



Junos[®] OS

CoS Packet Classification Based on Various Packet Header Fields

Release
13.1



Published: 2013-02-11

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

This product includes the Envoy SNMP Engine, developed by Epilogue Technology, an Integrated Systems Company. Copyright © 1986-1997, Epilogue Technology Corporation. All rights reserved. This program and its documentation were developed at private expense, and no part of them is in the public domain.

This product includes memory allocation software developed by Mark Moraes, copyright © 1988, 1989, 1993, University of Toronto.

This product includes FreeBSD software developed by the University of California, Berkeley, and its contributors. All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by the Regents of the University of California. Copyright © 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994. The Regents of the University of California. All rights reserved.

GateD software copyright © 1995, the Regents of the University. All rights reserved. Gate Daemon was originated and developed through release 3.0 by Cornell University and its collaborators. Gated is based on Kirton's EGP, UC Berkeley's routing daemon (routed), and DCN's HELLO routing protocol. Development of Gated has been supported in part by the National Science Foundation. Portions of the GateD software copyright © 1988, Regents of the University of California. All rights reserved. Portions of the GateD software copyright © 1991, D. L. S. Associates.

This product includes software developed by Maker Communications, Inc., copyright © 1996, 1997, Maker Communications, Inc.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

Junos® OS CoS Packet Classification Based on Various Packet Header Fields

13.1

Copyright © 2013, Juniper Networks, Inc.
All rights reserved.

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

	About the Documentation	vii
	Documentation and Release Notes	vii
	Supported Platforms	vii
	Using the Examples in This Manual	vii
	Merging a Full Example	viii
	Merging a Snippet	viii
	Documentation Conventions	ix
	Documentation Feedback	xi
	Requesting Technical Support	xi
	Self-Help Online Tools and Resources	xi
	Opening a Case with JTAC	xii
Part 1	Overview	
Chapter 1	Overview	3
	Multifield Classifier Overview	3
	Overview of Simple Filters	4
	Two-Color Policers and Shaping Rate Changes	5
	Understanding IEEE 802.1p Inheritance push and swap from a Transparent Tag	5
Part 2	Configuration	
Chapter 2	Configuration Tasks	9
	Configuring Multifield Classifiers	9
	Configuring Logical Bandwidth Policers	10
	Configuring IEEE 802.1p Inheritance push and swap from the Transparent Tag	10
Chapter 3	Examples	13
	Example: Classifying Packets Based on Their Destination Address	13
	Example: Configuring and Verifying a Complex Multifield Filter	14
	Configuring a Complex Multifield Filter	14
	Verifying a Complex Multifield Filter	16
	Example: Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets	17
	Example: Configuring a Simple Filter	19
	Example: Configuring a Logical Bandwidth Policer	20
	Example: Two-Color Policers and Shaping Rate Changes	21
	Example: Multifield Classifier Limitation on M Series Routers	22

Chapter 4	Configuration Statements	25
	[edit class-of-service] Hierarchy Level	25
	transparent	29
	[edit firewall] Hierarchy Level	29
	dscp (Multifield Classifier)	31
	family (Multifield Classifier)	32
	filter (Configuring)	33
	forwarding-class (Multifield Classifiers)	34
	from	34
	loss-priority (Normal Filter)	35
	loss-priority (Simple Filter)	35
	simple-filter (Configuring)	36
	term (Simple Filter)	37
	then	38
	[edit interfaces] Hierarchy Level	38
	filter (Applying to an Interface)	40
	simple-filter (Applying to an Interface)	41
Part 3	Index	
	Index	45

List of Tables

About the Documentation	vii
Table 1: Notice Icons	ix
Table 2: Text and Syntax Conventions	ix

About the Documentation

- [Documentation and Release Notes on page vii](#)
- [Supported Platforms on page vii](#)
- [Using the Examples in This Manual on page vii](#)
- [Documentation Conventions on page ix](#)
- [Documentation Feedback on page xi](#)
- [Requesting Technical Support on page xi](#)

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:

