

## Chapter 26

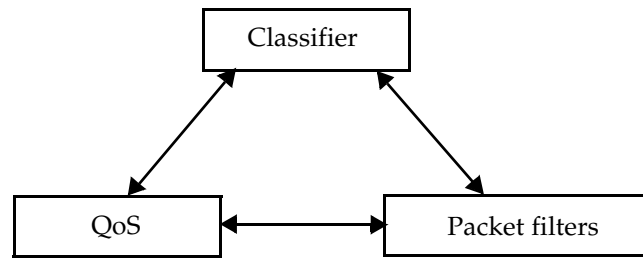
# Generic Packet Classifier

Introduction .....	26-2
Classification Functions .....	26-2
Configuration of Classifiers .....	26-3
Command Reference .....	26-4
create classifier .....	26-4
destroy classifier .....	26-15
set classifier .....	26-16
show classifier .....	26-24

## Introduction

Two generic software features, *Generic Packet Classifier* and *Quality of Service (QoS)*, are used on the switch. The Generic Packet Classifier (Classifier) performs packet classification, and QoS (Quality of Service) performs Layer 2 priority and queuing operations. Refer to [Chapter 27, Quality of Service \(QoS\)](#) for more information on Quality of Service. This chapter deals with the Classifier.

Figure 26-1: Interaction between the Classifier and QoS software



The Classifier defines packet-matching rules that classify packets into *data flows*. A data flow is a categorisation of packets that obey a predefined rule and are processed in a similar manner. For example, all packets with the same destination TCP/IP port may be defined to form a flow (e.g. Telnet or HTTP traffic). These packet-matching rules are referred to as *classifiers* on the command line.

Having defined packet-matching rules in the Classifier, it is up to the other software features to specify what to do with a packet when it matches a rule. You must create an association between the rule in the Classifier and an action elsewhere. To create an association see [Chapter 27, Quality of Service \(QoS\)](#) and [“Classifier-Based Packet Filters” on page 7-39 of Chapter 7, Switching](#).

For a Layer 3 switch handling packets internally within hardware, the result of the match will correspond to the actions available in the switching chipset. For a Layer 3 switch handling packets within the CPU, the result of the match will correspond to actions defined elsewhere. For example, classifiers may be incorporated into a sequence of hardware packet filters, to provide Access Control List (ACL) functionality (see [“Access Control Lists \(ACLs\)” on page 7-40 of Chapter 7, Switching](#)).

## Classification Functions

The following table explains key functions that packet classification performs.

Function	Description
Traffic containment	Contains traffic within the VLAN. Packets can be classified by protocol, subnet or application, which controls where the packets can go on the network.
Traffic filtering	Filters traffic so that required traffic goes to the VLAN. Traffic can be filtered by protocol, IP address, and applications, such as HTTP or SMTP.
Traffic security	Similar benefits to filtering. Specific IP addresses can be made secure to prevent “hackers”.
Traffic quality of service	Prioritises frames based on their classification. For instance, voice over IP traffic could be given a higher priority than Web traffic.

## Configuration of Classifiers

Configuring the classifier involves creating a set of packet-matching rules, called *classifiers*, using the command:

```
create classifier=1..9999 [options]
```

These classifiers can identify any single packet based on criteria such as those described in the following table.

Criteria	Description
Ethernet encapsulation type	Packets are classified depending on the specific protocol type of each frame. Different values indicate how the packet is formatted.
Source/destination MAC address	All frames from a specific source or destination MAC address are classified to the same VLAN and/or priority. This classification can be used for users on remote networks.
Source VLAN	Frames are classified based on the name of the source destination VLAN.
Layer 3 protocols	Frames are classified based on any value for Layer 3 protocols. The switch can match on all IP packets irrespective of the exact type of Ethernet encapsulation. Layer 3 protocol and Ethernet encapsulation types are interrelated.
Source/destination IP address	Frames are classified based on an exact match of the source or destination IP address information within the IP header of each frame.
Source/destination IPv6 address	For accelerated IPv6 traffic, frames are classified based on an exact match of the source or destination IPv6 address information within the IPv6 header of each frame. <b>Valid for x900-48FE and AT-9900 switches only</b>
IP TOS or DiffServ Code Point (DSCP)	Frames are classified based on an exact match of the Type of Service (TOS) or DSCP bits within the IP header of the frame. These values can be used by applications to indicate the priority and Quality of Service (QoS) for each frame.
Layer 4 protocol (TCP/UDP, etc.)	Frames are classified based on specific Layer 4 TCP or UDP destination and source port numbers contained within the header of an IP frame.
Layer 4 source/destination port	Frames are classified based on a specific port number or range.
Layer 5 byte	Frames are classified based on specific values in the Layer 5 part of IP packets.

## Command Reference

This section describes the commands available to configure and manage the Classifiers.

The shortest valid command is denoted by capital letters in the Syntax section. See [“Conventions” on page xlix of About this Software Reference](#) in the front of this manual for details of the conventions used to describe command syntax. See [Appendix A, Messages](#) for a complete list of messages and their meanings.

### create classifier

Classifier parameters are sorted in approximately the order the fields appear in the packet.

**Syntax:** For non-IPv6 traffic:  
**non-IPv6 traffic**

```
CREate CLASSifier=rule-id
    [MACSaddr={macadd|ANY|DHCP Snooping}] [MACDaddr={macadd|
    ANY}] [MACSMask=macadd] [MACDMask=macadd]
    [MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
    [TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
    [VLAN={vlanname|1..4094|ANY}] [INNERTPid={tpid|ANY}]
    [INNERVLANPriority={0..7|ANY}]
    [INNERVLANId={vlanname|1..4094|ANY}]
    [ETHFormat={802.2-Tagged|802.2-Untagged|ETHII-Tagged|
    ETHII-Untagged|NETWARERAW-Tagged|Netwareraw-untagged|
    SNAP-Tagged|SNAP-Untagged|ANY}]
    [PROTocol={protocoltype|IP|ANY}]
    [IPDScp={dscplist|ANY}] [IPTOs={0..7|ANY}]
    [IPSAddr={ipaddmask|ANY|DHCP Snooping}]
    [IPDAddr={ipaddmask|ANY}]
    [IPPRotocol={TCP|UDP|ICMp|IGMp|OSPf|ipprotocolnum|ANY}]
    [IPXDAddr={ipxadd|ANY}]
    [IPXDSocket={NCP|SAP|RIP|NNB|DIAG|NLSp|IPXwan|
    ipxsocketnum|ANY}]
    [IPXS Socket={NCP|SAP|RIP|NNB|DIAG|NLSp|IPXwan|
    ipxsocketnum|ANY}]
    [TCPSport={portid|port-range|ANY}]
    [TCPDport={portid|port-range|ANY}]
    [UDPSport={portid|port-range|ANY}]
    [UDPDport={portid|port-range|ANY}]
    [L4SMask=mask] [L4DMask=mask]
    [L5BYTE01=byteoffset,bytevalue[,bytemask]]
    [L5BYTE02=byteoffset,bytevalue[,bytemask]]
    [L5BYTE03=byteoffset,bytevalue[,bytemask]]
    [L5BYTE04=byteoffset,bytevalue[,bytemask]]
    [L5BYTE05=byteoffset,bytevalue[,bytemask]]
    [L5BYTE06=byteoffset,bytevalue[,bytemask]]
    [L5BYTE07=byteoffset,bytevalue[,bytemask]]
    [L5BYTE08=byteoffset,bytevalue[,bytemask]]
    [L5BYTE09=byteoffset,bytevalue[,bytemask]]
    [L5BYTE10=byteoffset,bytevalue[,bytemask]]
    [L5BYTE11=byteoffset,bytevalue[,bytemask]]
    [L5BYTE12=byteoffset,bytevalue[,bytemask]]
```

```
[L5BYTE13=byteoffset,bytevalue[,bytemask]]
[L5BYTE14=byteoffset,bytevalue[,bytemask]]
[L5BYTE15=byteoffset,bytevalue[,bytemask]]
[L5BYTE16=byteoffset,bytevalue[,bytemask]]
[TCPFlags={ {Urg|Ack|Rst|Syn|Fin}[,...] |ANY}]
[ICMptype={Any|ECHOReply|Unreachable|Quench|Redirect|
ECHO|Advertisement|Solicitation|Timeexceed|Parameter|
TSTAMP|TSTAMPReply|INFOREQ|INFOREP|ADDRREQ|ADDRREP|
NAMEREQ|NAMERPLY|icmp-type}]
[ICMPCode={Any|Filter|FRAGMENT|FRAGReasm|HOSTComm|
HOSTIsolated|HOSTPrec|HOSTRedirect|HOSTRTos|HOSTTos|
HOSTUNKNOWN|HOSTUNReach|NETComm|NETRedirect|NETRTos|
NETTos|NETUNKNOWN|NETUNReach|NOPtr|Portunreach|
PREcedent|PROtunreach|PTRproblem|Sourceroute|Ttl|
icmp-code}]
[IGMptype={ANY|Query|V1Report|DVmrp|PIMv1|CTRace|
V2Report|V2Leave|MCTRACEResponse|MCTRACE|V3Report|
MRAdvert|MRSolicit|MRTermination|igmp-type}]
[EIPBYTE01=byteoffset,bytevalue[,bytemask]]
[EIPBYTE02=byteoffset,bytevalue[,bytemask]]
[EIPBYTE03=byteoffset,bytevalue[,bytemask]]
[EIPBYTE04=byteoffset,bytevalue[,bytemask]]
[EIPBYTE05=byteoffset,bytevalue[,bytemask]]
[EIPBYTE06=byteoffset,bytevalue[,bytemask]]
[EIPBYTE07=byteoffset,bytevalue[,bytemask]]
[EIPBYTE08=byteoffset,bytevalue[,bytemask]]
[EIPBYTE09=byteoffset,bytevalue[,bytemask]]
[EIPBYTE10=byteoffset,bytevalue[,bytemask]]
[EIPBYTE11=byteoffset,bytevalue[,bytemask]]
[EIPBYTE12=byteoffset,bytevalue[,bytemask]]
[EIPBYTE13=byteoffset,bytevalue[,bytemask]]
[EIPBYTE14=byteoffset,bytevalue[,bytemask]]
[EIPBYTE15=byteoffset,bytevalue[,bytemask]]
[EIPBYTE16=byteoffset,bytevalue[,bytemask]]
```

