single-rate | two-rate) *policer-name*	Police the packet using the specified single-rate or two-rate three-color policer. You cannot also configure the **loss-priority** action for the same firewall filter term. These two actions are mutually exclusive.	• **family any** • **family inet** • **family mpls** • **family ccc**

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Standard Firewall Filter Match Conditions and Actions on ACX Series Routers Overview on page 389
- Standard Firewall Filter Terminating Actions on ACX Series Routers on page 397

Standard Firewall Filter Terminating Actions on ACX Series Routers

Standard stateless firewall filters support different sets of terminating actions for each protocol family.

NOTE: ACX Series routers do not support the **next term** action.

Table 36 on page 397 describes the terminating actions you can specify in a standard firewall filter term.

Table 36: Terminating Actions for Standard Firewall Filters on ACX Series Routers

Terminating Action	Description	Protocols
accept	Accept the packet.	• **family any** • **family inet** • **family mpls** • **family ccc**
discard	Discard a packet silently, without sending an Internet Control Message Protocol (ICMP) message. Discarded packets are available for logging and sampling.	• **family any** • **family inet** • **family mpls** • **family ccc**

Table 36: Terminating Actions for Standard Firewall Filters on ACX Series Routers *(continued)*

Terminating Action	Description	Protocols
reject ***message-type***	Reject the packet and return an ICMPv4 or ICMPv6 message: • If no message type is specified, a **destination-unreachable** message is returned by default. • If **tcp-reset** is specified as the message type, **tcp-reset** is returned only if the packet is a TCP packet. Otherwise, the **administratively-prohibited** message, which has a value of 13, is returned. • If any other message type is specified, that message is returned. NOTE: • Rejected packets can be sampled or logged if you configure the **sample** or **syslog** action. • This action is supported on ingress only. The ***message-type*** option can have one of the following values: **address-unreachable**, **administratively-prohibited**, **bad-host-tos**, **bad-network-tos**, **beyond-scope**, **fragmentation-needed**, **host-prohibited**, **host-unknown**, **host-unreachable**, **network-prohibited**, **network-unknown**, **network-unreachable**, **no-route**, **port-unreachable**, **precedence-cutoff**, **precedence-violation**, **protocol-unreachable**, **source-host-isolated**, **source-route-failed**, or **tcp-reset**.	**family inet**
routing-instance ***routing-instance-name***	Direct the packet to the specified routing instance.	• **family inet**

Related Documentation

- Guidelines for Configuring Firewall Filters on page 21
- Standard Firewall Filter Match Conditions and Actions on ACX Series Routers Overview on page 389
- Standard Firewall Filter Nonterminating Actions on ACX Series Routers on page 394

CHAPTER 29

Service Filter Match Conditions and Actions

Service Filter Match Conditions for IPv4 or IPv6 Traffic

Service filters support only a subset of the stateless firewall filter match conditions for IPv4 and IPv6 traffic. Table 37 on page 399 describess the service filter match conditions.