- [T Series](#)
- [M Series](#)
- [MX 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.xml;
    }
  }
}
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.xml; }
```


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 ix 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 ix 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
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies book names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS System Basics Configuration Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	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 <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> 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]
Indentation and braces ({ })	Identify a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; retain; } } }
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
J-Web GUI Conventions		
Bold text like this	Represents J-Web graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none"> In the Logical Interfaces box, select All Interfaces. To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of J-Web 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

- [Overview on page 3](#)

CHAPTER 1

Overview

- [Multifield Classifier Overview on page 3](#)
- [Overview of Simple Filters on page 4](#)
- [Two-Color Policers and Shaping Rate Changes on page 5](#)
- [Understanding IEEE 802.1p Inheritance push and swap from a Transparent Tag on page 5](#)

Multifield Classifier Overview

A multifield classifier is a method of classifying traffic flows. Devices that sit at the edge of a network usually classify packets according to codings that are located in multiple packet header fields. Multifield classification is normally performed at the network edge because of the general lack of DiffServ code point (DSCP) or IP precedence support in end-user applications.

In an edge router, a multifield classifier provides the filtering functionality that scans through a variety of packet fields to determine the forwarding class for a packet. Typically, a classifier performs matching operations on the selected fields against a configured value.

Unlike a behavior aggregate (BA), which classifies packets based on class-of-service (CoS) bits in the packet header, a multifield classifier can examine multiple fields in the packet header—for example, the source and destination address of the packet, and the source and destination port numbers of the packet. A multifield classifier typically matches one or more of the six packet header fields: destination address, source address, IP protocol, source port, destination port, and DSCP. Multifield classifiers are used when a simple BA classifier is insufficient to classify a packet.

In the Juniper Networks® Junos® operating system (Junos OS), you configure a multifield classifier with a firewall filter and its associated match conditions. This enables you to use any filter match criteria to locate packets that require classification. From a CoS perspective, multifield classifiers (or firewall filter rules) provide the following services:

- Classify packets to a forwarding class and loss priority. The forwarding class determines the output queue. The loss priority is used by schedulers in conjunction with the random early discard (RED) algorithm to control packet discard during periods of congestion.

- Police traffic to a specific bandwidth and burst size. Packets exceeding the policer limits can be discarded, or can be assigned to a different forwarding class, to a different loss priority, or to both.



NOTE: You *police* traffic on input to conform to established CoS parameters, setting loss handling and forwarding class assignments as needed. You *shape* traffic on output to make sure that router resources, especially bandwidth, are distributed fairly. However, input policing and output shaping are two different CoS processes, each with their own configuration statements.

Overview of Simple Filters

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

Unlike normal filters, simple filters are for IPv4 traffic only and have the following restrictions:

- The **next term** action is not supported.
- Qualifiers, such as the **except** and **protocol-except** statements, are not supported.
- Noncontiguous masks are not supported.
- Multiple source addresses and destination addresses in a single term are not supported. If you configure multiple addresses, only the last one is used.
- Ranges are only valid as source or destination ports. For example, **source-port 400-500** or **destination-port 600-700**.
- Output filters are not supported. You can apply a simple filter to ingress traffic only.
- Simple filters are not supported for interfaces in an aggregated-Ethernet bundle.
- Explicitly configurable terminating actions, such as **accept**, **reject**, and **discard**, are not supported. Simple filters always accept packets.



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

Two-Color Policers and Shaping Rate Changes

When you configure a change in shaping rate, it is important to consider the effect on the bandwidth limit. Whenever the shaping rate changes, the bandwidth limit is adjusted based on whether a logical interface (unit) or bandwidth percentage policer is configured.

When a logical interface bandwidth policer is configured, the order of priority for the shaping rate (if configured at that level) is:

- The shaping rate applied to the logical interface (unit).
- The shaping rate applied to the physical interface (port).
- The physical interface speed.

When a bandwidth percentage policer is configured, the order of priority for the shaping rate (if configured at that level) is:

- The shaping rate applied to the physical interface (port).
- The physical interface speed.

These guidelines must be kept in mind when calculating the logical link speed and link speed from the configured shaping rate, which determines the rate-limited bandwidth after the policer is applied.

Related Documentation

- [Example: Two-Color Policers and Shaping Rate Changes on page 21](#)

Understanding IEEE 802.1p Inheritance push and swap from a Transparent Tag

M320 router interfaces and MX Series router interfaces on Modular Interface Cards (MIC) or Modular Port Concentrators (MPCs) support configurable IEEE 802.1p inheritance of push and swap bits from the transparent tag of each incoming packet which allows you to classify incoming packets based on the IEEE 802.1p bits from the transparent tag.

During a tagging operation, Junos OS by default inherits the IEEE 802.1p bits from incoming tags in swap and push operations from the known tags configured on the interface.

It can be useful to override the default behavior by configuring Junos OS to inherit the IEEE 802.1p bits from a transparent tag, and to classify incoming packets based on the IEEE 802.1p bits of the incoming transparent tag. The configuration statements **swap-by-poppush** and **transparent** enable Junos OS to do this.

By default, during a swap operation, the IEEE 802.1p bits of the VLAN tag remain unchanged. When the **swap-by-poppush** operation is enabled on a logical interface, the swap operation is treated as a **pop** operation followed by **push** operation. The **pop** operation removes the existing tag and the associated IEEE 802.1p bits and the push operation copies the inner VLAN IEEE 802.1p bits to the IEEE bits of the VLAN or VLANs being pushed. As a result, the IEEE 802.1p bits are inherited from the incoming transparent tag.

To classify incoming packets based on the IEEE 802.1p bits from the transparent tag, include the **transparent** statement at the **[edit class-of-service interfaces *interface-name* unit *logical-unit-number* classifiers ieee-802.1 vlan-tag]** hierarchy level.

To configure Junos OS to inherit the IEEE 802.1p bits from the transparent tag, include the **swap-by-poppush** statement at the **[edit interfaces *interface-name* unit *logical-unit-number*]** hierarchy level.



NOTE: IEEE 802.1p Inheritance push and swap is only supported on untagged and single-tagged logical interfaces, and is not supported on dual-tagged logical interfaces.

**Related
Documentation**

- swap-by-poppush
- [transparent on page 29](#)
- Understanding swap-by-poppush
- [Configuring IEEE 802.1p Inheritance push and swap from the Transparent Tag on page 10](#)
- Understanding Transparent Tag Operations and IEEE 802.1p Inheritance

PART 2

Configuration

- [Configuration Tasks on page 9](#)
- [Examples on page 13](#)
- [Configuration Statements on page 25](#)

CHAPTER 2

Configuration Tasks

- [Configuring Multifield Classifiers on page 9](#)
- [Configuring Logical Bandwidth Policers on page 10](#)
- [Configuring IEEE 802.1p Inheritance push and swap from the Transparent Tag on page 10](#)

Configuring Multifield Classifiers

If you configure both a behavior aggregate (BA) classifier and a multifield classifier, BA classification is performed first; then multifield classification is performed. If they conflict, any BA classification result is overridden by the multifield classifier.



NOTE: For a specified interface, you can configure both a multifield classifier and a BA classifier without conflicts. Because the classifiers are always applied in sequential order, the BA classifier followed by the multifield classifier, any BA classification result is overridden by a multifield classifier if they conflict.

To activate a multifield classifier, you must configure it on a logical interface. There is no restriction on the number of multifield classifiers you can configure.



NOTE: For MX Series routers, if you configure a firewall filter with a DSCP action or traffic-class action on a DPC, the commit does not fail, but a warning displays and an entry is made in the syslog.

For an L2TP LNS on MX Series routers, you can attach firewall for static LNS sessions by configuring these at logical interfaces directly on the inline services device (si-fpc/pic/port). RADIUS-configured firewall attachments are not supported.

To configure multifield classifiers, include the following statements at the **[edit firewall]** hierarchy level:

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

```
filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      dscp 0;
      forwarding-class class-name;
      loss-priority (high | low);
    }
  }
}
simple-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      forwarding-class class-name;
      loss-priority (high | low | medium);
    }
  }
}
```

Configuring Logical Bandwidth Policers

When you configure a policer as a percentage (using the **bandwidth-percent** statement), the bandwidth is calculated as a percentage of either the physical interface media rate or the logical interface shaping rate. To specify that the bandwidth be calculated based on the logical interface shaping rate and not the physical interface media rate, include the **logical-bandwidth-policer** statement. If a shaping rate is not configured for the logical interface, the physical interface media rate is used, even if you include the **logical-bandwidth-policer**. You can configure the shaping rate on the logical interface using class-of-service statements.

```
[edit firewall policer policer-name]
logical-bandwidth-policer;
```

Configuring IEEE 802.1p Inheritance push and swap from the Transparent Tag

To classify incoming packets based on the IEEE 802.1p bits from the transparent tag, include the **transparent** statement at the **[edit class-of-service interfaces interface-name unit logical-unit-number classifiers ieee-802.1 vlan-tag]** hierarchy level.

Tagged Interface Example

The following example configuration specifies the classification based on the transparent VLAN tag.

```
edit
class-of-service {
  interfaces {
    ge-3/0/1 {
      unit 0 {
        classifiers {
```

```

        ieee-802.1p default vlan-tag transparent;
    }
}
}
}
}

```

To configure Junos OS to inherit the IEEE 802.1p bits from the transparent tag, include the **swap-by-poppush** statement at the **[edit interfaces *interface-name* unit *logical-unit-number*]** hierarchy level.

The following is a configuration to swap and push VLAN tags and allow inheritance of the IEEE 802.1p value from the transparent VLAN tag in incoming packets.

