

Classifier-Based Software Configuration

Contents

Using Classifier-Based Service Policies	9-2
Introduction	9-2
Classifier-Based Configuration Model	9-2
Creating a Traffic Class	9-4
Using Match Criteria	9-4
Class Configuration Procedure	9-5
Optional ICMP Match Criteria	9-13
Optional IGMP Match Criteria	9-16
Optional TCP and UDP Match Criteria	9-17
Using CIDR Notation for IPv4/IPv6 Addresses	9-19
Resequencing Match/Ignore Statements	9-23
Creating a Service Policy	9-24
Modifying Classes in a Policy	9-28
Resequencing Classes in a Policy	9-29
Applying a Service Policy to an Interface	9-30
Where to Go From Here	9-33

Using Classifier-Based Service Policies

Introduction

Classifier-based service policies are designed to work with existing globally-configured, switch-wide and port-wide configurations by allowing you to zoom in on a subset of port or VLAN traffic to further manage it. These policies take precedence over, and may override, globally-configured settings.

Classifier-based service policies provide greater control for managing network traffic. Using multiple match criteria, you can finely select and define the classes of traffic that you want to manage. Policy actions determine how you can handle the selected traffic.

Starting in software release K.14.01, the Classifier feature introduces:

- A finer granularity than globally-configured features for classifying network traffic (IPv4 or IPv6) into classes that can be used in cross-feature software configurations
- Additional policy actions to manage selected traffic, such as rate-limiting and IP precedence marking
- The configuration of service policies for classified traffic with the following software features:
 - Quality of Service (QoS)
 - Traffic mirroring
- The application of service policies to specific inbound traffic flows on individual port and VLAN interfaces (rather than only on switch-wide or port-wide traffic).

Classifier-Based Configuration Model

Classifier-Based Configuration Task	Page Reference
Creating a Traffic Class	page 9-4
Creating a Service Policy	page 9-24
Applying a Service Policy to an Interface	page 9-30

Classifier-based software configuration consists of the following general steps:

1. Determine the inbound traffic you want to manage and how you want to manage it; for example, rate-limit, prioritize, mirror, and so on.
2. Classify the traffic that you want to manage by configuring a class, using **match** and **ignore** commands. A traffic class is configured separately from service policies and can be used in various policies.
3. Configure a service policy for one or more traffic classes, including an optional, default class. A policy consists of configuration commands executed on specified traffic classes for one of the following software features:
 - Quality of Service (**policy qos** command)
 - Port mirroring (**policy mirror** command)
4. Assign the policy to an inbound port or VLAN interface using the **interface service-policy in** or **vlan service-policy in** command.

Figure 9-1 shows an overview of classifier-based software configuration.

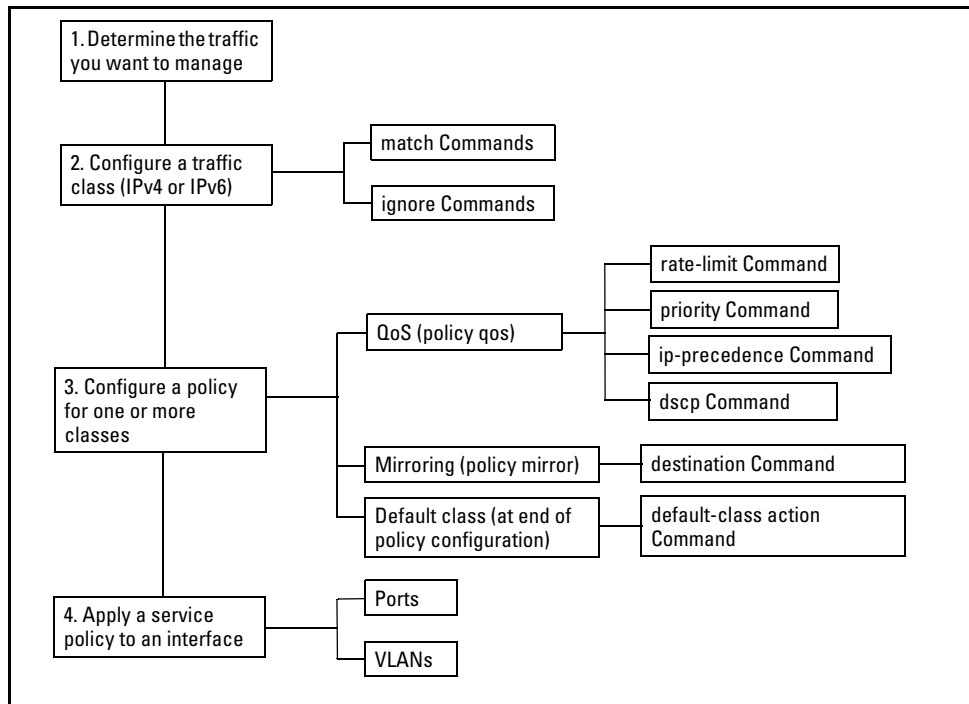


Figure 9-1. Classifier-Based Configuration Model

Creating a Traffic Class

In the classifier-based configuration model, you use match criteria to create a class of IPv4 or IPv6 traffic and select the packets you want to manage. In a class configuration, match criteria consist of **match** and **ignore** commands. These commands determine the packets that belong to a class. (Match/ignore criteria are modelled on the permit/deny criteria used in ACLs.)