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic

Match Condition	Description	Protocol Families
address *address*	Match the IP source or destination address field.	• **family inet** • **family inet6**
address *address* **except**	Do not match the IP source or destination address field.	• **family inet** • **family inet6**
ah-spi *spi-value*	(M Series routers, except M120 and M320) Match on the IPsec authentication header (AH) security parameter index (SPI) value.	• **family inet**
ah-spi-except *spi-value*	(M Series routers, except M120 and M320) Do not match on the IPsec AH SPI value.	• **family inet**
destination-address *address*	Match the IP destination address field. You cannot specify both the **address** and **destination-address** match conditions in the same term.	• **family inet** • **family inet6**
destination-address *address* **except**	Do not match the IP destination address field. You cannot specify both the **address** and **destination-address** match conditions in the same term.	• **family inet** • **family inet6**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families
destination-port *number*	Match the UDP or TCP destination port field. You cannot specify both the **port** and **destination-port** match conditions in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the **next-header udp** or **next-header tcp** match condition in the same term to specify which protocol is being used on the port. 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), **ldp** (646), **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** (49), **tacacs-ds** (65), **talk** (517), **telnet** (23), **tftp** (69), **timed** (525), **who** (513), or **xdmcp** (177).	• **family inet** • **family inet6**
destination-port-except *number*	Do not match the UDP or TCP destination port field. For details, see the **destination-port** match description.	• **family inet** • **family inet6**
destination-prefix-list *name*	Match the list of destination prefixes. The prefix list is defined at the **[edit policy-options prefix-list** *prefix-list-name*] hierarchy level.	• **family inet** • **family inet6**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families
esp-spi *value*	Match the IPsec encapsulating security payload (ESP) SPI value. Specify a single value or a range of values. You can specify a *value* in hexadecimal, binary, or decimal form. To specify the value in hexadecimal form, include **0x** as a prefix. To specify the value in binary form, include **b** as a prefix.	• **family inet** • **family inet6**
esp-spi-except *value*	Do not match the IPsec ESP SPI value or range of values. For details, see the **esp-spi** match condition.	• **family inet** • **family inet6**
first-fragment	Match if the packet is the first fragment of a fragmented packet. Do not match if the packet is a trailing fragment of a fragmented packet. The first fragment of a fragmented packet has a fragment offset value of **0**. This match condition is an alias for the bit-field match condition **fragment-offset 0** match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions: **first-fragment** and **is-fragment**.	• **family inet**
fragment-flags *number*	(Ingress only) Match the three-bit IP fragmentation flags field in the IP header. In place of the numeric field value, you can specify one of the following keywords (the field values are also listed): **dont-fragment** (0x4), **more-fragments** (0x2), or **reserved** (0x8).	• **family inet**
fragment-offset *number*	Match the 13-bit fragment offset field in the IP header. The value is the offset, in 8-byte units, in the overall datagram message to the data fragment. Specify a numeric value, a range of values, or a set of values. An offset value of **0** indicates the first fragment of a fragmented packet. The **first-fragment** match condition is an alias for the **fragment-offset 0** match condition. To match both first and trailing fragments, you can use two terms that specify different match conditions (**first-fragment** and **is-fragment**).	• **family inet**
fragment-offset-except *number*	Do not match the 13-bit fragment offset field.	• **family inet**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families
interface-group ***group-number***	Match the interface group (set of one or more logical interfaces) on which the packet was received. For ***group-number***, specify a value from **0** through **255**. For information about configuring interface groups, see "Filtering Packets Received on a Set of Interface Groups Overview" on page 63.	• **family inet** • **family inet6**
interface-group-except ***group-number***	Do not match the interface group on which the packet was received. for details, see the **interface-group** match condition.	• **family inet** • **family inet6**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families	
ip-options *values*	Match the 8-bit IP option field, if present, to the specified value or list of values. In place of a numeric value, you can specify one of the following text synonyms (the option values are also listed): **loose-source-route** (131), **record-route** (7), **router-alert** (148), **security** (130), **stream-id** (136), **strict-source-route** (137), or **timestamp** (68). To match *any* value for the IP option, use the text synonym **any**. To match on *multiple* values, specify the list of values within square brackets ('**[**' and '**]**'). To match a *range* of values, use the value specification **[** *value1-value2* **]**. For example, the match condition **ip-options [0-147]** matches on an IP options field that contains the **loose-source-route**, **record-route**, or **security** values, or any other value from 0 through 147. However, this match condition does not match on an IP options field that contains only the **router-alert** value (148). For most interfaces, a filter term that specifies an **ip-option** match on one or more *specific* IP option values (a value other than **any**) causes packets to be sent to the Routing Engine so that the kernel can parse the IP option field in the packet header. • For a firewall filter term that specifies an **ip-option** match on one or more specific IP option values, you cannot specify the **count**, **log**, or **syslog** nonterminating actions *unless* you also specify the **discard** terminating action in the same term. This behavior prevents double-counting of packets for a filter applied to a transit interface on the router (or switch). • Packets processed on the kernel might be dropped in case of a system bottleneck. To ensure that matched packets are instead sent to the Packet Forwarding Engine (where packet processing is implemented in hardware), use the **ip-options any** match condition. The 10-Gigabit Ethernet Modular Port Concentrator (MPC), 60-Gigabit Ethernet MPC, 60-Gigabit Queuing Ethernet MPC, 60-Gigabit Ethernet Enhanced Queuing MPC on MX Series routers and EX Series switches are capable of parsing the IP option field of the IPv4 packet header. This capability is supported on EX Series switches also. For interfaces configured on those MPCs, *all* packets that are matched using the **ip-options** match condition are sent to the Packet Forwarding Engine for processing.	**family inet**	• **family inet**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families
ip-options-except ***values***	Do not match the IP option field to the specified value or list of values. For details about specifying the ***values***, see the **ip-options** match condition.	• **family inet**
is-fragment	Match if the packet is a trailing fragment of a fragmented packet. Do not match the first fragment of a fragmented packet. This match condition is an alias for the bit-field match condition **fragment-offset 0 except** bits. NOTE: To match both first and trailing fragments, you can use two terms that specify different match conditions (**first-fragment** and **is-fragment**).	• **family inet**
port ***number***	Match the UDP or TCP source or destination port field. If you configure this match condition, you cannot configure the **destination-port** match condition or the **source-port** match condition in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the **protocol udp** or **protoco tcp** match statement in the same term to specify which protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the **next-header udp** or **next-header tcp** match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed under **destination-port**.	• **family inet** • **family inet6**
port-except ***number***	Do not match the UDP or TCP source or destination port field. For details, see the **port** match condition.	• **family inet** • **family inet6**
prefix-list ***prefix-list-name***	Match the prefixes of the source or destination address fields to the prefixes in the specified list. The prefix list is defined at the **[edit policy-options prefix-list** ***prefix-list-name*****]** hierarchy level.	• **family inet** • **family inet6**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families
protocol *number*	Match the IP protocol type field. In place of the numeric value, you can specify one of the following text synonyms (the field values are also listed): **ah** (51), **dstopts** (60), **egp** (8), **esp** (50), **fragment** (44), **gre** (47), **hop-by-hop** (0), **icmp** (1), **icmp6** (58), **icmpv6** (58), **igmp** (2), **ipip** (4), **ipv6** (41), **ospf** (89), **pim** (103), **rsvp** (46), **sctp** (132), **tcp** (6), **udp** (17), or **vrrp** (112).	• **family inet**
protocol-except *number*	Do not match the IP protocol type field. For details, see the **protocol** match condition.	• **family inet**
source-address *address*	Match the IP source address. You cannot specify both the **address** and **source-address** match conditions in the same term.	• **family inet** • **family inet6**
source-address *address* **except**	Do not match the IP source address. You cannot specify both the **address** and **source-address** match conditions in the same term.	• **family inet** • **family inet6**
source-port *number*	Match the UDP or TCP source port field. You cannot specify the **port** and **source-port** match conditions in the same term. If you configure this match condition for IPv4 traffic, we recommend that you also configure the **protocol udp** or **protocol tcp** match statement in the same term to specify which protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the **next-header udp** or **next-header tcp** match condition in the same term to specify which protocol is being used on the port. In place of the numeric value, you can specify one of the text synonyms listed with the **destination-port** *number* match condition.	• **family inet** • **family inet6**
source-port-except *number*	Do not match the UDP or TCP source port field. For details, see the **source-port** match condition.	• **family inet** • **family inet6**
source-prefix-list *name*	Match source prefixes in the specified list. Specify the name of a prefix list defined at the **[edit policy-options prefix-list** *prefix-list-name*] hierarchy level.	• **family inet** • **family inet6**