```

edit
ge-3/0/0 {
  vlan-tagging;
  encapsulation vlan-ccc;
  unit 0 {
    encapsulation vlan-ccc;
    vlan-id 100;
    swap-by-poppush;
    input-vlan-map {
      swap-push;
      tag-protocol-id 0x9100;
      inner-tag-protocol-id 0x9100;
      vlan-id 500;
      inner-vlan-id 400;
    }
    output-vlan-map {
      pop-swap;
      inner-vlan-id 100;
      inner-tag-protocol-id 0x88a8;
    }
  }
}

```

The **swap-by-poppush** statement causes a swap operation to be done as a pop followed by a push operation. So for the outer tag, the incoming S-Tag is popped and a new tag is pushed. As a result, the S-Tag inherits the IEEE 802.1p bits from the transparent tag. The inner tag is then pushed, which results in the inner tag inheriting the IEEE 802.1p bits from the transparent tag.

Untagged Interface Example

The following is a configuration to push two VLAN tags and allow inheritance of the IEEE 802.1p value from the transparent VLAN tag in the incoming packet.

```

[edit]
ge-3/0/1 {
  encapsulation ccc;
  unit 0 {
    input-vlan-map {
      push-push;
      tag-protocol-id 0x9100;
      inner-tag-protocol-id 0x9100;
      vlan-id 500;
      inner-vlan-id 400;
    }
  }
}

```

```
    }  
    output-vlan-map {  
        pop-pop;  
    }  
}  
}
```

No additional configuration is required to inherit the IEEE 802.1p value, as the **push** operation inherits the IEEE 802.1p values by default.

The following configuration specifies the classification based on the transparent VLAN tag.

```
[edit]  
class-of-service {  
    interfaces {  
        ge-3/0/1 {  
            unit 0 {  
                classifiers {  
                    ieee-802.1 default vlan-tag transparent;  
                }  
            }  
        }  
    }  
}
```

**Related
Documentation**

- [transparent on page 29](#)
- [swap-by-poppush](#)
- [Understanding IEEE 802.1p Inheritance push and swap from a Transparent Tag on page 5](#)
- [Understanding swap-by-poppush](#)
- [Understanding Transparent Tag Operations and IEEE 802.1p Inheritance](#)

CHAPTER 3

Examples

- [Example: Classifying Packets Based on Their Destination Address on page 13](#)
- [Example: Configuring and Verifying a Complex Multifield Filter on page 14](#)
- [Example: Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets on page 17](#)
- [Example: Configuring a Simple Filter on page 19](#)
- [Example: Configuring a Logical Bandwidth Policer on page 20](#)
- [Example: Two-Color Policers and Shaping Rate Changes on page 21](#)
- [Example: Multifield Classifier Limitation on M Series Routers on page 22](#)

Example: Classifying Packets Based on Their Destination Address

Configure a multifield classifier that ensures that all IPv4 packets destined for the **10.10.10.0/24** network are placed into the **platinum** forwarding class. This assignment occurs regardless of the received CoS bit values in the packet. Apply this filter to the inbound interface **so-1/2/2.0**.

To verify that your configuration is attached to the correct interface, issue the **show interfaces filters** command.

```
[edit]
firewall {
  family inet {
    filter set-FC-to-platinum {
      term match-a-single-route {
        from {
          destination-address {
            10.10.10.0/24;
          }
        }
        then {
          forwarding-class platinum;
          accept;
        }
      }
      term accept-all {
        then accept;
      }
    }
  }
}
```

```
    }
  }
  interfaces {
    so-1/2/2 {
      unit 0 {
        family inet {
          filter {
            input set-FC-to-platinum;
          }
        }
      }
    }
  }
}
```

Example: Configuring and Verifying a Complex Multifield Filter

In this example, SIP signaling (VoIP) messages use TCP/UDP, port 5060, and RTP media channels use UDP with port assignments from 16,384 through 32,767. See the following sections:

- [Configuring a Complex Multifield Filter on page 14](#)
- [Verifying a Complex Multifield Filter on page 16](#)

Configuring a Complex Multifield Filter

To configure the multifield filter, perform the following actions:

- Classify SIP signaling messages (VoIP network control traffic) as NC with a firewall filter.
- Classify VoIP traffic as EF with the same firewall filter.
- Police all remaining traffic with IP precedence **0** and make it BE.
- Police BE traffic to 1 Mbps with excess data marked with PLP high.
- Apply the firewall filter with policer to the interface.

The firewall filter called **classify** matches on the transport protocol and ports identified in the incoming packets and classifies packets into the forwarding classes specified by your criteria.

The first term, **sip**, classifies SIP signaling messages as network control messages. The **port** statement matches any source port or destination port (or both) that is coded to 5060.

Classifying SIP Signaling Messages

```
firewall {
  family inet {
    filter classify {
      interface-specific;
      term sip {
        from {
          protocol [ udp tcp ];
        }
      }
    }
  }
}
```

```

        port 5060;
    }
    then {
        forwarding-class network-control;
        accept;
    }
}
}
}
}
}

```

The second term, **rtp**, classifies VoIP media channels that use UDP-based transport.

Classifying VoIP Channels That Use UDP

```

term rtp {
    from {
        protocol udp;
        port 16384-32767;
    }
    then {
        forwarding-class expedited-forwarding;
        accept;
    }
}

```

The policer's burst tolerance is set to the recommended value for a low-speed interface, which is ten times the interface MTU. For a high-speed interface, the recommended burst size is the transmit rate of the interface times 3 to 5 milliseconds.

Configuring the Policer

```

policer be-policer {
    if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
    }
    then loss-priority high;
}

```

The third term, **be**, ensures that all remaining traffic is policed according to a bandwidth restriction.

Policing All Remaining Traffic

```

term be {
    then policer be-policer;
}

```

The **be** term does not include a **forwarding-class** action modifier. Furthermore, there is no explicit treatment of network control (NC) traffic provided in the **classify** filter. You can configure explicit classification of NC traffic and all remaining IP traffic, but you do not need to, because the default IP precedence classifier correctly classifies the remaining traffic.

Apply the **classify** classifier to the **fe-0/0/2** interface:

Applying the Classifier

```
interfaces {
  fe-0/0/2 {
    unit 0 {
      family inet {
        filter {
          input classify;
        }
        address 10.12.0.13/30;
      }
    }
  }
}
```

Verifying a Complex Multifield Filter

Before the configuration is committed, display the default classifiers in effect on the interface using the **show class-of-service interface *interface-name*** command. The display confirms that the **ipprec-compatibility** classifier is in effect by default.

Verifying Default Classification