**Syntax:** For IPv6 traffic that is switched at Layer 2:  
**IPv6 traffic switched at Layer 2**

```
CREate CLASSifier=rule-id
  ETHFormat={ETHII-Tagged|ETHII-Untagged} PROToCol=IPV6
  [MACSaddr={macadd|ANY}]
  [MACDaddr={macadd|ANY}] [MACDMask=macadd]
  [MACSMask=macadd]
  [MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
  [TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
  [VLAN={vlanname|1..4094|ANY}] [INNERTpId={tpid|ANY}]
  [INNERVLANPriority={0..7|ANY}]
  [INNERVLANId={vlanname|1..4094|ANY}]
  [IPDScp={dscplist|ANY}]
  [IPProtocol={TCP|UDP|ICMp|IGMp|OSpf|ipprotocolnum|ANY}]
```

**Syntax:** For accelerated IPv6 traffic, when applied on the Layer 2 processor of the accelerator (see [Figure 27-5 on page 27-10 of Chapter 27, Quality of Service \(QoS\)](#)):

```
CREate CLASSifier=rule-id ETHFormat=ETHII-Tagged
  PROTOcol=IPV6 [MACSaddr={macadd|ANY}]
  [MACDaddr={macadd|ANY}] [MACDMask=macadd]
  [MACSMask=macadd]
  [MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
  [TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
  [VLAN={vlanname|1..4094|ANY}] [INNERTPID={tpid|ANY}]
  [INNERVLANPriority={0..7|ANY}]
  [INNERVLANId={vlanname|1..4094|ANY}]
  [IPDScp={dscplist|ANY}]
  [IPPRotocol={TCP|UDP|ICMp|IGMp|OSPf|ipprotocolnum|ANY}]
```

**Syntax:** For accelerated IPv6 traffic, when applied on the Layer 3 processor of the accelerator (see [Figure 27-5 on page 27-10 of Chapter 27, Quality of Service \(QoS\)](#)):

```
CREate CLASSifier=rule-id [ETHFormat={ETHII-Tagged|ANY}]
  [PROTOcol=IPV6] [IPDScp={0..63|ANY}]
  [IPSAddr={ipv6-add/prefix-length|ANY}]
  [IPDAddr={ipv6-add/prefix-length|ANY}]
  [IPPRotocol={TCP|UDP|ICMp|IGMp|OSPf|ipprotocolnum|ANY}]
  [TCPSport={portid|port-range|ANY}]
  [TCPDport={portid|port-range|ANY}]
  [UDPSport={portid|port-range|ANY}]
  [UDPDPport={portid|port-range|ANY}]
  [L4SMask=mask] [L4DMask=mask]
```

where:

- *rule-id* is a decimal number from 1 to 9999.
- *macadd* is an Ethernet six-octet MAC address, expressed as six pairs of hexadecimal digits delimited by hyphens.
- *tpid* is a 2-byte hexadecimal number.
- *vlan-name* is a unique name from 1 to 32 characters. Valid characters are uppercase and lowercase letters, digits (0-9), the underscore character, and the hyphen. The *vlan-name* cannot be a number, **all**, or **any**.
- *protocoltype* is either a valid protocol number or a recognised protocol name. A protocol number can be either 1 byte for SAP, 2 bytes for ETHII or 5 bytes for an 802.3/802.2 SNAP type packet, and is specified in hexadecimal.
- *dscplist* is a single number or a group of numbers, either a comma separated list, a range (specified as n-m) or a combination of the two. Numbers start at 0 and end at 63.
- *ipprotocolnum* is either an IP protocol number or a recognised IP protocol name. An IP protocol number is expressed as a 1 byte decimal number.
- *ipaddmask* is an IP address in dotted decimal notation with an optional mask. The optional mask is specified by adding “/M” following the IP address, where M is the number of contiguous bits in the mask; M ranges from 0 to 32 inclusive.
- *ipv6-add* is a valid IPv6 address, with its prefix length indicated by slash notation.
- *prefix-length* is an integer from 1 to 128.

- *ipxaddr* is an IPX network address expressed as a 4 byte hexadecimal number.
- *ipxsocketnum* is either an IPX socket number or a recognised IPX socket type. An IPX socket number is expressed as a 2 byte hexadecimal number.
- *portid* is a TCP/IP or UDP/IP port number.
- *port-range* is a hyphen-separated range of TCP/IP or UDP/IP ports, such as 5550-5554.
- *mask* is a 2-byte hexadecimal number in the range 0001 to ffff.
- *byteoffset* is a decimal number:
  - For L5BYTExx, the range is 0 to 37
  - For EIPBYTExx, the range is 0 to 65
- *bytevalue* is a 2-digit hexadecimal number.
- *bytemask* is a 2-digit hexadecimal number.
- *icmp-type* is a decimal number in the range 0 to 255.
- *icmp-code* is a decimal number in the range 0 to 255.
- *igmp-type* is a 2-digit hexadecimal number.

**Description** This command creates a packet matching rule that identifies a particular data flow. This data flow may be general or specific, for example, IP packets with a particular TCP destination port is specific.

The **classifier** parameter specifies a unique rule ID for the packet matching rule. It does not imply an order between this rule and rules previously created.

The **macsaddr** parameter specifies the source MAC address of the packet. The **dhcpsnooping** option applies the classifier to entries in the DHCP snooping binding database. The default is **any**.

The **macdaddr** parameter specifies the destination MAC address of the packet. The default is **any**.

The **macdmask** and **macsmask** parameters specify masks to be used on the **macdaddr** and **macsaddr** parameters respectively. When a bit is set to 1 in the mask, the value of the bit at the same position in the byte value of the MAC address is used to determine a match. If a bit in either of the **macdmask** or **macsmask** parameters is 0, the corresponding bit in the **macdaddr** or **macsaddr** parameters is ignored. The default is **ff-ff-ff-ff-ff-ff**, which means the classifier matches against all bits in the MAC address.

The **mactype** parameter specifies whether the packet is a Layer 2 unicast (**l2ucast**), Layer 2 multicast (**l2mcast**) or Layer 2 broadcast Ethernet (**l2bcast**). The default is **any**, which indicates that the classifier does not match on MAC type.

The **tpid** parameter specifies the Tag Protocol Identifier field in the packet. The default is **any**.

The **vlanpriority** parameter specifies the 802.1P priority in the VLAN tag. The default is **any**.

The default VLAN is **any**. The **vlan** parameter specifies:

- For IPv4 packets and switched Layer 2 IPv6 packets, the source VLAN that the packet was associated with at the time of classification

- For accelerated IPv6 packets at the Layer 2 processor of the accelerator card, the destination VLAN (see [Figure 27-5 on page 27-10](#) of [Chapter 27, Quality of Service \(QoS\)](#))

The **innertpid**, **innervlanpriority**, and **innervlanid** parameters are for use on double tagged traffic. *Inner* refers to the second 802.1Q tag in the packet. This is the one being tunnelled. The following table shows where the inner and outer tags are matched in the packet. Matching outer parameters on customer ports is not possible.

	Outer VLAN parameters (normal)	Inner VLAN parameters
Customer port	Only VLAN ID.	1st tag
Core port	1st tag	2nd tag
Nested VLANs disabled	1st tag	2nd tag

If you attach the classifier to a hardware filter and nested VLANs are not being used, it is assumed that all incoming packets are double tagged. When nested VLANs are being used, outer parameters also cannot be used to match on customer ports. If the classifier is being attached to a number of ports, they are all be treated like core ports if even one port is a core port.

The **innertpid** parameter specifies the TPID in the second 802.1Q tag in the packet. The default is **any**.

The **innervlanpriority** parameter specifies the second 802.1P field in the packet. The default is **any**.

The **innervlanid** parameter specifies the tunnelled VLAN ID in the second 802.1Q tag in the packet. The default is **any**.

The **ethformat** parameter specifies the Ethernet encapsulation type of the packet. Parameter values and encapsulation types are explained in the following table.

Parameter Value	How the Packet is Formatted	Encapsulation Type
802.2	According to IEEE Standards 802.2 and 802.3 with a DSAP/SSAP (Destination Service Access Point/Source Service Access Point) value not equal to hexadecimal AAAA.	SAP
ETHII	According to RFC 894, <i>Standard for the transmission of IP datagrams over Ethernet networks</i> .	Ethernet II
SNAP	According to IEEE Standards 802.2 and 802.3 and RFC 1042, <i>Standard for the transmission of IP datagrams over IEEE 802 networks</i> .	SNAP

If an **ethformat** option is specified (excluding **any**), the **protocol** parameter must also be used. Specifying **ethformat=snap protocol=ip** means using **protocol=0800**. Specifying **ethformat=snap protocol=ipx** means using **protocol=8137**. Specifying **ethformat=802.2 protocol=ipx** means using **protocol=E0**. [Table 26-1 on page 26-12](#) shows the combinations of the **ethformat** and **protocol** parameters, and their implementation in the Classifier.



If you specify **protocol=ipx** you must also specify an **ethformat** other than **any**. Additionally, the IPX header parameters **ipxdaddr**, **ipxdsocket**, and **ipxssocket** are only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**.