Table 37: Service Filter Match Conditions for IPv4 or IPv6 Traffic *(continued)*

Match Condition	Description	Protocol Families
tcp-flags *value*	Match one or more of the low-order 6 bits in the 8-bit TCP flags field in the TCP header. To specify individual bit fields, you can specify the following text synonyms or hexadecimal values: • **fin** (0x01) • **syn** (0x02) • **rst** (0x04) • **push** (0x08) • **ack** (0x10) • **urgent** (0x20) In a TCP session, the SYN flag is set only in the initial packet sent, while the ACK flag is set in all packets sent after the initial packet. You can string together multiple flags using the bit-field logical operators. For combined bit-field match conditions, see the **tcp-established** and **tcp-initial** match conditions. If you configure this match condition for IPv4 traffic, we recommend that you also configure the **protocol tcp** match statement in the same term to specify that the TCP protocol is being used on the port. If you configure this match condition for IPv6 traffic, we recommend that you also configure the **next-header tcp** match condition in the same term to specify that the TCP protocol is being used on the port.	• **family inet** • **family inet6**

NOTE: If you specify an IPv6 address in a match condition (the **address**, **destination-address**, or **source-address** match conditions), use the syntax for text representations described in RFC 2373, *IP Version 6 Addressing Architecture*. For more information about IPv6 addresses, see *IPv6 Overview*and *Supported IPv6 Standards*.