```
user@host> show class-of-service fe-0/0/2
Physical interface: fe-0/0/2, Index: 135
Queues supported: 8, Queues in use: 4
Scheduler map: <default>, Index: 2032638653
```

```
Logical interface: fe-0/0/2.0, Index: 68
Shaping rate: 32000
Object      Name                               Type      Index
Scheduler-map <default>                               27
Rewrite      exp-default                               exp       21
Classifier    exp-default                               exp       5
Classifier    ipprec-compatibility                       ip        8
```

To view the default classifier mappings, use the **show class-of-service classifier *name*** command. The highlighted output confirms that traffic with IP precedence setting of 0 is correctly classified as BE, and NC traffic, with precedence values of 6 or 7, is properly classified as NC.

Displaying Default Classifier Mappings

```
user@host> show class-of-service classifier name ipprec-compatibility
Classifier: ipprec-compatibility, Code point type: inet-precedence, Index: 12
Code point  Forwarding class  Loss priority
000         best-effort  low
001         best-effort  high
010         best-effort  low
011         best-effort  high
100         best-effort  low
101         best-effort  high
110         network-control low
111         network-control high
```

After your configuration is committed, verify that your multifield classifier is working correctly. You can monitor the queue counters for the router's **egress** interface used when forwarding traffic received from the peer. Displaying the queue counters for the ingress interface (**fe-0/0/2**) does not allow you to check your ingress classification, because

queuing generally occurs only at egress in the Junos OS. (Ingress queuing is supported on Gigabit Ethernet IQ2 PICs and Enhanced IQ2 PICs only.)

To verify the operation of the multifield filter:

1. To determine which egress interface is used for the traffic, use the **traceroute** command.
2. After you identify the egress interface, clear its associated queue counters by issuing the **clear interfaces statistics interface-name** command.
3. Confirm the default forwarding class-to-queue number assignment. This allows you to predict which queues are used by the VoIP, NC, and other traffic. To do this, issue the **show class-of-service forwarding-class** command.
4. Display the queue counts on the interface by issuing the **show interfaces queue** command.

Example: Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets

On Juniper Networks M320 Multiservice Edge Routers and T Series Core Routers, you can selectively set the DSCP field of MPLS-tagged IPv4 and IPv6 packets to **000000**. In the same packets, you can set the MPLS EXP field according to a configured rewrite table, which is based on the forwarding classes that you set in incoming packets using a BA or multifield classifier.

Queue selection is based on the forwarding classes you assign in scheduler maps. This means that you can direct traffic to a single output queue, regardless of whether the DSCP field is unchanged or rewritten to **000000**. To do this, you must configure a multifield classifier that matches selected packets and modifies them with the **dscp 0** action.

Selective marking of DSCP fields to **0**, without affecting output queue assignment, can be useful. For example, suppose you need to use the MPLS EXP value to configure CoS applications for core provider routers. At the penultimate egress provider edge (PE) router where the MPLS labels are removed, the CoS bits need to be provided by another value, such as DSCP code points. This case illustrates why it is useful to mark both the DSCP and MPLS EXP fields in the packet. Furthermore, it is useful to be able to mark the two fields differently, because the CoS rules of the core provider router might differ from the CoS rules of the egress penultimate router. At egress, as always, you can use a rewrite table to rewrite the MPLS EXP values corresponding to the forwarding classes that you need to set.



NOTE: When both customer-facing and core-facing interfaces exist, you can derive the EXP value in the following precedence order, while adding the MPLS label:

1. EXP value provided by the CoS rewrite action.
2. EXP value derived from the top label of the stack (MPLS label stacking).
3. IPv4 or IPv6 precedence (Layer 3 VPN, Layer 2 VPN, and VPLS scenarios).

For IPv4 traffic, the **dscp 0** action modifier at the **[edit firewall family inet filter *filter-name* term *term-name* then]** hierarchy level is valid. However, for IPv6 traffic, you configure this feature by including the **traffic-class 0** action modifier at the **[edit firewall family inet6 filter *filter-name* term *term-name* then]** hierarchy level.

In the following IPv4 example, term 1 of the multifield classifier matches packets with DSCP **001100** code points coming from a certain VRF, rewrites the bits to DSCP **000000**, and sets the forwarding class to **best-effort**. In term 2, the classifier matches packets with DSCP **010110** code points and sets the forwarding class to **best-effort**. Because term 2 does not include the **dscp 0** action modifier, the DSCP **010110** bits remain unchanged. Because the classifier sets the forwarding class for both code points to **best-effort**, both traffic types are directed to the same output queue.



NOTE: If you configure a bit string in a DSCP match condition in a firewall filter, then you must include the letter “b” in front of the string, or the match rule creation fails on commit.

```
[edit]
firewall {
  family inet {
    filter vrf-rewrite {
      term 1 {
        from {
          dscp b001100;
        }
        then {
          dscp 0;
          forwarding-class best-effort;
        }
      }
      term 2 {
        from {
          dscp b010110;
        }
        then {
          forwarding-class best-effort;
        }
      }
    }
  }
}
```

Applying the Multifield Classifier

Apply the filter to an input interface corresponding to the VRF:

```
[edit]
interfaces {
  so-0/1/0 {
    unit 0 {
      family inet {
        filter input vrf-rewrite;
      }
    }
  }
}
```

}



NOTE: The `dscp 0` action is supported in both input and output filters. You can use this action for non-MPLS packets as well as for IPv4 and IPv6 packets entering an MPLS network. All IPv4 and IPv6 firewall filter match conditions are supported with the `dscp 0` action.

The following limitations apply:

- You can use a multifield classifier to rewrite DSCP fields to value 0 only. Other values are not supported.
- If a packet matches a filter that has the `dscp 0` action, then the outgoing DSCP value of the packet is 0, even if the packet matches a rewrite rule, and the rewrite rule is configured to mark the packet to a non-zero value. The `dscp 0` action overrides any other rewrite rule actions configured on the router.
- Although you can use the `dscp 0` action on an input filter, the output filter and other classifiers do not see the packet as being marked `dscp 0`. Instead, they classify the packet based on its original incoming DSCP value. The DSCP value of the packet is set to 0 after all other classification actions have completed on the packet.

Example: Configuring a Simple Filter

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. 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`.

```
[edit]
firewall {
  family inet {
    simple-filter filter1 {
      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;
    }
}
then loss-priority high;
term 3 {
    from {
        destination-address {
            6.6.6.6/32;
        }
    }
    then {
        loss-priority low;
        forwarding-class best-effort;
    }
}
}
}
}
}
interfaces {
    ge-0/0/1 {
        unit 0 {
            family inet {
                simple-filter {
                    input filter1;
                }
                address 10.1.2.3/30;
            }
        }
    }
}
}
```

Example: Configuring a Logical Bandwidth Policer

This example applies a logical bandwidth policer rate to two logical interfaces on interface **ge-0/2/7**. The policed rate on **unit 0** is 2 Mbps (50 percent of 4 Mbps) and the policed rate on **unit 1** is 1 Mbps (50 percent of 2 Mbps).