Note that if an **ethformat** option is specified, it must include either **tagged** or **untagged**. For example, either **ethii-tagged** or **ethii-untagged**, **netwareraw-tagged** or **netwareraw-untagged**, etc.

For IPv6 packets on x900-24X switches at switched at Layer 2, valid **ethformat** options are **ethii-tagged** and **ethii-untagged**. For accelerated IPv6 packets at the accelerator card's Layer 2 processor, the valid option is **ethii-tagged**.

The **protocol** parameter specifies the protocol of the packet, as shown in [Table 26-2 on page 26-13](#). The Classifier provides a predefined list of common protocols, as shown in [Table 26-3 on page 26-13](#). Three user-specified protocols may be entered for any encapsulation type. If the command specifies a TCP or UDP packet matching rule, then this parameter will default to IP. If the command specifies **protocol=ip**, the IP packets will have either ETHII or SNAP encapsulation (e.g. **ethformat=ethii** or **ethformat=snap**). If the command specifies **protocol=ipx**, the IPX packets may have any of the four types of encapsulation (e.g. **ethformat={802.2|ethii|netwareraw|snap}**). If a ten digit hexadecimal number is specified, e.g. **protocol=xxxxxxabcd** the last four digits (**abcd**) are used for packet classification. The default is **any**.

If you specify **protocol=ipx** you must also specify an **ethformat** other than **any**. Additionally, the IPX header parameters **ipxdaddr**, **ipxdsocket**, and **ipxssocket** are only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**.

The **ipdscp** parameter specifies the Code Point bits of the DiffServ field of an IP packet (from 0 to 63, or **any**). This parameter cannot be specified in conjunction with the **iptos** parameter. An IPv6 classifier accepts only a single value for this parameter. The default is **any**.

The **iptos** parameter specifies the value of the precedence field within the TOS byte of an IP packet. This parameter cannot be specified in conjunction with the **ipdscp** parameter, and cannot be used in classifiers with a **protocol** of IPV6. The default is **any**.

The **ipprotocol** parameter specifies a Layer 4 IP protocol of an IP packet. For IPv6 packets, the **ipprotocol** parameter matches against the Next Header field of the IPv6 packet header. If the command specifies a TCP/IP packet matching rule, (e.g. **tcpdport** is specified), then the default value for this parameter is **tcp**. If the command specifies a UDP/IP packet matching rule, (e.g. the **udpport** parameter is specified), then the default value for this parameter is **udp**.

The **ipsaddr** parameter specifies the source IP address (either host or subnet) of an IP packet. If you enter an IPv6 address, **protocol** is set to IPV6. The **dhcpsnooping** option applies the classifier to entries in the DHCP snooping binding database. The default is **any**.

The **ipdaddr** parameter specifies the destination IP address (either host or subnet) of an IP packet. If you enter an IPv6 address, **protocol** is set to IPV6. The default is **any**.

The **ipxdaddr** parameter specifies the destination network address of an IPX packet. This parameter is only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**. The default is **any**.

The **ipxdssocket** specifies the destination IPX socket number of an IPX packet. This parameter is only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**. The default is **any**.

The **ipxssocket** specifies the source IPX socket number of an IPX packet. This parameter is only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**. The default is **any**.

The **tcpdport** parameter specifies the TCP destination port of a TCP/IP packet, or a range of destination ports. The classifier matches all packets that have the specified destination TCP port or that have a destination TCP port in that range. The **l4dmask** parameter is invalid if you specify a range. The default is **any**.

The **tcpsport** parameter specifies the TCP source port of a TCP/IP packet, or a range of source ports. The classifier matches all packets that have the specified source TCP port or that have a source TCP port in that range. The **l4smask** parameter is invalid if you specify a range. The default is **any**.

The **udpdport** parameter specifies the UDP destination port of an UDP/IP packet, or a range of destination ports. The classifier matches all packets that have the specified destination UDP port or that have a destination UDP port in that range. The **l4dmask** parameter is invalid if you specify a range. The default is **any**.

The **udpsport** parameter specifies the UDP source port of an UDP/IP packet, or a range of source ports. The classifier matches all packets that have the specified source UDP port or that have a source UDP port in that range. The **l4smask** parameter is invalid if you specify a range. The default is **any**.

The **l4dmask** parameter specifies a 2-byte hexadecimal mask to match a range of TCP/UDP destination ports in the packet, when you also specify a single TCP or UDP destination port number. The default is **ffff**.

The **l4smask** parameter specifies a 2-byte hexadecimal mask to match a range of TCP/UDP source ports in the packet, when you also specify a single TCP or UDP source port number. The default is **ffff**.

The **l5byte01** to **l5byte16** parameters each specify the properties of a single byte field to match in the Layer 5 part of IP packets, which is the TCP or UDP payload. For each byte field you want to match, specify:

- *byteoffset*, which is a decimal number in the range 0 to 37. This specifies the location of the byte to match. It refers to the offset from the start of Layer 5, after the UDP or TCP header.
- *bytevalue*, which is a 2-digit hexadecimal number. This specifies the value of the byte at the position in the frame that is determined by *byteoffset*. The classifier matches packets that have this value at this location
- (optionally) *bytemask*, which is a 2-digit hexadecimal number. This specifies an eight-bit binary mask to apply to the field. When a bit is set to **1** in the mask, the value of the bit at the same position in the byte value is used to determine a match. A **0** in the mask means that the corresponding bit is ignored. The default is **ff**, which means the classifier matches against all bits in the byte.

**Important** The classifier matches Layer 5 bytes that are within the first 80 bytes of the IP packet. The classifier does not match against bytes that are later in the packet, even if they match the classifier's settings.

When you consider packet length, remember that the location of Layer 5 in a frame varies depending on the length of the lower-layer headers. This is because header values such as the VLAN tag can be missing, and header values such as the Ethernet format specification vary in length.

You must use **l5byte01** as the first byte field and you must number additional byte fields sequentially. Each field must have a greater offset than the fields that precede it.

The **l5byte** parameters match frames with a valid TCP or UDP header, when the network protocol is IP version 4. Therefore, **protocol** defaults to **ip** and does not need to be specified. You also do not have to specify **ipprotocol**, but by default the classifier applies to both TCP and UDP packets.

The **tcpflags** parameter specifies the TCP flags of an IPv4 or IPv6 packet, one or more of **urg**, **ack**, **rst**, **syn** and **fin**. If **any** is specified, TCP flags are ignored. The default is **any**.

The **icmptype** parameter specifies the ICMP type of an IPv4 packet. This can be one of the list of available options, or a decimal value in the range 0 to 255. The **icmptype** parameter is valid only if the **ipprotocol** parameter has either not been specified, or **ipprotocol=icmp** has been specified. If **any** is specified, the ICMP type is ignored. The default is **any**.

The **icmpcode** parameter specifies the ICMP code of an IPv4 packet. This can be one of the list of available options, or a decimal value in the range 0 to 255. The **icmpcode** parameter is valid only if the **ipprotocol** parameter has either not been specified, or **ipprotocol=icmp** has been specified. If **any** is specified, the ICMP code is ignored. The default is **any**.

The **igmp** parameter specifies the IGMP type of an IPv4 packet. This can be one of the list of available options, or a hexadecimal value in the range of 00 to ff. The **igmp** parameter is valid only if the **ipprotocol** parameter has either not been specified, or **ipprotocol=igmp** has been specified. If **any** is specified, the IGMP type is ignored. The default is **any**.

The **eipbyte01** to **eipbyte16** parameters each specify the properties of a single byte field to match in the Layer 3 header and data of a non-IPv4 and non-IPv6 packet. The **eipbyte01** parameter must be used as the first byte field, and additional byte fields must increment sequentially, for example **eipbyte01**, **eipbyte02**, **eipbyte03**. Each field must have a greater offset than the field that precedes it.

For each byte field you want to match, specify a *byteoffset* and a *bytevalue*, and optionally, a *bytemask*.

- *byteoffset* is a decimal number in the range 0 to 65. This specifies the location of the byte to match. It refers to the offset from the start of Layer 3, after the Layer 2 encapsulation format of an Ethernet frame.
- *bytevalue* is a 2-digit hexadecimal number. This specifies the value of the byte at the frame position determined by the *byteoffset*. The classifier matches packets that have this value at this location.
- (optional) *bytemask* is a 2-digit hexadecimal number. This specifies an eight-bit binary mask to apply to the field. When a bit is set to 1 in the mask, the value of the bit at the same position in the byte is used to determine a match. If the *bytemask* is 0, the corresponding bit is ignored. The default is ff, which means the classifier matches against all bits in the byte.

**Important** The classifier matches Layer 3 bytes that are within the first 80 bytes of the non-IPv4 and non-IPv6 packet. The classifier does not match against bytes that are later in the packet, even if they match the classifier's settings. When you consider packet length, remember that the location of Layer 3 in a frame varies depending on the encapsulation length of the Ethernet frame format. This is because header values such as the VLAN tag can be missing.

Table 26-1: Available **ethformat** and **protocol** parameter combinations

ETHFORMAT=	PROTOCOL=	CLASSIFIER	ASIC Chip
ETHII	[not specified]	OK	Error
	ANY	OK	Error
	IP	OK (1)	Ok
	IPX	OK (2)	OK
	IPv6	OK	OK
	<i>protocoltype</i>	OK	OK
NETWARERAW	[not specified]	OK (3)	OK
	ANY	OK (3)	OK
	IP	Error	n/a
	IPX	OK (3)	OK
	"IPX 802.3"	OK	OK
	IPv6	Error	n/a
SNAP	[not specified]	OK	Error
	ANY	OK	Error
	IP	OK	OK Protocol=xxxxxx0800
	IPX	OK	OK Protocol=xxxxxx8137
	IPv6	Error	n/a
	<i>protocoltype</i>	OK	OK
802.2	[not specified]	OK	Error
	ANY	OK	Error
	IP	Error	n/a
	IPX	OK (4)	OK
	IPv6	Error	n/a
	<i>protocoltype</i>	OK	OK

#### Key to table

- [not specified] = the **protocol** parameter is not specified on the command line
- (1) = equivalent to specifying **protocol=0800**
- (2) = equivalent to specifying **protocol=8137**
- (3) = equivalent to specifying **protocol="IPX 802.3"**
- (4) = equivalent to specifying **protocol=E0**.