The traffic classes you configure can be used later in the service policies you create for different software features, such as QoS and port mirroring. The match criteria used in match/ignore statements are the same across software features.

Using Match Criteria

To identify the packets that belong to a traffic class for further processing by policy actions, use **match** and **ignore** commands in a class configuration:

- **match** commands define the values that header fields must contain for a packet to belong to the class and be managed by policy actions.
- **ignore** commands define the values which, if contained in header fields, exclude a packet from the policy actions configured for the class. An ignored packet is transmitted without having a policy action performed on it.

Match/ignore statements compare the values in packet fields with specified criteria in the sequential order in which the statements are entered in the class, until a match is found. Be sure to enter match/ignore statements in the *precise order* in which you want their criteria to be used to check packets.

- As soon as a field in a packet header matches the criteria in a **match** statement, the sequential comparison of match criteria in the class stops, and the policy actions configured for the class are executed on the packet (see “Creating a Service Policy” on page 9-24).
- If a packet matches the criteria in an **ignore** statement, the sequential comparison of match criteria in the class stops, and no policy action is performed on the packet.

If a packet does not match the criteria in any match/ignore statement in a class configuration, one of the following actions is taken:

- The packet is transmitted without a policy action performed on it.

- If a default class is configured in the policy, the actions specified in the **default-class** command are performed on packets that do not match the criteria in preceding classes in the policy (see Step 3 in “Creating a Service Policy” on page 9-24).

The following match criteria are supported in match/ignore statements for inbound IPv4/IPv6 traffic:

- IP source address (IPv4 and IPv6)
- IP destination address (IPv4 and IPv6)
- Layer 2 802.1Q VLAN ID
- Layer 3 IP protocol
- Layer 3 IP precedence bits
- Layer 3 DSCP bits
- Layer 4 TCP/UDP application port (including TCP flags)
- VLAN ID

Class Configuration Procedure

To configure a traffic class to be used in one or more policies, follow these steps:

1. Enter the **class** command from the global configuration context.

Context: Global configuration

Syntax: [no] class < ipv4 | ipv6 > <classname >

Defines a traffic class and specifies whether a policy is to be applied to IPv4 or IPv6 packets, where < classname > is a text string (64 characters maximum).

*After you enter the **class** command, you enter the class configuration context to specify match criteria. A traffic class contains a series of **match** and **ignore** commands, which specify the criteria used to classify packets.*

2. Enter one or more **match** or **ignore** commands from the class configuration context to filter traffic and determine the packets on which policy actions will be performed.

Context: Class configuration

Syntax: [no] [seq-number] < match | ignore > < ip-protocol >
< source-address > < destination-address > [ip-dscp codepoint]
[precedence precedence-value] [tos tos-value] [vlan vlan-id]

[seq-number]

*The (optional) **seq-number** parameter sequentially orders the match/ignore statements that you enter in a traffic class configuration. Packets are checked by the statements in numerical order.*

*Default: Match/ignore statements are numbered in increments of 10, starting at 10. To re-number the match/ignore statements in a class configuration, use the **resequence** command (see “Resequencing Match/Ignore Statements” on page 9-23).*

< match | ignore >

Defines the classifier criteria used to determine which packets belong to the traffic class.

*If a packet matches a **match** criterion, it becomes a member of the traffic class and is forwarded according to the actions configured with the **policy** command. If a packet matches an **ignore** criterion, no policy action is performed on the packet. You can enter one or more match/ignore statements in a traffic class.*

*To remove a match/ignore statement from a class configuration, enter the **no** < seq-number > command or the complete form of a **no match ...** or **no ignore ...** command.*

< ip-protocol >

*Specifies an IP protocol to be matched in packet fields of IPv4 or IPv6 traffic, where **ip-protocol** is one of the values described below.*

When entering a match/ignore command in an IPv4 or IPv6 class, type ? to display a list of valid **ip-protocol** entries.

- In an IPv4 class, you can enter any of the following IPv4 protocol match criteria:

ah	esp	gre	icmp*	igmp*
ip	ip-in-ip	ipv6-in-ip	ospf	pim
sctp	snmp	tcp*	udp*	vrp

* For IPv4 ICMP, IGMP, TCP, and UDP packets, you can enter additional match criteria; see:

“Optional ICMP Match Criteria” on page 9-13

“Optional IGMP Match Criteria” on page 9-16

“Optional TCP and UDP Match Criteria” on page 9-17

To specify an IPv4 protocol as match criteria, you can also enter its protocol number. Valid values are from **0** to **255**.

For example, **8** means Exterior Gateway Protocol; **121** means Simple Message Protocol. For a list of IPv4 protocol numbers and corresponding protocol names, refer to the IANA “Protocol Number Assignment Services” at www.iana.com.