```
[edit firewall]
policer Logical_Policer {
    logical-bandwidth-policer; # This applies the policer to logical interfaces
    if-exceeding {
        bandwidth-percent 50; # This applies 50 percent to the shaping-rate
        burst-size-limit 125k;
    }
    then discard;
}

[edit class-of-service]
interfaces {
    ge-0/2/7 {
        unit 0 {
            shaping-rate 4m # This establishes the rate to be policed on unit 0
        }
    }
}
```



```

    }
    unit 1 {
        shaping-rate 2m # This establishes the rate to be policed on unit 1
    }
}
[edit interfaces ge-0/2/7]
per-unit-scheduler;
vlan-tagging;
unit 0 {
    vlan-id 100;
    family inet {
        policer {
            input Logical_Policer;
            output Logical_Policer;
        }
        address 172.1.1.1/30;
    }
}
unit 1 {
    vlan-id 200;
    family inet {
        policer {
            input Logical_Policer;
            output Logical_Policer;
        }
        address 172.2.1.1/30;
    }
}
}

```

Example: Two-Color Policers and Shaping Rate Changes

In this example, the shaping rate has been configured for the logical interface, but a bandwidth percentage policer is also configured. Therefore policing is based on the physical interface speed of 1 Gbps.

If both a shaping rate and a bandwidth percentage policer is configured on the same logical interface, the policing is based on the physical interface speed. Here is the example configuration:

```

[edit interfaces]
ge-0/1/0 {
    per-unit-scheduler;
    vlan-tagging;
    unit 0 {
        vlan-id 1;
        family inet {
            policer {
                output policer_test;
            }
            address 10.0.7.1/24;
        }
    }
}
}

```

```
[edit firewall]
policer policer_test {
  if-exceeding {
    bandwidth-percent 75;
    burst-size-limit 256k;
  }
  then discard;
}

[edit]
class-of-service {
  interfaces {
    ge-0/1/0 {
      unit 0 {
        shaping-rate 15m;
      }
    }
  }
}
}
```

Example: Multifield Classifier Limitation on M Series Routers

On M Series routers (except M120 routers), multifield classifiers are limited such that they cannot classify packets with an output filter match based on the ingress classification that is set with an input filter.

For example, in the following configuration, the filter called **ingress** assigns all incoming IPv4 packets to the **expedited-forwarding** class. The filter called **egress** counts all packets that were assigned to the **expedited-forwarding** class in the **ingress** filter. This configuration does not work on most M Series routers. It works on all other routing platforms, including M120 routers, T Series routers, and MX Series routers.

```
user@host # show firewall
family inet {
  filter ingress {
    term 1 {
      then {
        forwarding-class expedited-forwarding;
        accept;
      }
    }
    term 2 {
      then accept;
    }
  }
  filter egress {
    term 1 {
      from {
        forwarding-class expedited-forwarding;
      }
      then count ef;
    }
    term 2 {
      then accept;
    }
  }
}
```

```
    }  
  }  
}
```

As a workaround, you can configure all of the actions in the ingress filter. For example:

```
user@host # show firewall  
family inet {  
  filter ingress {  
    term 1 {  
      then {  
        forwarding-class expedited-forwarding;  
        accept;  
        count ef;  
      }  
    }  
    term 2 {  
      then accept;  
    }  
  }  
}
```


CHAPTER 4

Configuration Statements

- [\[edit class-of-service\] Hierarchy Level on page 25](#)
- [\[edit firewall\] Hierarchy Level on page 29](#)
- [\[edit interfaces\] Hierarchy Level on page 38](#)

[edit class-of-service] Hierarchy Level

This topic shows the complete configuration for class of service (CoS) statements for the **[edit class-of-service]** hierarchy level, listing all possible configuration statements and showing their level in the configuration hierarchy. When you are configuring Junos OS, your current hierarchy level is shown in the banner on the line preceding the **user@host#** prompt.

For a complete list of the Junos OS configuration statements, see the *Junos OS Hierarchy and RFC Reference*.

```
[edit class-of-service]
adjustment-control-profiles {
  profile-name {
    application {
      ancp;
      radius-coa;
      pppoe-tags;
    }
  }
}
classifiers {
  (dscp | dscp-ipv6 | exp | ieee-802.1 | inet-precedence) classifier-name {
    import (classifier-name | default);
    forwarding-class class-name {
      loss-priority level code-points [ aliases ] [ bit-patterns ];
    }
  }
}
code-point-aliases {
  (dscp | dscp-ipv6 | exp | ieee-802.1 | inet-precedence) {
    alias-name bits;
  }
}
copy-plp-all;
drop-profiles {
```

```
    profile-name {
        fill-level percentage drop-probability percentage;
        interpolate {
            drop-probability [ values ];
            fill-level [ values ];
        }
    }
}
fabric {
    scheduler-map {
        priority (high | low) scheduler scheduler-name;
    }
}
forwarding-classes {
    class class-name queue-num queue-number priority (high | low);
    queue queue-number class-name priority (high | low) [ policing-priority (premium |
        normal) ];
}
forwarding-class-map forwarding-class-map-name {
    class class-name queue-num queue-number [ restricted-queue queue-number ];
}
forwarding-policy {
    next-hop-map map-name {
        forwarding-class class-name {
            next-hop [ next-hop-name ];
            lsp-next-hop [ lsp-regular-expression ];
            non-lsp-next-hop;
            discard;
        }
    }
    class class-name {
        classification-override {
            forwarding-class class-name;
        }
    }
}
fragmentation-maps {
    map-name {
        forwarding-class class-name {
            drop-timeout milliseconds;
            fragment-threshold bytes;
            multilink-class number;
            no-fragmentation;
        }
    }
}
host-outbound-traffic {
    dscp-code-point value;
    forwarding-class class-name;
    ieee-802.1 {
        default value;
        rewrite-rules;
    }
}
interfaces {
    interface-name {
```