Table 26-2: Protocol values of encapsulated packets

Encapsulation Type	Value
SAP	The value of the DSAP field.
ETHII	The value of the ETYPE field.
NETWARERAW	The value of the IPX checksum field (hexadecimal FFFF).
SNAP	The value of the ETYPE field.

Table 26-3: Predefined protocol types for use in the **protocol** parameter

Protocol Name	Protocol No.	Encapsulation	Enter Min. Characters
SNA Path Control	04	SAP	3
PROWAY-LAN	0E	SAP	7
EIA-RS	4E	SAP	3
PROWAY	8E	SAP	3
IPX 802.2	E0	SAP	9
NetBEUI	F0	SAP	3
ISO CLNS IS	FE	SAP	5
IP ETHII	0800	EthII	8
X.75 Internet	0801	EthII	4
NBS Internet	0802	EthII	3
ECMA Internet	0803	EthII	4
Chaosnet	0804	EthII	4
X.25 Level 3	0805	EthII	4
ARP	0806	EthII	3
XNS Compat	0807	EthII	3
Banyan Systems	0BAD	EthII	3
BBN Simnet	5208	EthII	3
DEC MOP Dump/Ld	6001	EthII	9
DEC MOP Rem Cons	6002	EthII	9
DEC DECNET	6003	EthII	7
DEC LAT	6004	EthII	7
DEC Diagnostic	6005	EthII	7
DEC Customer	6006	EthII	7
DEC LAVC	6007	EthII	7
RARP	8035	EthII	4
DEC LANBridge	8038	EthII	7
DEC Encryption	803D	EthII	7
AppleTalk	809B	EthII	3
IBM SNA	80D5	EthII	7
IPX EthII	8137	EthII	9
AppleTalk AARP	80F3	EthII	11
SNMP	814C	EthII	4

Table 26-3: Predefined protocol types for use in the **protocol** parameter (cont.)

Protocol Name	Protocol No.	Encapsulation	Enter Min. Characters
IPv6 EthII	86DD	EthII	10
IPX 802.3	FFFF	NetWare 802.3 Raw	9
ETHERTALK 2	080007809B	SNAP	11
ETHERTALK 2 AARP	00000080F3	SNAP	13
IPX SNAP	0000008137	SNAP	8

Note: When you enter a protocol name that contains spaces, enclose the name in double quotation marks. You can use lowercase or uppercase letters. For example, to specify ETHERTALK 2 AARP, enter **protocol="ethertalk 2 aarp"** or **protocol="ethertalk 2 a"**.

**Examples** To create a packet matching rule that matches all IP packets from the IP subnet 192.168.100.2 (mask=255.255.255.0) with a destination TCP port of 23, use one of the commands:

```
cr class=1 ipsa=192.168.100.2/24 tcpd=23
cr class=1 mact=any prot=ip ipsa=192.168.100.2/24 tcpd=23
```

To create a packet matching rule that matches all other IP packets with a destination TCP port of 23, use one of the commands:

```
cr class=2 tcpd=23
cr class=2 ipsa=any tcpd=23
cr class=2 ipda=any ipsa=any tcpd=23
```

These two classifiers can be used in combination in hardware filters or QoS flow groups to separate Telnet traffic from this subnet from other Telnet traffic.

To create classifier 10 which selects all packets with a destination TCP port in the range 5550 to 5554, use the command:

```
cr class=10 tcpd=5550-5554
```

To create classifier 10 to match DHCP snooping entries, use any of the commands:

```
create classifier=10 ipsa=dhcps
create classifier=10 macs=dhcps
create classifier=10 ipsa=dhcps macs=dhcps
```

**Related Commands** [destroy classifier](#)  
[set classifier](#)  
[show classifier](#)

## destroy classifier

---

**Syntax** DESTroy CLASSifier={*rule-list*|ALL}

where *rule-list* is a single rule ID or a group, either a comma-separated list, a range (specified as *n-m*), or a combination of the two. Rule IDs start at 1.

**Description** This command destroys one or more packet matching rules.

The **classifier** parameter specifies the ID of an existing packet matching rule. This rule cannot be associated with any action in another software module. If **all** is specified, then all packet matching rules are destroyed.

**Examples** To destroy the packet matching rules with rule IDs 3, 5 and 9 to 12, use the command:

```
dest class=3,5,9-12
```

To destroy all packet matching rules, use the command:

```
dest class=all
```

**Related Commands** [create classifier](#)  
[show classifier](#)

## set classifier

Classifier parameters are sorted in approximately the order the fields appear in the packet.

**Syntax:** For non-IPv6 traffic:  
**non-IPv6 traffic**

```
SET CLASSifier=rule-id
  [MACSaddr={macadd|ANY|DHCP Snooping}] [MACDaddr={macadd|
  ANY}] [MACSMask=macadd] [MACDMask=macadd]
  [MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
  [TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
  [VLAN={vlannname|1..4094|ANY}] [INNERTPid={tpid|ANY}]
  [INNERVLANPriority={0..7|ANY}]
  [INNERVLANId{vlannname|1..4094|ANY}]
  [ETHFormat={802.2-Tagged|802.2-Untagged|ETHII-Tagged|
  ETHII-Untagged|NETWARERAW-Tagged|Netwareraw-untagged|
  SNAP-Tagged|SNAP-Untagged|ANY}]
  [PROTOCOL={protocoltype|IP|IPX|ANY}]
  [IPDScp={dscplist|ANY}] [IPTOs={0..7|ANY}]
  [IPSAddr={ipaddmask|ANY|DHCP Snooping}]
  [IPDAddr={ipaddmask|ANY}]
  [IPProtocol={TCP|UDP|ICMP|IGMP|OSPF|ipprotocolnum|ANY}]
  [IPXDAddr={ipxadd|ANY}]
  [IPXDSocket={NCP|SAP|RIP|NNB|DIAG|NLSp|IPXwan|
ipxsocketnum|ANY}]
  [IPXSSocket={NCP|SAP|RIP|NNB|DIAG|NLSp|IPXwan|
ipxsocketnum|ANY}]
  [TCPSport={portid|port-range|ANY}]
  [TCPDport={portid|port-range|ANY}]
  [UDPSport={portid|port-range|ANY}]
  [UDPDPport={portid|port-range|ANY}]
  [L4SMask=mask] [L4DMask=mask]
  [L5BYTE01=byteoffset,bytevalue[,bytemask]]
  [L5BYTE02=byteoffset,bytevalue[,bytemask]]
  [L5BYTE03=byteoffset,bytevalue[,bytemask]]
  [L5BYTE04=byteoffset,bytevalue[,bytemask]]
  [L5BYTE05=byteoffset,bytevalue[,bytemask]]
  [L5BYTE06=byteoffset,bytevalue[,bytemask]]
  [L5BYTE07=byteoffset,bytevalue[,bytemask]]
  [L5BYTE08=byteoffset,bytevalue[,bytemask]]
  [L5BYTE09=byteoffset,bytevalue[,bytemask]]
  [L5BYTE10=byteoffset,bytevalue[,bytemask]]
  [L5BYTE11=byteoffset,bytevalue[,bytemask]]
  [L5BYTE12=byteoffset,bytevalue[,bytemask]]
  [L5BYTE13=byteoffset,bytevalue[,bytemask]]
  [L5BYTE14=byteoffset,bytevalue[,bytemask]]
  [L5BYTE15=byteoffset,bytevalue[,bytemask]]
  [L5BYTE16=byteoffset,bytevalue[,bytemask]]
  [TCPFlags={{Urg|Ack|Rst|Syn|Fin}[,...]|ANY}]
  [ICMptype={Any|ECHO Rply|Unreachable|Quench|Redirect|
  ECHO|Advertisement|Solicitation|Timeexceed|Parameter|
  TSTAMP|TSTAMP Rply|INFOREQ|INFOREP|ADDRREQ|ADDRREP|
  NAMEREQ|NAMERPLY|icmp-type}]
  [ICMPCode={Any|Filter|FRAGMENT|FRAGReassm|HOSTComm|
  HOSTIsolated|HOSTPrec|HOSTREdirect|HOSTRTos|HOSTTos|
  HOSTUNKnown|HOSTUNReach|NETComm|NETREdirect|NETRTos|
  NETTos|NETUNKnown|NETUNReach|NOptr|Portunreach|
```



```

PREcedent | PROtunreach | PTRproblem | Sourceroute | Ttl |
icmp-code} ]
[ IGmptype={ ANY | QUery | V1Report | DVmrp | PIMv1 | CTRace |
V2Report | V2Leave | MCTRACEResponse | MCTRACE | V3Report |
MRAdvert | MRSolicit | MRTermination | igmp-type} ]
[ EIPBYTE01=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE02=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE03=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE04=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE05=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE06=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE07=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE08=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE09=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE10=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE11=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE12=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE13=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE14=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE15=byteoffset,bytevalue[, bytemask] ]
[ EIPBYTE16=byteoffset,bytevalue[, bytemask] ]