- In an IPv6 class, you can enter any of the following IPv6 protocol match criteria:

ah	esp	icmp*	ipv6
sctp	tcp*	udp*	

* For IPv6 ICMP, TCP, and UDP packets, you can enter additional match criteria; see:

“Optional ICMP Match Criteria” on page 9-13

“Optional TCP and UDP Match Criteria” on page 9-17

< source-address > < destination-address >

Define the source IP address (SA) and destination IP address (DA) that a packet must contain to match a match/ignore statement in an IPv4 or IPv6 traffic class. Note that both the source and destination address parameters are required entries in a match/ignore statement.

Valid values for < **source-address** > and < **destination-address** > are as follows:

- **any** — Matches IPv4 or IPv6 packets from, or destined to, any SA or DA.

- **host < SA | DA >** — *Matches only packets from a specified IPv4 or IPv6 host address. Use this match criterion when you want to match IP packets from only one SA/DA.*

- **SAv4 mask | DAv4 mask** — *Matches packets received from, or destined to, a subnet or a group of IPv4 addresses defined by the IPv4 mask. Enter an IPv4 mask in dotted-decimal format for an IPv4 address (for example, 10.28.31.1 0.0.0.255).*

See “Using CIDR Notation for IPv4/IPv6 Addresses” on page 9-19 for information on how IPv4 mask bit sets define a match.

Note that an IPv6 address and mask are not supported as <SAv6 mask> and <DAv6 mask> match criteria.)

- **SAv4/mask-length | DAv4/mask-length** — *Matches packets received from, or destined to, an IPv4 subnet or a group of IPv4 addresses defined by the mask length. Enter the mask length for an IPv4 SA or DA mask in CIDR format by using the number of significant bits. (for example, 10.28.31.3/24).*

An IPv4 mask-length is applied to an SA or DA in a match/ignore statement to define which bits in a packet’s SA/DA must exactly match the specified SA/DA and which bits need not match.

For example, 10.28.31.3/24 means that the leftmost 24 bits in an IPv4 source or destination address in a packet header must match the same bit set in the specified IPv4 address (in this case, 10.28.3.3).

For more information, see “Using CIDR Notation for IPv4/IPv6 Addresses” on page 9-19.

An IPv4 mask-length is applied from right to left, starting from the rightmost bits.

Example: *10.10.10.1/24 and 10.10.10.1 0.0.0.255 both match IPv4 addresses in the range 10.10.10. (1 to 255).*

Note: *Specifying a group of non-contiguous IP source addresses may require more than one match/ignore statement.*

- **SAv6/prefix-length | DAv6/prefix-length** — Matches packets received from, or destined to, an IPv6 subnet or a group of IPv6 addresses defined by the prefix length. Enter the prefix length for an IPv6 SA/DA in CIDR format by using the number of significant bits; for example:
2001:db8:2620:212::01b4/64.

An IPv6 prefix-length is applied to an SA/DA in a match/ignore statement to define which bits in a packet's SA/DA must exactly match the specified SA/DA and which bits need not match.

For example, 2001:db8:2620:212::01b4/64 means that the leftmost 64 bits in a 128-bit IPv6 source or destination address in a packet header must match the same bit set in the specified IPv6 address (in this case, 2001:db8:2620:212::01b4).

For more information, see “Using CIDR Notation for IPv4/IPv6 Addresses” on page 9-19.

An IPv6 prefix-length is applied from left to right, starting from the leftmost bits.

Example: 2001:db8::0001:2620:a03:e102:127/64 and 2001:db8::1:244:17ff:feb6:d37d/64 both match IPv6 addresses with a network prefix of 2001:db8:0000:0001.

[ip-dscp codepoint]

(Optional) Matches the six-bit DSCP codepoint in IPv4 or IPv6 packets to further define match criteria.

Valid values for **codepoint** are one of the following:

- Numeric equivalent of a binary DSCP bit set from **0** (low priority) to **63** (high priority)

-ASCII standard name for a binary DSCP bit set:

af11 (001010)	af42 (100100)
af12 (001100)	af43 (100110)
af13 (001110)	ef (101110)
af21 (010010)	cs1 (001000) = precedence 1
af22 (010100)	cs2 (010000) = precedence 2
af23 (010110)	cs3 (011000) = precedence 3
af31 (011010)	cs4 (100000) = precedence 4
af32 (011100)	cs5 (101000) = precedence 5
af33 (011110)	cs6 (110000) = precedence 6
af41 (100010)	cs7 (111000) = precedence 7
default (000000)	

To display a list of valid **codepoint** entries when you enter **ip-dscp** in a match/ignore statement, type **?**. The DSCP codepoints are the leftmost six bits of the ToS/Traffic Class byte (see Figure 9-2).

[precedence precedence-value]

(Optional) Matches the three-bit IP precedence value in IPv4 or IPv6 packets to further define match criteria. Valid values for **precedence-value** are either the numeric value (0 to 7) or corresponding name of an IP precedence bit set:

- 0 routine
- 1 priority
- 2 immediate
- 3 flash
- 4 flash-override
- 5 critical
- 6 internet (for internetwork control)
- 7 network (for network control)

To display a list of valid **precedence-value** entries when you enter **precedence** in a match/ignore statement, type **?**.

Notes: When used as a match criteria, the IP precedence value is applied in addition to all other criteria configured in the match/ignore statement. You can enter a match/ignore statement either with or without a **precedence-value**.