```

classifiers {
    dscp (classifier-name | default);
    ieee-802.1 (classifier-name | default) vlan-tag (inner | outer | classifier-name);
    inet-precedence (classifier-name | default);
}
input-scheduler-map map-name;
input-shaping-rate rate;
irb {
    unit logical-unit-number {
        classifiers {
            dscp (classifier-name | default) {
                family [ inet mpls ];
            }
            dscp-ipv6 (classifier-name | default) {
                family [ inet mpls ];
            }
            exp (classifier-name | default);
            ieee-802.1 (classifier-name | default) vlan-tag (inner | outer | transparent);
        }
        rewrite-rules {
            dscp (rewrite-name | default);
            dscp-ipv6 (rewrite-name | default);
            exp (rewrite-name | default) protocol protocol-types;
            ieee-802.1 (rewrite-name | default) vlan-tag (outer | outer-and-inner);
            inet-precedence (rewrite-name | default);
        }
    }
}
output-forwarding-class-map forwarding-class-map-name;
member-link-scheduler (replicate | scale);
rewrite-rules {
    dscp (rewrite-name | default);
    ieee-802.1 (rewrite-name | default) vlan-tag (outer);
    inet-precedence (rewrite-name | default);
}
scheduler-map map-name;
scheduler-map-chassis map-name;
shaping-rate rate;
unit logical-unit-number {
    classifiers {
        (dscp | dscp-ipv6 | exp | ieee-802.1 | inet-precedence) (classifier-name | default)
        family (mpls | inet);
    }
    forwarding-class class-name;
    fragmentation-map map-name;
    input-scheduler-map map-name;
    input-shaping-rate (percent percentage | rate);
    input-traffic-control-profile profile-name shared-instance instance-name;
    loss-priority-maps {
        frame-relay-de (name | default);
    }
    loss-priority-rewrites {
        frame-relay-de (name | default);
    }
    output-traffic-control-profile profile-name shared-instance instance-name;
    per-session-scheduler;
}

```

```
rewrite-rules {
    dscp (rewrite-name | default) protocol protocol-types;
    dscp-ipv6 (rewrite-name | default);
    exp (rewrite-name | default) protocol protocol-types;
    exp-push-push-push default;
    exp-swap-push-push default;
    ieee-802.1 (rewrite-name | default) vlan-tag (outer | outer-and-inner);
    inet-precedence (rewrite-name | default) protocol protocol-types;
}
scheduler-map map-name;
shaping-rate rate;
translation-table (to-dscp-from-dscp | to-dscp-ipv6-from-dscp-ipv6 |
    to-exp-from-exp | to-inet-precedence-from-inet-precedence) table-name;
}
}
loss-priority-maps {
    frame-relay-de (Defining Loss Priority Maps) name {
        loss-priority level code-points [alias | bits ];
    }
}
loss-priority-rewrites {
    frame-relay-de (Defining Loss Priority Maps) name {
        loss-priority level code-point (alias | bits );
    }
}
restricted-queues {
    forwarding-class class-name queue queue-number;
}
rewrite-rules {
    (dscp | dscp-ipv6 | exp | ieee-802.1 | ieee-802.1ad | inet-precedence) rewrite-name {
        import (rewrite-name | default);
        forwarding-class class-name {
            loss-priority level code-point (alias | bits);
        }
    }
}
routing-instances routing-instance-name {
    classifiers {
        exp (classifier-name | default);
        dscp (classifier-name | default);
        dscp-ipv6 (classifier-name | default);
    }
}
scheduler-maps {
    map-name {
        forwarding-class class-name scheduler scheduler-name;
    }
}
schedulers {
    scheduler-name {
        buffer-size (percent percentage | remainder | temporal microseconds);
        drop-profile-map loss-priority (any | low | medium-low | medium-high | high) protocol
            (any | non-tcp | tcp) drop-profile profile-name;
        excess-priority (low | high);
        excess-rate percent percentage;
    }
}
```



```

    excess-rate (percent percentage | proportion value);
    priority priority-level;
    transmit-rate (rate | percent percentage | remainder) <exact | rate-limit>;
  }
}
system-defaults {
  classifiers (classifier-name | exp)
traffic-control-profiles profile-name {
  delay-buffer-rate (percent percentage | rate);
  excess-rate (percent percentage | proportion value);
  guaranteed-rate (percent percentage | rate);
  overhead-accounting (frame-mode | cell-mode) <bytes byte-value>;
  scheduler-map map-name;
  shaping-rate (percent percentage | rate);
}
translation-table {
  (to-dscp-from-dscp | to-dscp-ipv6-from-dscp-ipv6 | to-exp-from-exp |
  to-inet-precedence-from-inet-precedence) table-name {
    to-code-point value from-code-points (* | [ values ]);
  }
}
tri-color;

```

On Juniper Networks MX Series 3D Universal Edge Routers with Enhanced Queuing DPCs, you can configure the following CoS statements at the **[edit class-of-service interfaces]** hierarchy level:

```

interface-set interface-set-name {
  excess-bandwidth-share (proportional value | equal);
  internal-node;
  traffic-control-profiles profile-name;
  output-traffic-control-profile-remaining profile-name;
}

```

transparent

Syntax	transparent;
Hierarchy Level	[edit class-of-service interfaces <i>interface-name</i> unit <i>logical-unit-number</i> classifiers ieee802.1 vlan-tag]
Release Information	Statement introduced in Junos OS Release 11.2
Description	Packet classification based on the transparent VLAN tag.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

[edit firewall] Hierarchy Level

The following CoS statements can be configured at the **[edit firewall]** hierarchy level. This is not a comprehensive list of statements available at the **[edit firewall]** hierarchy level.

```
[edit firewall]
```

```
atm-policer policer-name {
  cdvt rate;
  logical-interface-policer;
  max-burst-size max-burst-size;
  peak-rate rate;
  policing-action (discard | discard-tag | count);
  sustained-rate rate;
}
family family-name {
  filter filter-name {
    term term-name {
      from {
        match-conditions;
      }
      then {
        dscp 0;
        forwarding-class class-name;
        loss-priority (high | low);
        three-color-policer {
          (single-rate | two-rate) policer-name;
        }
      }
    }
  }
}
simple-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      forwarding-class class-name;
      loss-priority (high | low | medium);
    }
  }
}
}
policer policer-name {
  logical-bandwidth-policer;
  shared-bandwidth-policer;
  if-exceeding {
    bandwidth-limit rate;
    bandwidth-percent number;
    burst-size-limit bytes;
  }
  then {
    policer-action;
  }
}
three-color-policer policer-name {
  action {
    loss-priority high then discard;
  }
  logical-interface-policer;
  shared-bandwidth-policer;
  single-rate {
    (color-aware | color-blind);
  }
}
```