```

**Syntax:**  
**IPv6 traffic switched**  
**at Layer 2**

For IPv6 traffic that is switched at Layer 2:

```

SET CLASSifier=rule-id
ETHFormat={ETHII-Tagged|ETHII-Untagged} PROToCol=IPV6
[MACSaddr={macadd|ANY}]
[MACDaddr={macadd|ANY}] [MACDMask=macadd]
[MACSMask=macadd]
[MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
[TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
[VLAN={vlanname|1..4094|ANY}] [INNERTPid={tpid|ANY}]
[INNERVLANPriority={0..7|ANY}]
[INNERVLANId={vlanname|1..4094|ANY}]
[IPDScp={dscplist|ANY}]
[IPPRotocol={TCP|UDP|ICMp|IGMp|OSpf|ipprotocolnum|ANY}]

```

**Syntax:** For accelerated IPv6 traffic, when applied on the Layer 2 processor of the accelerator (see [Figure 27-5 on page 27-10 of Chapter 27, Quality of Service \(QoS\)](#)):

```
SET CLASSifier=rule-id ETHFormat=ETHII-Tagged
  PROTOcol=IPV6 [MACSaddr={macadd|ANY}]
  [MACDaddr={macadd|ANY}] [MACDMask=macadd]
  [MACSMask=macadd]
  [MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
  [TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
  [VLAN={vlanname|1..4094|ANY}] [INNERTPID={tpid|ANY}]
  [INNERVLANPriority={0..7|ANY}]
  [INNERVLANId={vlanname|1..4094|ANY}]
  [IPDScp={dscplist|ANY}]
  [IPProtocol={TCP|UDP|ICMp|IGMp|OSPf|ipprotocolnum|ANY}]
```

**Syntax:** For accelerated IPv6 traffic, when applied on the Layer 3 processor of the accelerator (see [Figure 27-5 on page 27-10 of Chapter 27, Quality of Service \(QoS\)](#)):

```
SET CLASSifier=rule-id [ETHFormat={ETHII-Tagged|ANY}]
  [PROTOcol=IPV6] [IPDScp={0..63|ANY}]
  [IPSAddr={ipv6-add/prefix-length|ANY}]
  [IPDAddr={ipv6-add/prefix-length|ANY}]
  [IPProtocol={TCP|UDP|ICMp|IGMp|OSPf|ipprotocolnum|ANY}]
  [TCPSport={portid|port-range|ANY}]
  [TCPDport={portid|port-range|ANY}]
  [UDPSport={portid|port-range|ANY}]
  [UDPdport={portid|port-range|ANY}]
  [L4SMask=mask] [L4DMask=mask]
```

where:

- *rule-id* is a decimal number from 1 to 9999.
- *macadd* is an Ethernet six-octet MAC address, expressed as six pairs of hexadecimal digits delimited by hyphens.
- *tpid* is a 2-byte hexadecimal number.
- *vlan-name* is a unique name from 1 to 32 characters. Valid characters are uppercase and lowercase letters, digits (0-9), the underscore character, and the hyphen. The *vlan-name* cannot be a number, **all**, or **any**.
- *protocoltype* is either a valid protocol number or a recognised protocol name. A protocol number can be either 1 byte for SAP, 2 bytes for ETHII or 5 bytes for an 802.3/802.2 SNAP type packet, and is specified in hexadecimal.
- *dscplist* is a single number or a group of numbers, either a comma separated list, a range (specified as n-m) or a combination of the two. Numbers start at 0 and end at 63.
- *ipprotocolnum* is either an IP protocol number or a recognised IP protocol name. An IP protocol number is expressed as a 1 byte decimal number.
- *ipaddmask* is an IP address in dotted decimal notation with an optional mask. The optional mask is specified by adding “/M” following the IP address, where M is the number of contiguous bits in the mask; M ranges from 0 to 32 inclusive.

- *ipv6-add* is a valid IPv6 address, with its prefix length indicated by slash notation
- *prefix-length* is an integer from 1 to 128
- *ipxadd* is an IPX network address expressed as a 4 byte hexadecimal number.
- *ipxsocketnum* is either an IPX socket number or a recognised IPX socket type. An IPX socket number is expressed as a 2 byte hexadecimal number.
- *portid* is a TCP/IP or UDP/IP port number.
- *port-range* is a hyphen-separated range of TCP/IP or UDP/IP ports, such as 5550-5554.
- *mask* is a 2-byte hexadecimal number in the range 0001 to ffff.
- *byteoffset* is a decimal number:
  - For L5BYTExx, the range is 0 to 37
  - For EIPBYTExx, the range is 0 to 65
- *bytevalue* is a 2-digit hexadecimal number.
- *bytemask* is a 2-digit hexadecimal number.
- *icmp-type* is a decimal number in the range 0 to 255.
- *icmp-code* is a decimal number in the range 0 to 255.
- *igmp-type* is a 2-digit hexadecimal number.

**Description** This command sets a packet matching rule that identifies a particular data flow. Data flow can be general or specific, for example, IP packets with a particular TCP destination port is specific.

The **classifier** parameter specifies a unique rule ID for the packet matching rule. It does not imply an order between this rule and rules previously created.

The **macsaddr** parameter specifies the source MAC address of the packet. The **dhcpsnooping** option applies the classifier to entries in the DHCP snooping binding database. The default is **any**.

The **macdaddr** parameter specifies the destination MAC address of the packet. The default is **any**.

The **macdmask** and **macsmask** parameters specify masks to be used on the **macdaddr** and **macsaddr** parameters respectively. When a bit is set to 1 in the mask, the value of the bit at the same position in the byte value of the MAC address is used to determine a match. If a bit in either of the **macdmask** or **macsmask** parameters is 0, the corresponding bit in the **macdaddr** or **macsaddr** parameters is ignored. The default is **ff-ff-ff-ff-ff-ff**, which means the classifier matches against all bits in the MAC address.

The **mactype** parameter specifies whether the packet is a Layer 2 unicast (**l2ucast**), Layer 2 multicast (**l2mcast**) or Layer 2 broadcast Ethernet (**l2bcast**). The default is **any**, which indicates that the classifier does not match on MAC type.

The **tpid** parameter specifies the Tag Protocol Identifier field in the packet. The default is **any**.

The **vlanpriority** parameter specifies the 802.1P priority in the VLAN tag. The default is **any**.

The default VLAN is **any**. The **vlan** parameter specifies:

- For IPv4 packets and switched Layer 2 IPv6 packets, the source VLAN that the packet was associated with at the time of classification
- For accelerated IPv6 packets at the Layer 2 processor of the accelerator card, the destination VLAN (see [Figure 27-5 on page 27-10](#) of [Chapter 27, Quality of Service \(QoS\)](#))

The **innertpid**, **innervlanpriority**, and **innervlanid** parameters are for use on double tagged traffic. *Inner* refers to the second 802.1Q tag in the packet. This is the one being tunnelled. The following table shows where the inner and outer tags are matched in the packet. Matching outer parameters on customer ports is not possible.

	Outer VLAN parameters (normal)	Inner VLAN parameters
Customer port	Only VLAN ID	1st tag
Core port	1st tag	2nd tag
Nested VLANs disabled	1st tag	2nd tag

If you attach the classifier to a hardware filter and nested VLANs are not being used, it is assumed that all incoming packets are double tagged. When nested VLANs are being used, outer parameters also cannot be used to match on customer ports. If the classifier is being attached to a number of ports, they are all be treated like core ports if even one port is a core port.

The **innertpid** parameter specifies the TPID in the second 802.1Q tag in the packet. The default is **any**.

The **innervlanpriority** parameter specifies the second 802.1P field in the packet. The default is **any**.

The **innervlanid** parameter specifies the tunnelled VLAN ID in the second 802.1Q tag in the packet. The default is **any**.

The **ethformat** parameter specifies the Ethernet encapsulation type of the packet. See the [create classifier](#) command for details about parameter values and encapsulation types.

If an **ethformat** option is specified (excluding **any**), the **protocol** parameter must also be used. Specifying **ethformat=snap protocol=ip** means using **protocol=0800**. Specifying **ethformat=snap protocol=ipx** means using **protocol=8137**. Specifying **ethformat=802.2 protocol=ipx** means using **protocol=E0**. [Table 26-1 on page 26-12](#) shows possible combinations for the **ethformat** and **protocol** parameters, and their implementation in the Classifier.

If you specify **protocol=ipx** you must also specify an **ethformat** other than **any**. Additionally, the IPX header parameters **ipxdaddr**, **ipxdsocket**, and **ipxssocket** are only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**.

Note that if an **ethformat** option is specified, it must include either **tagged** or **untagged**. For example, either **ethii-tagged** or **ethii-untagged**, **netwareraw-tagged** or **netwareraw-untagged**, etc.

For IPv6 packets on x900-24X switches and switched at Layer 2, valid **ethformat** options are **ethii-tagged** and **ethii-untagged**. For accelerated IPv6 packets at the accelerator card's Layer 2 processor, the valid option is **ethii-tagged**.

The **protocol** parameter specifies the protocol of the packet, as shown in [Table 26-2 on page 26-13](#). The Classifier provides a predefined list of common protocols, as shown in [Table 26-3 on page 26-13](#). Three user-specified protocols may be entered for any encapsulation type. If the command specifies a TCP or UDP packet matching rule, then this parameter will default to IP. If the command specifies **protocol=ip**, the IP packets will have either ETHII or SNAP encapsulation (e.g. **ethformat=ethii** or **ethformat=snap**). If the command specifies **protocol=ipx**, the IPX packets may have any of the four types of encapsulation (e.g. **ethformat={802.2 | ethii | netwareraw | snap}**). If a ten digit hexadecimal number is specified, e.g. **protocol=xxxxxxabcd** the last four digits (*abcd*) are used for packet classification. The default is **any**.

If you specify **protocol=ipx** you must also specify an **ethformat** other than **any**. Additionally, the IPX header parameters **ipxdaddr**, **ipxsocket**, and **ipxssocket** are only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**.