The IP precedence bits are the leftmost three bits of the ToS/Traffic Class byte (see Figure 9-2). The numeric value (0 to 7) of the IP precedence bits corresponds to the hexadecimal equivalent of the three binary “0” and/or “1” bits in the IP precedence field. For example if the IP precedence-bit binary values are **111**, the numeric value is **7** (1+2+4). Similarly, if the IP precedence bits are **010**, the numeric value is **2** (0+2+0).

[tos tos-value]

(Optional) Matches the Delay Throughput Reliability (DTR) bit set in the IPv4 Type-of-Service or IPv6 Traffic Class byte to further define match criteria.

Valid values are the numeric value or corresponding name of the DTR bit set. Some useful values are as follows:

- 0 normal
- 2 max-reliability
- 4 max-throughput
- 8 minimize-delay

Default: **0** or **normal**.

To display a list of valid **tos-value** entries when you enter **tos** in a match/ignore statement, type **?**.

Notes: When used as a match criteria, the ToS/Traffic Class byte entry is applied in addition to all other criteria configured in the match/ignore statement. You can enter a match/ignore statement either with or without a **tos-value**.

Figure 9-2 shows the DTR bit set in a ToS/Traffic Class byte in an IPv4/IPv6 packet header and the difference between the DSCP, DTR, and precedence bits.

[vlan *vlan-id*]

(Optional) Matches the VLAN ID number in the Layer 2 header of 802.1Q VLAN packets to further define match criteria. Valid VLAN IDs are from 1 to 4094.

Figure 9-2 uses a sample ToS/Traffic Class field of **10101000** to show the differences between the IP precedence (**101**), DSCP (**101010**), and ToS/Traffic Class (**10101000**) bits. Note that the rightmost two bits are reserved as **00**.

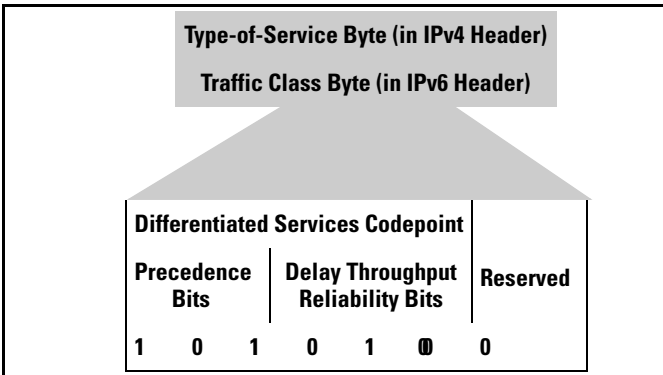


Figure 9-2. Example of a ToS/Traffic Class Field

3. Enter the **exit** command to exit the class configuration context.

To display a class configuration, enter the **show class < ipv4 | ipv6 > < classname >** command (see Figure 9-7).

To edit a class configuration, re-enter the class configuration context (**class** command) and enter new match/ignore statements as follows:

- If you do not enter a sequence number, a new statement is inserted at the end of the class configuration.
- To remove a match/ignore statement from a class configuration, enter the **no <sequence-number >** command or the complete form of the **no match ...** or **no ignore ...** command.
- To resequence the order in which match/ignore statements are listed, enter the **resequence** command (see “Resequencing Match/Ignore Statements” on page 9-23).
- To replace an existing match/ignore statement, enter the **no <sequence-number >** command to delete the entry and re-enter a complete **<sequence-number > match ...** or **<sequence-number > ignore ...** command.

When you exit class configuration context, the changes are automatically saved and applied to existing policy configurations on the switch that use the class *if the policies have not been applied to an interface*. If a policy has already been applied to an interface, the editing changes are not accepted and an error message is displayed.

Example. Figure 9-3 shows an example of two class configurations:

- “AdminTraffic” selects the administrative traffic sent to, and received from, the IPv4 address of an administrator’s PC.
- “http” selects HTTP traffic sent to TCP ports 80, 443, and 8080, and excludes HTTP traffic sent to, and received from, TCP port 1214.

```
ProCurve(config)# class ipv4 AdminTraffic
ProCurve(class-config)# match ip 15.29.16.1/10 any
ProCurve(class-config)# match ip any 15.29.16.1/10
ProCurve(class-config)# exit
ProCurve(config)# class ipv4 http
ProCurve(class-config)# match tcp any any eq 80
ProCurve(class-config)# match tcp any any eq 443
ProCurve(class-config)# match tcp any any eq 8080
ProCurve(class-config)# ignore tcp any eq 1214 any
ProCurve(class-config)# ignore tcp any any eq 1214
ProCurve(class-config)# exit
```

Figure 9-3. Example of a Class Configuration

Optional ICMP Match Criteria

To more precisely define the ICMP packets that you want to match in an IPv4 or IPv6 traffic class, use the optional parameter settings described in this section. For example, instead of matching or ignoring all ICMP traffic, you can configure a class that matches only a specific ICMP packet type by entering its numeric value.