```

    committed-information-rate bps;
    committed-burst-size bytes;
    excess-burst-size bytes;
  }
  two-rate {
    (color-aware | color-blind);
    committed-information-rate bps;
    committed-burst-size bytes;
    peak-information-rate bps;
    peak-burst-size bytes;
  }
}

```

dscp (Multifield Classifier)

Syntax	<code>dscp [0 <i>value</i>];</code>
Hierarchy Level	[edit firewall family <i>family-name</i> filter <i>filter-name</i> term <i>term-name</i> then]
Release Information	Statement introduced in Junos OS Release 7.4.
Description	<p>For M320 and T Series routers, set the DSCP field of incoming or outgoing packets to 000000. On the same packets, you can use a behavior aggregate (BA) classifier and a rewrite rule to rewrite the MPLS EXP field.</p> <p>For MX Series routers with MPCs, the DSCP field can be set from a numeric range.</p> <p>For MX Series routers, if you configure a firewall filter with a DSCP action or traffic-class action on a DPC, the commit does not fail, but the filter is not applied to the interface, a warning displays, and an entry is made in the syslog.</p>
Options	value —For MX Series routers with MPCs, specify the field of incoming or outgoing packets in the range from 0 through 63 .
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none"> Applying Tricolor Marking Policers to Firewall Filters

family (Multifield Classifier)

Syntax family *family-name* {
 filter *filter-name* {
 term *term-name* {
 from {
 match-conditions;
 }
 then {
 dscp 0;
 forwarding-class *class-name*;
 loss-priority (high | low);
 three-color-policer {
 (single-rate | two-rate) *policer-name*;
 }
 }
 }
 }
 }
 }

Hierarchy Level [edit firewall]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure a firewall filter for IP version 4 (IPv4) or IP version 6 (IPv6) traffic.

Options *family-name*—Protocol family:

- **ccc**—Circuit cross-connect parameters
- **inet**—IPv4 parameters
- **inet6**—IPv6 protocol parameters
- **iso**—OSI ISO protocol parameters
- **mlppp**—Multilink PPP protocol parameters
- **mpls**—MPLS protocol parameters
- **tcc**—Translational cross-connect parameters
- **vpls**—Virtual private LAN service parameters.

The remaining statements are explained separately.

Required Privilege Level interface—To view this statement in the configuration.
 interface-control—To add this statement to the configuration.

Related Documentation • [Configuring Multifield Classifiers on page 9](#)

filter (Configuring)

Syntax	<pre> filter <i>filter-name</i> { accounting-profile <i>name</i>; enhanced-mode; interface-shared; interface-specific; physical-interface-filter; term <i>term-name</i> { filter <i>filter-name</i>; from { <i>match-conditions</i>; } then { <i>actions</i>; } } } </pre>
Hierarchy Level	<p>[edit dynamic-profiles <i>profile-name</i> firewall family <i>family-name</i>], [edit firewall family <i>family-name</i>], [edit logical-systems <i>logical-system-name</i> firewall family <i>family-name</i>]</p>
Release Information	<p>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 <i>family-name</i>] hierarchy level introduced in Junos OS Release 11.4. Support for the interface-shared> statement introduced in Junos OS Release 12.2.</p>
Description	Configure firewall filters.
Options	<p><i>filter-name</i>—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).</p> <p>The remaining statements are explained separately.</p>
Required Privilege Level	<p>firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration.</p>
Related Documentation	<ul style="list-style-type: none"> Guidelines for Configuring Standard Firewall Filters Guidelines for Applying Standard Firewall Filters Configuring Multifield Classifiers on page 9 Using Multifield Classifiers to Set PLP simple-filter

forwarding-class (Multifield Classifiers)

Syntax	<code>forwarding-class class-name;</code>
Hierarchy Level	<code>[edit firewall family <i>family-name</i> filter <i>filter-name</i> term <i>term-name</i> then]</code>
Release Information	Statement introduced before Junos OS Release 7.4.
Description	Set the forwarding class of incoming packets.
Options	<i>class-name</i> —Name of the forwarding class.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• Configuring Multifield Classifiers on page 9

from

Syntax	<pre>from { applications [<i>application-name</i>]; application-sets [<i>set-name</i>]; destination-address (CoS) <i>address</i>; source-address <i>address</i>; }</pre>
Hierarchy Level	<code>[edit services cos rule <i>rule-name</i> term <i>term-name</i>]</code>
Release Information	Statement introduced in Junos OS Release 8.1.
Description	Specify input conditions for a CoS term.
Options	The remaining statements are explained separately.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• Configuring CoS Rule Sets

loss-priority (Normal Filter)

Syntax	loss-priority (high low);
Hierarchy Level	[edit firewall family <i>family-name</i> filter <i>filter-name</i> term <i>term-name</i> then]
Release Information	Statement introduced before Junos OS Release 7.4.
Description	Set the loss priority of incoming packets.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none"> • Configuring Multifield Classifiers on page 9

loss-priority (Simple Filter)

Syntax	loss-priority (high low medium);
Hierarchy Level	[edit firewall family <i>family-name</i> simple-filter <i>filter-name</i> term <i>term-name</i> then]
Release Information	Statement introduced in Junos OS Release 7.6.
Description	Set the loss priority of incoming packets.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none"> • Configuring Multifield Classifiers on page 9

simple-filter (Configuring)

Syntax	<pre>simple-filter <i>filter-name</i> { term <i>term-name</i> { from { match-conditions; } then { forwarding-class <i>class-name</i>; loss-priority (high low medium); } } }</pre>
Hierarchy Level	[edit firewall family inet filter <i>filter-name</i>]
Release Information	Statement introduced in Junos OS Release 7.6.
Description	Define a simple filter. Simple filters are recommended for metropolitan Ethernet applications.
Options	<p>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.</p> <p>match-conditions—One or more conditions to use to make a match.</p> <p>term-name—Name that identifies the term. 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 (" ").</p> <p>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.</p> <p>The remaining statements are explained separately. Only forwarding-class and loss-priority are valid in a simple filter configuration.</p>
Required Privilege Level	firewall—To view this statement in the configuration. firewall-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• Configuring Multifield Classifiers on page 9• filter (Applying to an Interface) on page 40• simple-filter (Applying to an Interface) on page 41

term (Simple Filter)

Syntax	<pre>term <i>term-name</i> { from { <i>match-conditions</i>; } then { forwarding-class <i>class-name</i>; loss-priority (high low medium); } }</pre>
Hierarchy Level	[edit firewall family inet simple-filter <i>filter-name</i>]
Release Information	Statement introduced in Junos OS Release 7.6.
Description	Define a simple filter term.
Options	<p>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.</p> <p>match-conditions—One or more conditions to use to make a match.</p> <p>term-name—Name that identifies the term. 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 (" ").</p> <p>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. For CoS, only the actions listed are allowed. These statements are explained separately.</p>
Required Privilege Level	<p>firewall—To view this statement in the configuration.</p> <p>firewall-control—To add this statement to the configuration.</p>
Related Documentation	<ul style="list-style-type: none"> • Multifield Classification • Simple Filter Overview • Standard Firewall Filter Match Conditions for IPv4 Traffic • Standard Firewall Filter Match Conditions for IPv6 Traffic

then

Syntax	<pre>then { application-profile <i>profile-name</i>; dscp (<i>alias</i> <i>bits</i>); forwarding-class <i>class-name</i>; syslog; (reflexive reverse) { application-profile <i>profile-name</i>; dscp (<i>alias</i> <i>bits</i>); forwarding-class <i>class-name</i>; syslog; } }</pre>
Hierarchy Level	[edit services cos rule <i>rule-name</i> term <i>term-name</i>]
Release Information	Statement introduced in Junos OS Release 8.1.
Description	Define the CoS term actions. The remaining statements are explained separately.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">Configuring Actions in a CoS RuleConfiguring Actions in CoS Rules

[edit interfaces] Hierarchy Level

The following CoS statements can be configured at the **[edit interfaces]** hierarchy level. This is not a comprehensive list of statements available at the **[edit interfaces]** hierarchy level. Only the statements that are also documented in this manual are listed here. For more information about interface configuration, see the Junos® OS Network Interfaces.