Related Documentation

- Service Filter Overview on page 79
- Guidelines for Configuring Service Filters on page 82
- Example: Configuring and Applying Service Filters on page 257
- Service Filter Terminating Actions on page 407
- Service Filter Nonterminating Actions on page 407

Service Filter Nonterminating Actions

Service filters support different sets of terminating actions for each protocol family.

NOTE: Service filters do not support the next term action.

Table 38 on page 407 describes the nonterminating actions you can configure in a service filter term.

Table 38: Nonterminating Actions for Service Filters

Nonterminating Action	Description	Protocol Families
count *counter-name*	Count the packet in the named counter.	• **inet** • **inet6**
log	Log the packet header information in a buffer within the Packet Forwarding Engine. You can access this information by issuing the **show firewall log** command at the command-line interface (CLI).	• **inet** • **inet6**
port-mirror	Port-mirror the packet based on the specified family. Supported on M120 routers, M320 routers configured with Enhanced III FPCs, MX Series routers, and EX Series switches only.	• **inet** • **inet6**
sample	Sample the packet.	• **inet** • **inet6**

Related Documentation

- Service Filter Overview on page 79
- Guidelines for Configuring Service Filters on page 82
- Example: Configuring and Applying Service Filters on page 257
- Service Filter Match Conditions for IPv4 or IPv6 Traffic on page 399
- Service Filter Terminating Actions on page 407

Service Filter Terminating Actions

Service filters support different sets of terminating actions than standard stateless firewall filters or simple filters.

NOTE: Service filters do not support the next term action.

Table 39 on page 408 describes the terminating actions you can configure in a service filter term.

Table 39: Terminating Actions for Service Filters

Terminating Action	Description	Protocol Families
service	Direct the packet to service processing.	• **inet** • **inet6**
skip	Let the packet bypass service processing.	• **inet** • **inet6**

Related Documentation

- Service Filter Overview on page 79
- Guidelines for Configuring Service Filters on page 82
- Example: Configuring and Applying Service Filters on page 257
- Service Filter Match Conditions for IPv4 or IPv6 Traffic on page 399
- Service Filter Nonterminating Actions on page 407

CHAPTER 30

Reference Information for Firewall Filters in Logical Systems

Unsupported Firewall Filter Statements for Logical Systems

Table 40 on page 409 shows statements that are supported at the **[edit firewall]** hierarchy level but not at the **[edit logical-systems *logical-system-name* firewall]** hierarchy level.

Table 40: Unsupported Firewall Statements for Logical Systems

Statement	Example	Description

accounting-profile	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  family inet {
	  filter myfilter {
	  accounting-profile fw-profile;
	  ...
	  term accept-all {
	  then {
	  count counter1;
	  accept;
	  }
	  }
	  }
	  }
	  }
	  }
	}
		In this example, the **accounting-profile** statement is not allowed because the accounting profile **fw-profile** is configured under the **[edit accounting-options]** hierarchy.

Table 40: Unsupported Firewall Statements for Logical Systems *(continued)*

Statement	Example	Description

hierarchical-policer	[edit]
	logical-systems {
	  lr1 {
	  firewall {
	  hierarchical-policer {
	  ...
	  }
	  }
	  }
	}
		In this example, the **hierarchical policer** statement requires a class-of-service configuration, which is not supported under logical systems.

load-balance-group	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  load-balance-group lb-group {
	  next-hop-group nh-group;
	  }
	  }
	  }
	}
		This configuration is not allowed because the **next-hop-group nh-group** statement must be configured at the **[edit forwarding-options next-hop-group]** hierarchy level—outside the **[edit logical-systems *logical-system-name* firewall]** hierarchy.
		