Context: Class configuration

Syntax: [no] [seq-number] < match | ignore > **icmp**
< source-address > < destination-address >
[**icmp-type-number** | **icmpv4-type-name** | **icmpv6-type-name**]
[ip-dscp *codepoint*] [precedence *precedence-value*]
[tos *tos-value*] [vlan *vlan-id*]

*If you enter **icmp** as the IP protocol type in a match/ignore statement, you can optionally specify an ICMP packet type to more precisely define match criteria for a traffic class. Enter the optional ICMP match criteria immediately after the destination address (DA) value in the command syntax; for example:*

```
ProCurve(config-class)# match icmp any any host-unknown
```

```
ProCurve(config-class)# match icmp any any 3 7
```

[*icmp-type-number*]

Configures an ICMP packet type as match criteria in a class configuration by entering its numeric identifier.

*Valid values are from **0** to **255**.*

For information on ICMP packet-type names and numeric identifiers, go to the Internet Assigned Numbers Authority (IANA) website at www.iana.com, click on “Protocol Number Assignment Services”, and then go to the selections under “Internet Control Message Protocol (ICMP) Parameters”.

[*icmpv4-type-name*]

You can also enter any of the following ICMPv4 packet-type names to configure more precise match criteria for ICMP packets in an IPv4 class configuration.

*To display a list of valid **icmpv4-type-name** entries when you enter **icmp** as the IP protocol type in a match/ignore statement, type **?**. Some of the valid values are as follows:*

administratively-prohibited	net-tos-unreachable
alternate-address	net-unreachable
conversion-error	network-unknown
dod-host-prohibited	no-room-for-option
dod-net-prohibited	option-missing
echo	packet-too-big
echo-reply	parameter-problem
general-parameter-problem	port-unreachable
host-isolated	precedence-unreachable
host-precedence-unreachable	protocol-unreachable
host-redirect	reassembly-timeout
host-tos-redirect	redirect
host-tos-unreachable	router-advertisement
host-unknown	router-solicitation
host-unreachable	source-quench
information-reply	source-route-failed
information-request	time-exceeded
mask-reply	timestamp-reply
mask-request	timestamp-request
mobile-redirect	traceroute
net-redirect	ttl-exceeded
net-tos-redirect	unreachable

[*icmpv6-type-name*]

You can also enter any of the following ICMPv6 packet-type names to configure more precise match criteria for ICMP packets in an IPv6 class configuration.

*To display a list of valid **icmpv6-type-name** entries when you enter **icmp** as the IP protocol type in a match/ignore statement, type **?**. Some of the valid values are as follows:*

cert-path-advertise	mobile-advertise
cert-path-solicit	mobile-solicit
destination-unreachable	nd-na
echo-reply	nd-ns
echo-request	node-info
home-agent-reply	node-query
home-agent-request	packet-too-big
inv-nd-na	parameter-problem
inv-nd-ns	redirect
mcast-router-advertise	router-advertisement
mcast-router-solicit	router-renum
mcast-router-terminate	router-solicitation
mld-done	time-exceeded
mld-query	ver2-mld-report
mld-report	

Optional IGMP Match Criteria

To more precisely define the IGMP packets that you want to match in an IPv4 traffic class, use the optional parameter settings described in this section. For example, instead of matching all IGMP traffic, you can configure a class that matches only a specific IGMP packet type.

Context: Class configuration

Syntax: [no] [seq-number] < match | ignore > **igmp**
< source-address > < destination-address > [**igmp-type**]
[ip-dscp codepoint] [precedence precedence-value]
[tos tos-value] [vlan vlan-id]

*If you enter **igmp** as the IP protocol type in a match/ignore statement, you can optionally specify an IGMP packet type to more precisely define match criteria for a traffic class. Enter the optional IGMP match criteria immediately after the destination IP address (DA) value in the command syntax; for example:*

```
ProCurve(config-class)# match igmp any any host-  
query
```

[**igmp-type**]

Configures an IGMP packet type as match criteria in a class configuration. Some of the valid values for IGMP packet-type names are as follows:

dvmrp	mtrace-request	trace
host-query	mtrace-reply	v2-host-leave
host-report	pim	v2-host-report
		v3-host-report

*To display a list of valid **igmp-type** entries when you enter **igmp** as the IP protocol type in a match/ignore statement, type ?.*

Optional TCP and UDP Match Criteria

In a class configuration, you can enter match/ignore statements that more precisely define the TCP or UDP traffic that you want to match in an IPv4 or IPv6 traffic class. For example, you can enter a port number as a match criterion that specifies one or more TCP source ports, destination ports, or both.

Context: Class configuration

Syntax: [no] [seq-number] < match | ignore > < tcp | udp >
< source-address > [operator < tcp-src-port | udp-src-port >]
< destination-address > [operator < tcp-dest-port [established]
[tcp-flag [tcp-flag ...]] | udp-dest-port >] [ip-dscp codepoint]
[precedence precedence-value] [tos tos-value] [vlan vlan-id]

If you use TCP or UDP as the IP protocol type in a match/ignore statement, you can optionally configure TCP or UDP source and/or destination port numbers or ranges of numbers to more precisely define match criteria for a traffic class. Enter the optional TCP/UDP match criteria immediately after the source and/or destination address in the command syntax; for example:

```
ProCurve(config-class)# match tcp host 10.20.10.17
eq 23 host 10.20.10.155 established
ProCurve(config-class)# match tcp host 10.10.10.100
host 10.20.10.17 eq telnet
ProCurve(config-class)# ignore udp 10.30.10.1/24 host
10.20.10.17 range 161 162
```

[operator < tcp-src-port | udp-src-port >]

To specify a TCP or UDP source port number as a match criteria, enter a comparison operator from the following list with a TCP/UDP port number or well-known port name immediately after the source-address value in the command.