```
[edit interfaces]
interface-name {
  atm-options {
    linear-red-profiles profile-name {
      high-plp-max-threshold percent;
      low-plp-max-threshold percent;
      queue-depth cells high-plp-threshold percent low-plp-threshold percent;
    }
  }
  plp-to-clp;
  scheduler-maps map-name {
    forwarding-class class-name {
      epd-threshold cells plp1 cells;
      linear-red-profile profile-name;
      priority (high | low);
      transmit-weight (cells number | percent number);
    }
  }
}
```

```

    }
    vc-cos-mode (alternate | strict);
  }
}
per-unit-scheduler;
shared-scheduler;
schedulers number;
unit logical-unit-number {
  atm-scheduler-map (map-name | default);
  copy-tos-to-outer-ip-header;
  family family {
    address address {
      destination address;
    }
    filter {
      input filter-name;
      output filter-name;
    }
    policer {
      input policer-name;
      output policer-name;
    }
    simple-filter {
      input filter-name;
    }
  }
}
layer2-policer {
  input-policer policer-name;
  input-three-color policer-name;
  output-policer policer-name;
  output-three-color policer-name;
}
plp-to-clp;
shaping {
  (cbr rate | rtvbr peak rate sustained rate burst length |vbr peak rate sustained rate
  burst length);
}
vci vpi-identifier.vci-identifier;
}
}

```

On the Juniper Networks MX Series 3D Universal Edge Routers with Enhanced Queuing DPCs and on M Series and T Series routers with IQ2E PIC, you can configure the following CoS statements at the [edit interfaces] hierarchy level:

```

hierarchical-scheduler;
interface-set interface-set-name {
  ethernet-interface-name {
    [interface-parameters];
  }
}
}

```

filter (Applying to an Interface)

Syntax	<pre>filter { input <i>filter-name</i>; output <i>filter-name</i>; }</pre>
Hierarchy Level	[edit interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family <i>family</i>], [edit logical-systems <i>logical-system-name</i> interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family <i>family</i>]
Release Information	Statement introduced before Junos OS Release 7.4.
Description	Apply a filter to an interface. You can also use filters for encrypted traffic. When you configure filters, you can configure the family inet , inet6 , mpls , or vpls only.
Options	<p>input <i>filter-name</i>—Name of one filter to evaluate when packets are received on the interface.</p> <p>output <i>filter-name</i>—Name of one filter to evaluate when packets are transmitted on the interface.</p>
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• simple-filter on page 41• Applying Firewall Filter Tricolor Marking Policers to Interfaces• Example: Classifying Packets Based on Their Destination Address on page 13• Example: Configuring and Verifying a Complex Multifield Filter on page 14• Example: Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets on page 17• Example: Configuring a Simple Filter on page 19• Example: Configuring a Logical Bandwidth Policer on page 20• Example: Two-Color Policers and Shaping Rate Changes on page 21

simple-filter (Applying to an Interface)

Syntax	<code>simple-filter { input <i>filter-name</i>; }</code>
Hierarchy Level	[edit interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family inet], [edit logical-systems <i>logical-system-name</i> interfaces <i>interface-name</i> unit <i>logical-unit-number</i> family inet]
Release Information	Statement introduced in Junos OS Release 7.6.
Description	Apply a simple filter to an interface. You can apply simple filters to the family inet only, and only in the input direction.
Options	input <i>filter-name</i> —Name of one filter to evaluate when packets are received on the interface.
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• Configuring Multifield Classifiers on page 9• filter on page 40

PART 3

Index

- [Index on page 45](#)

Index

Symbols

#, comments in configuration statements.....	x
(), in syntax descriptions.....	x
< >, in syntax descriptions.....	x
[], in configuration statements.....	x
{ }, in configuration statements.....	x
(pipe), in syntax descriptions.....	x

B

braces, in configuration statements.....	x
brackets	
angle, in syntax descriptions.....	x
square, in configuration statements.....	x

C

classification	
multifield	
example configuration.....	14, 17
Layer 3 VPN.....	17
VoIP.....	14
VRF.....	17
comments, in configuration statements.....	x
conventions	
text and syntax.....	ix
CoS	
multifield classifier.....	3
simple filter rules.....	4
curly braces, in configuration statements.....	x
customer support.....	xi
contacting JTAC.....	xi

D

documentation	
comments on.....	xi
dscp statement	
usage guidelines.....	17

E

example	
multifield classifier for destination address.....	13
policers and shaping rate changes.....	21

F

family statement	
multifield classifier	
usage guidelines.....	9
multifield classifiers.....	32
filter statement	
firewall.....	33
font conventions.....	ix
from statement	
CoS.....	34

I

IEEE 802.1p inheritance	
swap-by-poppush statement.....	10
transparent statement.....	10
transparent tag.....	10

L

Layer 3 VPN	
multifield classification.....	17
logical bandwidth policer	
example.....	20
logical-bandwidth-policer statement	
usage guidelines.....	10

M

manuals	
comments on.....	xi
multifield classifier	
CoS.....	3
example configuration for destination	
address.....	13
multifield classifiers	
limitation on M Series routers.....	22

O

output filters	
limitation on M Series routers.....	22

P

parentheses, in syntax descriptions.....	x
policers	
and shaping rate changes.....	5, 21

S

shaping rate	
changes and policers.....	5, 21
simple filter	
rules.....	4

simple-filter statement	
interfaces.....	41
usage guidelines.....	19
support, technical See technical support	
swap-by-poppush statement.....	5, 10
syntax conventions.....	ix

T

technical support	
contacting JTAC.....	xi
term statement	
firewall	
simple filter.....	37
then statement	
CoS.....	38
transparent statement.....	5, 10
CoS.....	29

V

VLAN tagging	
swap-by-poppush statement.....	5
transparent statement.....	5
VoIP traffic classification.....	14