The **ipdscp** parameter specifies the Code Point bits of the DiffServ field of an IP packet (from 0 to 63, or **any**). This parameter cannot be specified in conjunction with the **iptos** parameter. An IPv6 classifier will only accept a single value for this parameter. The default is **any**.

The **iptos** parameter specifies the value of the precedence field within the TOS byte of an IP packet. This parameter cannot be specified in conjunction with the **ipdscp** parameter, and cannot be used in classifiers with a **protocol** of IPV6. The default is **any**.

The **ipprotocol** parameter specifies a Layer 4 IP protocol of an IP packet. For IPv6 packets, the **ipprotocol** parameter matches against the Next Header field of the IPv6 packet header. If the command specifies a TCP/IP packet matching rule, (e.g. **tcpdport** is specified), then the default value for this parameter is **tcp**. If the command specifies a UDP/IP packet matching rule, (e.g. the **udpport** parameter is specified), then the default value for this parameter is **udp**.

The **ipsaddr** parameter specifies the source IP address (either host or subnet) of an IP packet. If you enter an IPv6 address, **protocol** is set to IPV6. The **dhcpsnooping** option applies the classifier to entries in the DHCP snooping binding database. The default is **any**.

The **ipdaddr** parameter specifies the destination IP address (either host or subnet) of an IP packet. If you enter an IPv6 address, **protocol** is set to IPV6. The default is **any**.

The **ipxdaddr** parameter specifies the destination network address of an IPX packet. This parameter is only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**. The default is **any**.

The **ipxsocket** specifies the destination IPX socket number of an IPX packet. This parameter is only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**. The default is **any**.

The **ipxssocket** specifies the source IPX socket number of an IPX packet. This parameter is only supported for **ethformat=ethii-tagged**, **ethii-untagged**, **snap-tagged**, and **snap-untagged**. The default is **any**.

The **tcpdport** parameter specifies the TCP destination port of a TCP/IP packet, or a range of destination ports. The classifier matches all packets that have the specified destination TCP port or that have a destination TCP port in that range. The **l4mask** parameter is invalid if you specify a range. The default is **any**.

The **tcpsport** parameter specifies the TCP source port of a TCP/IP packet, or a range of source ports. The classifier matches all packets that have the specified source TCP port or that have a source TCP port in that range. The **l4smask** parameter is invalid if you specify a range. The default is **any**.

The **udpdport** parameter specifies the UDP destination port of an UDP/IP packet, or a range of destination ports. The classifier matches all packets that have the specified destination UDP port or that have a destination UDP port in that range. The **l4dmask** parameter is invalid if you specify a range. The default is **any**.

The **udpsport** parameter specifies the UDP source port of an UDP/IP packet, or a range of source ports. The classifier matches all packets that have the specified source UDP port or that have a source UDP port in that range. The **l4smask** parameter is invalid if you specify a range. The default is **any**.

The **l4dmask** parameter specifies a 2-byte hexadecimal mask to match a range of TCP/UDP destination ports in the packet, when you also specify a single TCP or UDP destination port number. The default is **ffff**.

The **l4smask** parameter specifies a 2-byte hexadecimal mask to match a range of TCP/UDP source ports in the packet, when you also specify a single TCP or UDP source port number. The default is **ffff**.

The **l5byte01** to **l5byte16** parameters each specify the properties of a single byte field to match in the Layer 5 part of IP packets, which is the TCP or UDP payload. For each byte field you want to match, specify:

- *byteoffset*, which is a decimal number in the range 0 to 37. This specifies the location of the byte to match. It refers to the offset from the start of Layer 5, after the UDP or TCP header.
- *bytevalue*, which is a 2-digit hexadecimal number. This specifies the value of the byte at the position in the frame that is determined by *byteoffset*. The classifier matches packets that have this value at this location
- (optionally) *bytemask*, which is a 2-digit hexadecimal number. This specifies an eight-bit binary mask to apply to the field. When a bit is set to **1** in the mask, the value of the bit at the same position in the byte value is used to determine a match. A **0** in the mask means that the corresponding bit is ignored. The default is **ff**, which means the classifier matches against all bits in the byte.

**Important** The classifier matches Layer 5 bytes that are in the first 80 bytes of the IP packet. The classifier does not match against bytes that are later in the packet, even if they match the classifier's settings.

When you consider packet length, remember that the location of Layer 5 in a frame varies depending on the length of the lower-layer headers. This is because header values such as the VLAN tag can be missing, and header values such as the Ethernet format specification vary in length.

You must use **l5byte01** as the first byte field and you must number additional byte fields sequentially. Each field must have a greater offset than the fields that precede it.

The **l5byte** parameters match frames with a valid TCP or UDP header, when the network protocol is IP version 4. Therefore, **protocol** defaults to **ip** and does not need to be specified. You also do not have to specify **ipprotocol**, but by default the classifier applies to both TCP and UDP packets.

The **tcpflags** parameter specifies the TCP flags of an IPv4 or IPv6 packet, one or more of **urg**, **ack**, **rst**, **syn** and **fin**. If **any** is specified, TCP flags are ignored. The default is **any**.

The **icmp** parameter specifies the ICMP type of an IPv4 packet. This can be one of the list of available options, or a decimal value in the range 0 to 255. The **icmp** parameter is valid only if the **ipprotocol** parameter has either not been specified, or **ipprotocol=icmp** has been specified. If **any** is specified, the ICMP type is ignored. The default is **any**.

The **icmpcode** parameter specifies the ICMP code of an IPv4 packet. This can be one of the list of available options, or a decimal value in the range 0 to 255. The **icmpcode** parameter is valid only if the **ipprotocol** parameter has either not been specified, or **ipprotocol=icmp** has been specified. If **any** is specified, the ICMP code is ignored. The default is **any**.

The **igmp** parameter specifies the IGMP type of an IPv4 packet. This can be one of the list of available options, or a hexadecimal value in the range of 00 to ff. The **igmp** parameter is valid only if the **ipprotocol** parameter has either not been specified, or **ipprotocol=igmp** has been specified. If **any** is specified, the IGMP type is ignored. The default is **any**.

The **eipbyte01** to **eipbyte16** parameters each specify the properties of a single byte field to match in the Layer 3 header and data of a non-IPv4 and non-IPv6 packet. The **eipbyte01** parameter must be used as the first byte field, and additional byte fields must increment sequentially, for example **eipbyte01**, **eipbyte02**, **eipbyte03**. Each field must have a greater offset than the field that precedes it.

For each byte field you want to match, specify a *byteoffset* and a *bytevalue*, and optionally, a *bytemask*.

- *byteoffset* is a decimal number in the range 0 to 65. This specifies the location of the byte to match. It refers to the offset from the start of Layer 3, after the Layer 2 encapsulation format of an Ethernet frame.
- *bytevalue* is a 2-digit hexadecimal number. This specifies the value of the byte at the frame position determined by the *byteoffset*. The classifier matches packets that have this value at this location.
- (optional) *bytemask* is a 2-digit hexadecimal number. This specifies an eight-bit binary mask to apply to the field. When a bit is set to 1 in the mask, the value of the bit at the same position in the byte is used to determine a match. If the *bytemask* is 0, the corresponding bit is ignored. The default is ff, which means the classifier matches against all bits in the byte.

**Important** The classifier matches Layer 3 bytes that are within the first 80 bytes of the non-IPv4 and non-IPv6 packet. The classifier does not match against bytes that are later in the packet, even if they match the classifier's settings. When you consider packet length, remember that the location of Layer 3 in a frame varies depending on the encapsulation length of the Ethernet frame format. This is because header values such as the VLAN tag can be missing.

**Examples** To set packet matching rule 1 so that it matches all IP packets from the IP subnet 192.168.100.2 (mask=255.255.255.0), with a destination TCP port of 23, use one of the commands:

```
set class=1 ipsa=192.168.100.2/24 tcpd=23
set class=1 mact=any prot=ip ipsa=202.36.164.2/24 tcpd=23
```