Comparison Operators:

- **eq** < tcp/udp-port-number > — “Equal To” matches a packet with the same TCP or UDP source port number as < tcp/udp-port-number >.
- **gt** < tcp/udp-port-number > — “Greater Than” matches any packet with a TCP or UDP source port number greater than < tcp/udp-port-number >.
- **lt** < tcp/udp-port-number > — “Less Than” matches any packet with a TCP or UDP source port number less than < tcp/udp-port-number >.

- **neg** < tcp/udp-port-number > — “Not Equal” matches any packet with a TCP or UDP source port number that is not equal to < tcp/udp-port-number >.
- **range** < start-port-number > < end-port-number > — Matches any packet with a TCP or UDP source port number in the range < start-port-number > to < end-port-number >.

TCP/UDP Well-Known Source-Port Names and Numbers

— Enter a comparison operator with the source TCP or UDP port number used by the applications you want to match. Valid port numbers are from **0** to **255**.

You can also enter well-known TCP or UDP port names as an alternative to the corresponding port number; for example:

- **TCP**: bgp, dns, ftp, http, imap4, ldap, nntp, pop2, pop3, smtp, ssl, telnet
- **UDP**: bootpc, bootps, dns, ntp, radius, radius-old, rip, snmp, snmp-trap, tftp

To display a list of valid TCP/UDP source ports, type **?** after you enter an operator.

[operator < tcp-dest-port [established] [tcp-flag [tcp-flag ...]] | udp-dest-port >]

To specify a TCP or UDP destination port number as a match criteria, enter a comparison operator with a TDP/UDP port number or well-known port name immediately after the destination-address value in the command.

Note: The optional **established** and < tcp-flag > values apply only to TCP destination-port criteria.

TCP/UDP Well-Known Destination-Port Names and Numbers

— The same operators, port numbers and well-known names are supported for TCP/UDP destination-port match criteria as for TCP/UDP source-port criteria. To display a list of valid TCP/UDP destination ports, type **?** after you enter an operator.

[established]

(Optional) Applies only to TCP destination-port match criteria and matches only on the TCP Acknowledge (ACK) or Reset (RST) flags.

The **established** keyword ignores the synchronizing packet associated with the establishment of a TCP connection in one direction on a port or VLAN, and matches all other IP traffic in the opposite direction.

*For example, a Telnet connection requires TCP traffic to move both ways between a host and the target device. If you configure a match statement for inbound Telnet traffic, policy actions are normally applied to Telnet traffic in both directions because responses to outbound requests are also matched. However, if you enter the **established** option, inbound Telnet traffic arriving in response to outbound Telnet requests is matched, but inbound Telnet traffic trying to establish a connection is not matched.*

`[tcp-flag [tcp-flag ...]]`

*(Optional) Applies only to TCP bit settings in packets destined to a TCP destination port configured as match criteria (with the **operator** <tcp-dest-port>parameter) and can be one or more of the following values:*

- **ack** — “Acknowledge” matches TCP packets with the ACK flag.
- **fin** — “Finish” matches TCP packets with the FIN flag.
- **rst** — “Reset” matches TCP packets with the RST bit set.
- **syn** — “Synchronized” matches TCP packets with the SYN flag.

Using CIDR Notation for IPv4/IPv6 Addresses

You can use CIDR (Classless Inter-Domain Routing) notation to enter an IPv4 mask-length or an IPv6 prefix-length with a source and destination address that are used as match criteria in a match/ignore statement. The switch interprets the IP address with CIDR notation to compute the range of corresponding IP source or destination addresses in packet headers that are considered to be a match for the traffic class.

When the switch uses a match/ignore statement to compare an IP address and corresponding mask/prefix length to the IP source/destination address carried in a packet, the IPv4 mask-bit settings and IPv6 prefix-bit settings select packets in different ways.

- An IPv4 mask length creates a mask in which:
 - A mask-bit setting set to **0** (“off”) requires the corresponding bit in a packet’s IPv4 source/destination address to be the same binary value as the mask-bit in the matching IPv4 source/destination address.
 - A mask-bit setting set to **1** (“on”) is used as a wildcard and allows the corresponding bit in a packet’s IPv4 source/destination address to be either binary value (0 or 1).