		Currently, the **forwarding-options dhcp-relay** statement is the only forwarding option supported for logical systems.

virtual-channel	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  family inet {
	  filter foo {
	  term one {
	  from {
	  source-address 10.1.0.0/16;
	  }
	  then {
	  virtual-channel sammy;
	  }
	  }
	  }
	  }
	  }
	  }
	}
		This configuration is not allowed because the virtual channel **sammy** refers to an object defined at the **[edit class-of-service]** hierarchy level, and class of service is not supported for logical systems.
		
		NOTE:
		
		The **virtual-channel** statement is supported for J Series devices only, provided the firewall filter is configured outside of a logical-system.

Related Documentation

- Firewall Filters in Logical Systems Overview on page 31
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32
- Unsupported Actions for Firewall Filters in Logical Systems on page 411
- *Introduction to Logical Systems*
- *Logical Systems Operations and Restrictions*

Unsupported Actions for Firewall Filters in Logical Systems

Table 41 on page 411 describes the firewall filter actions that are supported at the **[edit firewall]** hierarchy level, but not supported at the **[edit logical-systems *logical-system-name* firewall]** hierarchy level.

Table 41: Unsupported Actions for Firewall Filters in Logical Systems

Firewall Filter Action	Example	Description
Terminating Actions Not Supported in a Logical System		

logical-system	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  family inet {
	  filter foo {
	  term one {
	  from {
	  source-address 10.1.0.0/16;
	  }
	  then {
	  logical-system fred;
	  }
	  }
	  }
	  }
	  }
	  }
	}
		Because the **logical-system** action refers to **fred**—a logical system defined outside the local logical system—, this action is not supported.

Nonterminating Actions Not Supported in a Logical System		

ipsec-sa	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  family inet {
	  filter foo {
	  term one {
	  from {
	  source-address 10.1.0.0/16;
	  }
	  then {
	  ipsec-sa barney;
	  }
	  }
	  }
	  }
	  }
	  }
	}
		Because the **ipsec-sa** action modifier references **barney**—a security association defined outside the local logical system—this action is not supported.

Table 41: Unsupported Actions for Firewall Filters in Logical Systems *(continued)*

Firewall Filter Action	Example	Description

next-hop-group	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  family inet {
	  filter foo {
	  term one {
	  from {
	  source-address 10.1.0.0/16;
	  }
	  then {
	  next-hop-group fred;
	  }
	  }
	  }
	  }
	  }
	  }
	}
		Because the **next-hop-group** action refers to **fred**—an object defined at the **[edit forwarding-options next-hop-group]** hierarchy level—this action is not supported.

port-mirror	[edit]
	logical-systems {
	  ls1 {
	  firewall {
	  family inet {
	  filter foo {
	  term one {
	  from {
	  source-address 10.1.0.0/16;
	  }
	  then {
	  port-mirror;
	  }
	  }
	  }
	  }
	  }
	  }
	}
		Because the **port-mirror** action relies on a configuration defined at the **[edit forwarding-options port-mirroring]** hierarchy level, this action is not supported.

Table 41: Unsupported Actions for Firewall Filters in Logical Systems *(continued)*

Firewall Filter Action	Example	Description

sample	[edit]
	logical-systems {
	ls1 {
	firewall {
	family inet {
	filter foo {
	term one {
	from {
	source-address 10.1.0.0/16;
	}
	then {
	sample;
	}
	}
	}
	}
	}
	}
	}
		In this example, the **sample** action depends on the sampling configuration defined under the **[edit forwarding-options]** hierarchy. Therefore, the **sample** action is not supported.

syslog	[edit]
	logical-systems {
	ls1 {
	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;
	}
	}
	}
	}
	}
	}
		In this example, there must be at least one system log (**system syslog file *filename***) with the **firewall** facility enabled for the **icmp-syslog** filter's logs to be stored.
		
		Because this firewall configuration relies on a configuration outside the logical system, the **syslog** action modifier is not supported.

Related Documentation

- Firewall Filters in Logical Systems Overview on page 31
- Guidelines for Configuring and Applying Firewall Filters in Logical Systems on page 32
- Unsupported Firewall Filter Statements for Logical Systems on page 409
- *Introduction to Logical Systems*

- *Logical Systems Operations and Restrictions*

PART 4

Index

Index

Symbols

A

B

C

D

L

M

N

O

P

R

S