**Related Commands** [create classifier](#)  
[show classifier](#)

## show classifier

**Syntax** `SHOW CLASSifier [= {id-list | ALL}]`

```

[MACSaddr={macadd|ANY}] [MACDaddr={macadd|ANY}]
[MACDMask=macadd] [MACSMask=macadd]
[MACType={L2Ucast|L2Mcast|L2Bcast|ANY}]
[TPID={tpid|ANY}] [VLANPriority={0..7|ANY}]
[VLAN={vlanname|1..4094|ANY}] [INNERTpid={tpid|ANY}]
[INNERVLANPriority={0..7|ANY}]
[INNERVLANId={vlanname|1..4094|ANY}]
[ETHFormat={802.2-Tagged|802.2-Untagged|ETHII-Tagged|
ETHII-Untagged|NETWARERAW-Tagged|Netwareraw-untagged|
SNAP-Tagged|SNAP-Untagged|ANY}]
[PROTOCOL={protocoltype|IP|IPV6|IPX|ANY}]
[IPDScp={dscplist|ANY}] [IPTOs={0..7|ANY}]
[IPSAAddr={ipaddmask|ipv6-add/prefix-length|ANY}]
[IPDAddr={ipaddmask|ipv6-add/prefix-length|ANY}]
[IPProtocol={TCP|UDP|ICMp|IGMp|OSPF|ipprotocolnum|ANY}]
[IPXDAddr={ipxadd|ANY}]
[IPXDSocket={NCP|SAP|RIP|NNB|DIAG|NLSp|IPXwan|
ipxsocketnum|ANY}]
[IPXSsocket={NCP|SAP|RIP|NNB|DIAG|NLSp|IPXwan|
ipxsocketnum|ANY}]
[TCPSPort={portid|port-range|ANY}]
[TCPPort={portid|port-range|ANY}]
[UDPSport={portid|port-range|ANY}]
[UDPDPort={portid|port-range|ANY}] [L4SMask=mask]
[L4DMask=mask] [L5BYTE01=byteoffset,bytevalue[, bytemask]]
[L5BYTE02=byteoffset,bytevalue[, bytemask]]
[L5BYTE03=byteoffset,bytevalue[, bytemask]]
[L5BYTE04=byteoffset,bytevalue[, bytemask]]
[L5BYTE05=byteoffset,bytevalue[, bytemask]]
[L5BYTE06=byteoffset,bytevalue[, bytemask]]
[L5BYTE07=byteoffset,bytevalue[, bytemask]]
[L5BYTE08=byteoffset,bytevalue[, bytemask]]
[L5BYTE09=byteoffset,bytevalue[, bytemask]]
[L5BYTE10=byteoffset,bytevalue[, bytemask]]
[L5BYTE11=byteoffset,bytevalue[, bytemask]]
[L5BYTE12=byteoffset,bytevalue[, bytemask]]
[L5BYTE13=byteoffset,bytevalue[, bytemask]]
[L5BYTE14=byteoffset,bytevalue[, bytemask]]
[L5BYTE15=byteoffset,bytevalue[, bytemask]]
[L5BYTE16=byteoffset,bytevalue[, bytemask]]
[TCPFlags={{Urg|Ack|Rst|Syn|Fin}[,...]|ANY}]
[ICMptype={Any|ECHOReply|Unreachable|Quench|Redirect|
ECHO|Advertisement|Solicitation|Timeexceed|Parameter|
TSTAMP|TSTAMPReply|INFOREQ|INFOREP|ADDRREQ|ADDRREP|
NAMEREQ|NAMERPLY|icmp-type}]
[ICMPCode={Any|Filter|FRAGMENT|FRAGReasm|HOSTComm|
HOSTIsolated|HOSTPrec|HOSTRedirect|HOSTRTos|HOSTTos|
HOSTUNKNOWN|HOSTUNReach|NETComm|NETRedirect|NETRTos|
NETTos|NETUNKNOWN|NETUNReach|NOptr|Portunreach|
PREcedent|PROtunreach|PTRproblem|Sourceroute|Ttl|
icmp-code}]
[IGMptype={ANY|Query|V1Report|DVmrp|PIMv1|CTRace|
V2Report|V2Leave|MCTRACEResponse|MCTRACE|V3Report|
MRAdvert|MRSolicit|MRTermination|igmp-type}]
[EIPBYTE01=byteoffset,bytevalue[, bytemask]]

```



```
[EIPBYTE02=byteoffset,bytevalue[,bytemask]]
[EIPBYTE03=byteoffset,bytevalue[,bytemask]]
[EIPBYTE04=byteoffset,bytevalue[,bytemask]]
[EIPBYTE05=byteoffset,bytevalue[,bytemask]]
[EIPBYTE06=byteoffset,bytevalue[,bytemask]]
[EIPBYTE07=byteoffset,bytevalue[,bytemask]]
[EIPBYTE08=byteoffset,bytevalue[,bytemask]]
[EIPBYTE09=byteoffset,bytevalue[,bytemask]]
[EIPBYTE10=byteoffset,bytevalue[,bytemask]]
[EIPBYTE11=byteoffset,bytevalue[,bytemask]]
[EIPBYTE12=byteoffset,bytevalue[,bytemask]]
[EIPBYTE13=byteoffset,bytevalue[,bytemask]]
[EIPBYTE14=byteoffset,bytevalue[,bytemask]]
[EIPBYTE15=byteoffset,bytevalue[,bytemask]]
[EIPBYTE16=byteoffset,bytevalue[,bytemask]]
```

where:

- *id-list* is a classifier ID number or a range of ID numbers separated by a hyphen, or a comma-separated list of classifier ID numbers and/or ranges, for example 1, 2, 4-9. Classifier ID numbers range from 1 to 9999.
- *macadd* is an Ethernet six-octet MAC address, expressed as six pairs of hexadecimal digits delimited by hyphens.
- *tpid* is a 2-byte hexadecimal number.
- *vlan-name* is a unique name from 1 to 32 characters. Valid characters are uppercase and lowercase letters, digits (0-9), the underscore character, and the hyphen. The *vlan-name* cannot be a number, **all**, or **any**.
- *protocoltype* is either a valid protocol number or a recognised protocol name. A protocol number can be either 1 byte for SAP, 2 bytes for ETHII or 5 bytes for an 802.3/802.2 SNAP type packet, and is specified in hexadecimal.
- *dscplist* is a single number or a group of numbers, either a comma separated list, a range (specified as n-m) or a combination of the two. Numbers start at 0 and end at 63.
- *ipprotocolnum* is either an IP protocol number or a recognised IP protocol name. An IP protocol number is expressed as a 1 byte decimal number.
- *ipaddmask* is an IP address in dotted decimal notation with an optional mask. The optional mask is specified by adding “/M” following the IP address, where M is the number of contiguous bits in the mask; M ranges from 0 to 32 inclusive.
- *ipv6-add* is a valid IPv6 address, with its prefix length indicated by slash notation.
- *prefix-length* is an integer from 1 to 128.
- *ipxadd* is an IPX network address expressed as a 4 byte hexadecimal number.
- *ipxsocketnum* is either an IPX socket number or a recognised IPX socket type. An IPX socket number is expressed as a 2 byte hexadecimal number.
- *portid* is a TCP/IP or UDP/IP port number.
- *port-range* is a hyphen-separated range of TCP/IP or UDP/IP ports, such as 5550-5554.
- *mask* is a 2-byte hexadecimal number in the range 0001 to ffff.
- *byteoffset* is a decimal number:
  - For L5BYTE<sub>xx</sub>, the range is 0 to 37
  - For EIPBYTE<sub>xx</sub>, the range is 0 to 65

- *bytevalue* is a 2-digit hexadecimal number.
- *bytemask* is a 2-digit hexadecimal number.
- *icmp-type* is a decimal number in the range 0 to 255.
- *icmp-code* is a decimal number in the range 0 to 255.
- *igmp-type* is a 2 digit hexadecimal number.

**Description** This command displays information about the specified classifier or classifiers, and packet matching rules. The information that is displayed is dependant on what you specify for the **classifier** parameter:

- If the **classifier** parameter specifies no value, then all rules are displayed.
- If the **classifier** parameter specifies a value, then a detailed output is displayed. In the detailed output, only parameters that have a non-default value are shown.

You can use any of the optional parameters as a filter to limit the number of classifiers that are displayed. For the meaning of each parameter, see the [create classifier command on page 26-4](#).

For example output, see:

- [Figure 26-2](#) for summary output, which is displayed if you specify **show classifier** with no other options
- [Figure 26-3 on page 26-27](#), and [Figure 26-4 on page 26-27](#) for classifiers that match on TCP/IP and UDP/IP data flows
- [Figure 26-5 on page 26-27](#) for classifiers that match on ICMP data flow
- [Figure 26-6 on page 26-28](#) for classifiers that match on IGMP data flow
- [Figure 26-7 on page 26-28](#) for classifiers that match on Layer 3 byte data
- [Figure 26-8 on page 26-28](#) for classifiers that match on MAC address
- [Figure 26-9 on page 26-29](#) for classifiers that match on IPX data flows
- [Figure 26-10 on page 26-29](#) for classifiers that match on double tagging
- [Figure 26-11 on page 26-29](#) for classifiers that match on Layer 5 byte data
- [Figure 26-12 on page 26-30](#) for all classifiers

For all parameter descriptions, see:

- [Table 26-4 on page 26-30](#)

Figure 26-2: Example output from the **show classifier** command

Classifier General Info		
-----		
Total number of rules .... 7		
Rule	Type	Related Module(s)
-----		
1	L2	L3 switch
2	L2	L3 switch, QOS
100	L4,L3,L2	QOS
200	L3,L2	None
500	L5,L3	None
600	L3	None
9999	Match all	None
-----		

Figure 26-3: Example output from the **show classifier** command (TCP/IP data flow)

```

Classifier Rules
-----
Rule ..... 1
  M-Type ..... L2UCAST
  VLAN ..... vlan1234 (1234)
  E-Format ..... ETHII-UNTAGGED
  Protocol ..... 0800 (IP EthII)
  S-IP Address ..... 192.168.123.123/32
  D-IP Address ..... 192.168.123.123/32
  IP Protocol ..... TCP
  S-TCP Port ..... 23
  D-TCP Port ..... 23
  TCP Flags ..... SYN,FIN
-----

```

Figure 26-4: Example output from the **show classifier** command (UDP/IP data flow)

```

Classifier Rules
-----
Rule ..... 21
  M-Type ..... L2UCAST
  VLAN ..... vlan1234 (1234)
  E-Format ..... ETHII-UNTAGGED
  Protocol ..... 0800 (IP EthII)
  S-IP Address ..... 192.168.123.123/32
  D-IP Address ..... 192.168.123.123/32
  IP Protocol ..... UDP
  S-UDP Port ..... 23
  D-UDP Port ..... 23
-----

```

Figure 26-5: Example output from the **show classifier** command (ICMP data flow)

```

Classifier Rules
-----
Rule ..... 21
  M-Type ..... L2UCAST
  VLAN ..... vlan1234 (1234)
  E-Format ..... ETHII-UNTAGGED
  Protocol ..... 0800 (IP EthII)
  S-IP Address ..... 192.168.123.123/32
  D-IP Address ..... 192.168.123.123/32
  IP Protocol ..... ICMP
  ICMP code ..... 7 (HOSTUNKNOWN)
  ICMP type ..... 3 (UNREACHABLE)
-----