Table 9-1. How CIDR Notation is Used with IPv4 SA/DA Match Criteria

IPv4 Source/Destination Address Used with CIDR Notation in a Match/Ignore Statement	Resulting Mask	Range of IPv4 Addresses Selected by the Match Criteria
10.38.240.125/15	0.1.255.255	The leftmost 15 bits must match; the remaining bits are wildcards.
10.38.240.125/20	0.0.15.255	The leftmost 20 bits must match; the remaining bits are wildcards.
10.38.240.125/21	0.0.7.255	The leftmost 21 bits must match; the remaining bits are wildcards.
10.38.240.125/24	0.0.0.255	The leftmost 24 bits must match; the remaining bits are wildcards.
18.38.240.125/32	0.0.0.0	All bits must match.

- An IPv6 prefix-length creates a mask in which:
 - A mask-bit setting set to **1** (“on”) requires the corresponding bit in a packet’s IPv6 source/destination address to be the same binary value as the mask-bit in the matching IPv6 source/destination address.
 - A mask-bit setting set to **0** (“off”) is used as a wildcard and allows the corresponding bit in a packet’s IPv6 source/destination address to be either binary value (0 or 1).

Table 9-2. How CIDR Notation is Used with IPv6 SA/DA Match Criteria

IPv6 Source/Destination Address Used with CIDR Notation in a Match/Ignore Statement	Resulting Mask	Range of IPv6 Addresses Selected by the Match Criteria
2001:db8:0:7::5/64	FFFF:FFFF:FFFF:FFFF::	The leftmost 64 bits must match; the remaining bits are wildcards.
2001:db8:0:7::5/72	FFFF:FFFF:FFFF:FFFF:FF00::	The leftmost 72 bits must match; the remaining bits are wildcards.
2001:db8::244:17ff:feb6:d37d/126	FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFC	The first 126 bits must match; the “C” value in the mask allows four possible combinations (D37C, D37D, D37E, and D37F) in the last block of a matching IPv6 address.
2001:db8:0:7:af:e2:c1:5/128	FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF	All bits must match.

Note

Although IPv4 and IPv6 masks are applied in opposite directions:

- An IPv4 mask-length is applied from right to left, starting from the rightmost bits.
- An IPv6 prefix-length is applied from left to right, starting from the leftmost bits.

The behavior of IPv4 and IPv6 masks as match criteria and wildcards is the same.

Example of How IPv4 Mask Bit Settings Define a Match. For this example, the following configuration exists:

- A match statement in a class configuration uses an IPv4 source-address/mask-length of 10.38.31.125/21. The mask-length of 21 results in an IPv4 mask of 0.0.7.255. In the second octet of the mask, 7 means that the rightmost three bits are “on”, or “1” (see the “Mask for SA” row in Table 9-3).
- The second octet of the corresponding source address is 31, which means that the rightmost five bits are “on”, or “1” (see the SA in Match Statement row in Table 9-3).

In this example, a match occurs when the second octet of the SA in a packet being classified has a value in the range of 24 (binary 00011000) to 31 (binary 00001111), as shown in the last row in Table 9-3.

Table 9-3. Example of How the IPv4 Mask Defines a Match

Location of Octet	Bit Position in the Octet							
	128	64	32	16	8	4	2	1
SA in match statement	0	0	0	1	1	1	1	1
Mask for SA	0	0	0	0	0	1	1	1
Bits in the corresponding octet of a packet's SA that must exactly match	0	0	0	1	1	0/1	0/1	0/1
<p>The shaded area indicates the bits in the packet that must exactly match the bits in the source IPv4 address in the match/ignore statement.</p> <ul style="list-style-type: none"> • If a mask bit is “1”(wildcard value), the corresponding bits in a source/destination address in an IPv4 packet header can be any value. • If a mask bit is “0”, the corresponding bits in a source/destination address must be the same value as in the IPv4 address in the match/ignore statement. <p>Note: This example covers only one octet in an IPv4 address used as a match criterion. The mask in a match/ignore statement may apply a packet filter to all four octets of a source/destination address in IPv4 packet headers.</p>								

Example of How IPv6 Mask Bit Settings Define a Match. Figure 9-4 shows an example in which an IPv6 prefix-length of 126 is used to select four IPv6 addresses in a match statement. The specified source IPv6 address is: **2001:DB8:0000:0000:244:17FF:FEB6:D37D**. The IPv6 prefix-length (/126) results in the IPv6 mask: **FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFC**.

	1st Block	2nd Block	3rd Block	4th Block	5th Block	6th Block	7th Block	8th Block	Manager- or Operator-Level Access
IPv6 mask for /126 prefix	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFC	The "F" value in the first 126 bits of the mask specifies that only the exact value of each corresponding bit in an IPv6 address is allowed. However, the binary equivalent (1100) of the "C" value in the mask allows four possible combinations (D37C, D37D, D37E, and D37F) in the last block of a matching IPv6 address.
IPv6 address	2001	DB8	0000	0000	244	17FF	FEB6	D37D	

Figure 9-4. Example: Mask for Matching Four IPv6 Devices

Figure 9-5 shows the "on" and "off" settings in the last block of the resulting IPv6 mask that determine the matching IPv6 addresses. In this mask, all bits except the last two are set to 1 ("on") and must be the same in an IPv6 address. The binary equivalent of hexadecimal **C** is 1100, which allows the last two bits to differ.