```

Figure 26-6: Example output from the **show classifier** command (IGMP data flow)

```

Classifier Rules
-----
Rule ..... 21
  M-Type ..... L2UCAST
  VLAN ..... vlan1234 (1234)
  E-Format ..... ETHII-UNTAGGED
  Protocol ..... 0800 (IP EthII)
  S-IP Address ..... 192.168.123.123/32
  D-IP Address ..... 192.168.123.123/32
  IP Protocol ..... IGMP
  IGMP type ..... 0x17 (V2LEAVE)
-----

```

Figure 26-7: Example output from the **show classifier** command (Layer 3 byte data)

```

Classifier Rules
-----
Rule ..... 2222
  D-MAC Address ..... aa-bb-cc-dd-ee-ff
  S-MAC Address ..... aa-bb-cc-dd-ee-ff
  M-Type ..... L2UCAST
  VLAN ..... vlan1234 (1234)
  E-Format ..... SNAP
  Protocol ..... 1234567890 (-)
  Layer 3 Byte 01:
    Offset ..... 0
    Value ..... 50
  Layer 3 Byte 02:
    Offset ..... 1
    Value ..... 4f
  Layer 3 Byte 03:
    Offset ..... 2
    Value ..... 53
  Layer 3 Byte 04:
    Offset ..... 3
    Value ..... 54
    Mask ..... fc
-----

```

Figure 26-8: Example output from the **show classifier** command (MAC address)

```

Classifier Rules
-----
Rule ..... 2222
  D-MAC Address ..... aa-bb-cc-dd-ee-ff
  S-MAC Address ..... aa-bb-cc-dd-ee-ff
  M-Type ..... L2UCAST
  VLAN ..... vlan1234 (1234)
  E-Format ..... SNAP-TAGGED
  Protocol ..... 1234567890 (-)
-----

```

Figure 26-9: Example output from the **show classifier** command (IPX data flow)

```

Classifier Rules
-----
Rule ..... 31
  E-Format ..... ETHII-TAGGED
  Protocol ..... 8137 (IPX EthII)
  D-IPX Socket ..... RIP
-----

```

Figure 26-10: Example output from the **show classifier** command (double tag matching)

```

Classifier Rules
-----
Rule ..... 1
  TPID ..... 8100
  VLAN Priority ..... 3
  Inner TPID ..... 8100
  Inner VLAN ID ..... 10
-----

```

Figure 26-11: Example output from the **show classifier** command (layer 5 byte data)

```

Classifier Rules
-----
Rule ..... 1
  Layer 5 Byte 01:
    Offset ..... 0
    Value ..... 50
  Layer 5 Byte 02:
    Offset ..... 1
    Value ..... 4f
  Layer 5 Byte 03:
    Offset ..... 2
    Value ..... 53
  Layer 5 Byte 04:
    Offset ..... 3
    Value ..... 54
    Mask ..... fc
-----

```

Figure 26-12: Example output from the **show classifier=all** command

```

Classifier Rules
-----
Rule ..... 1
  VLAN ..... default (1)

Rule ..... 2
  VLAN ..... v2 (2)

Rule ..... 100
  Protocol ..... IP
  D-IP Address ..... 10.0.0.1/32
  IP Protocol ..... TCP
  D-TCP Port ..... 23

Rule ..... 200
  Protocol ..... IP
  D-IP Address ..... 10.0.0.1/32

Rule ..... 500
  Layer 5 Byte 01:
    Offset ..... 1
    Value ..... 42
    Mask ..... c3

Rule ..... 600
  Layer 3 Byte 01:
    Offset ..... 2
    Value ..... 18
    Mask ..... 59

Rule ..... 9999
  Match all frames
-----

```

Table 26-4: Parameters in output of the **show classifier** command

Parameter	Meaning
Rule	The rule identifier for the packet matching rule/classifier.
Type	A list of the OSI layers at which specified parameters (those with non-default values) in the rule operate, one or more of: <b>L5</b> , <b>L4</b> , <b>L3</b> , <b>L2</b> , <b>L1</b> .
Related module(s)	The name of the module(s) that are currently using the rule.
D-MAC Address	The destination MAC address field of a packet.
D-MAC Addr mask	A MAC address that specifies a 48-bit binary mask to apply to the destination MAC address before determining a match. A 1 in the mask means that the value of the bit in that position is used to determine a match, and a 0 means that the bit is ignored. The default mask value is ff-ff-ff-ff-ff-ff.
S-MAC Address	The source MAC address field of a packet. If DHCP Snooping is displayed, the classifier is being applied to entries in the DHCP snooping binding database.
S-MAC Addr mask	A MAC address that specifies a 48-bit binary mask to apply to the source MAC address before determining a match. A 1 in the mask means that the value of the bit in that position is used to determine a match, and a 0 means that the bit is ignored. The default mask value is ff-ff-ff-ff-ff-ff.

Table 26-4: Parameters in output of the **show classifier** command (cont.)

Parameter	Meaning
M-Type	The type of MAC address specified in the destination MAC address field, one of L2UCAST, L2BMCAS, or ANY.
VLAN	The name of a VLAN. The VLAN Identifier appears in brackets. This VLAN is the source VLAN that a packet is associated with on ingress to the Layer 3 switch.
E-Format	The Ethernet encapsulation format for the packet, one of 802.2-TAGGED, 802.2-UNTAGGED, ETHII-TAGGED, ETHII-UNTAGGED, NETWARERAW-TAGGED, NETWARERAW-UNTAGGED, SNAP-TAGGED, SNAP-UNTAGGED, or ANY.
Protocol	The hexadecimal value of the protocol. If the protocol is not for the general family of IP and IPX protocols, and the commonly known name for the protocol is known to the Classifier, then this common name is printed in brackets after the hexadecimal number.
S-IP Address	The source IP address field of a packet. If DHCP Snooping is displayed, the classifier is being applied to entries in the DHCP snooping binding database.
D-IP Address	The destination IP address field of a packet.
IP Protocol	The Layer 4 IP protocol field of a packet.
ICMP Code	The ICMP message reason code to match against the ICMP code field in an ICMP packet header. A decimal value is shown, with an equivalent parameter option in brackets if available.
ICMP Type	The ICMP message type to match against the ICMP type field in an ICMP packet header. A decimal value is shown, with an equivalent parameter option in brackets if available.
IGMP Type	The IGMP message type to match against the IGMP type field in an IGMP packet header. A hexadecimal value is shown, with an equivalent parameter option in brackets if available.
TOS/DSCP	The IP TOS or DiffServ Code Point field of a packet.
S-TCP Port	The source TCP/IP port field of a packet.
S-TCP Port range	A range of values in the source TCP/IP port field amongst a group of packets.
D-TCP Port	The destination TCP/IP port field of a packet.
D-TCP Port range	A range of values in the destination TCP/IP port field amongst a group of packets.
TCP Flags	TCP data flow only. A series of letters representing the TCP/IP flag field, one of URG, ACK, RST, SYN, or FIN.
S-UDP Port	The source UDP/IP port field of a packet.
S-UDP Port range	A range of values in the source UDP/IP port field amongst a group of packets.
D-UDP Port	The destination UDP/IP port field of a packet.
D-UDP Port range	A range of values in the destination UDP/IP port field amongst a group of packets.
S-TCP Mask	A 16-bit binary mask to apply against the value of the source TCP/IP port field of a packet. This does not apply if a range of values is specified for the source TCP/IP port field.
D-TCP Mask	A 16-bit binary mask to apply against the value of the destination TCP/IP port field of a packet. This does not apply if a range of values is specified for the destination TCP/IP port field.

Table 26-4: Parameters in output of the **show classifier** command (cont.)

Parameter	Meaning
S-UDP Mask	A 16-bit binary mask to apply against the value of the source UDP/IP port field of a packet. This does not apply if a range of values is specified for the source UDP/IP port field
D-UDP Mask	A 16-bit binary mask to apply against the value of the destination UDP/IP port field of a packet. This does not apply if a range of values is specified for the destination UDP/IP port field.
D-IPX Address	The destination IPX network address field of a packet.
D-IPX Socket	The destination IPX socket field of a packet.
S-IPX Socket	The source IPX socket field of a packet.
TPID	The Tag Protocol Identifier field in the packet.
VLAN Priority	The 802.1P priority of the first VLAN tag.
Inner TPID	The Tag Protocol Identifier in the second 802.1Q tag in the packet.
Inner VLAN Priority	The 802.1P priority of the second VLAN tag.
Inner VLAN ID	The VLAN ID of the second VLAN tag.
Layer 5 Byte 01 to Layer 5 Byte 16	Each Layer 5 Byte field specifies the properties of a single byte field to match in the Layer 5 part of IP packets, which is the TCP or UDP payload.
	Offset The offset of a byte from the start of Layer 5. This specifies the location of the byte to match.
	Value The hexadecimal value to match at the location specified by Offset
	Mask A hexadecimal number that specifies an eight-bit binary mask to apply to the value before determining a match. A 1 in the mask means that the value of the bit in that position is used to determine a match, and a 0 means that the bit is ignored.
Layer 3 Byte 01 to Layer 3 Byte 16	Each Layer 3 Byte field specifies the properties of a single byte field to match in the Layer 3 part of non-IPv4 and IPv6 packets.
	Offset The offset of a byte from the start of Layer 3. This specifies the location of the byte to match.
	Value The hexadecimal value to match at the location specified by Offset.
	Mask A hexadecimal number that specifies an eight-bit binary mask to apply to the value before determining a match. A 1 in the mask means that the value of the bit in that position is used to determine a match, and a 0 means that the bit is ignored.

**Examples** To display all the packet matching rules, use one of the commands:

```
sh class
sh class=all
```

**Related Commands** [create classifier](#)  
[destroy classifier](#)  
[set classifier](#)