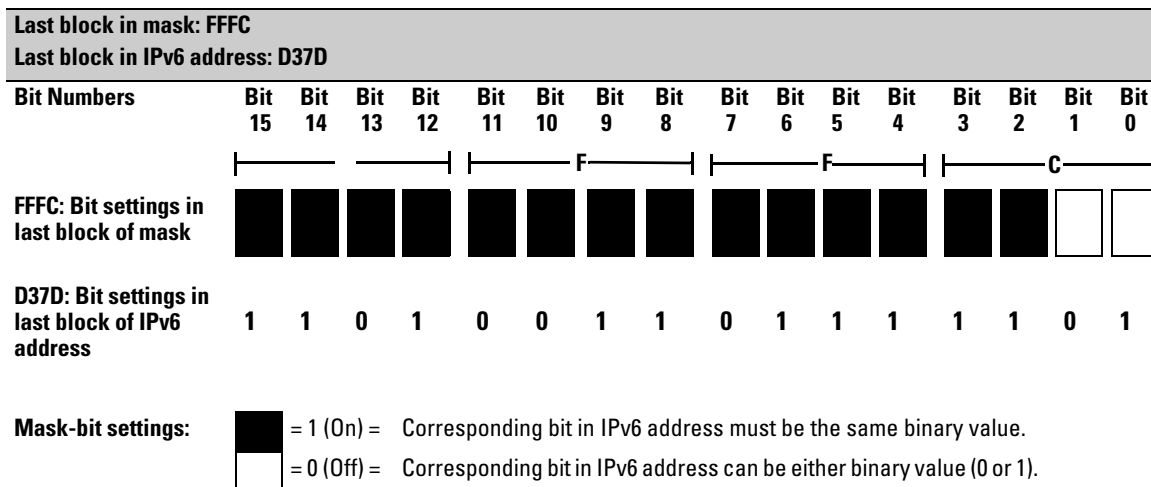


Figure 9-5. Example: How a Mask Determines Four Authorized IPv6 Manager Addresses

Figure 9-6 shows how the binary equivalent (1100) of the “C” value in the last block of the resulting IPv6 mask supports four possible combinations (D37C, D37D, D37E, and D37F) in the last block of a matching IPv6 address. Therefore, the IPv6 mask that results from a /126 prefix-length matches inbound traffic from four IPv6-based devices.

	1st Block	2nd Block	3rd Block	4th Block	5th Block	6th Block	7th Block	8th Block
IPv6 mask	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFC
IPv6 address entered with a “match” command	2001	DB8	0000	0000	244	17FF	FEB6	D37D
Other matching IPv6 addresses	2001	DB8	0000	0000	244	17FF	FEB6	D37C
	2001	DB8	0000	0000	244	17FF	FEB6	D37E
	2001	DB8	0000	0000	244	17FF	FEB6	D37F

Figure 9-6. Example: How Hexadecimal C in an IPv6 Mask Matches Four IPv6 Addresses

CIDR Notation. For more detailed information on how to use CIDR notation to specify masks in match criteria, refer to the “How an ACE Uses a Mask To Screen Packets for Matches” section in the *Access Control Lists (ACLs)* chapter in the *Access Security Guide*.

Resequencing Match/Ignore Statements

In the class configuration context (see “Creating a Traffic Class” on page 9-4), you can use the **resequence** command to reconfigure the number at which the first match/ignore statement in the class starts, and reset the interval used to number other match/ignore statements.

Resequencing match/ignore statements is useful when you want to insert a new match/ignore statement between two numbered entries (see Figure 9-7).

Context: Class configuration

Syntax: `resequence < seq-number > < interval >`

Resets the sequence numbers for all match/ignore statements in the class.

< seq-number > : *Specifies the sequence number of the first match/ignore statement in the class. Default: 10.*

< interval > : *Specifies the interval between sequence numbers of match/ignore statements in the class to allow additional match/ignore statements to be inserted. Default: 10.*

To view the current sequence numbering in a class, enter the **show class < ipv4 | ipv6 > < classname >** command.

The following example shows how to resequence a class configuration so that you can insert new match/ignore statements between sequentially numbered statements. In this example, the resequenced class contains two additional match/ignore statements and rennumbers the criteria with an interval of 10.

```
ProCurve(config)# show class ipv4 My-devices

Class "My-devices"
  1 match 10.10.10.25 0.0.0.0
  2 ignore 10.10.10.1 0.0.0.255
  3 ignore 10.20.10.1 0.0.0.255
  4 match 0.0.0.0 255.255.255.255
  exit
. . .
ProCurve(config)# class ipv4 My-devices
ProCurve(config-class)# resequence My-devices 10 10
ProCurve(config-class)# 15 match 10.10.10.2 0.0.0.255
ProCurve(config-class)# 25 match 10.20.10.1 0.0.0.255
ProCurve(config-class)# exit
ProCurve(config)# show class My-devices

Class "My-devices"
  10 match 10.10.10.25 0.0.0.0
  15 match 10.10.10.2 0.0.0.255
  20 ignore 10.10.10.1 0.0.0.255
  25 ignore 10.20.10.1 0.0.0.255
  30 ignore 10.20.10.2 0.0.0.255
  40 match 0.0.0.0 255.255.255.255
  exit
```

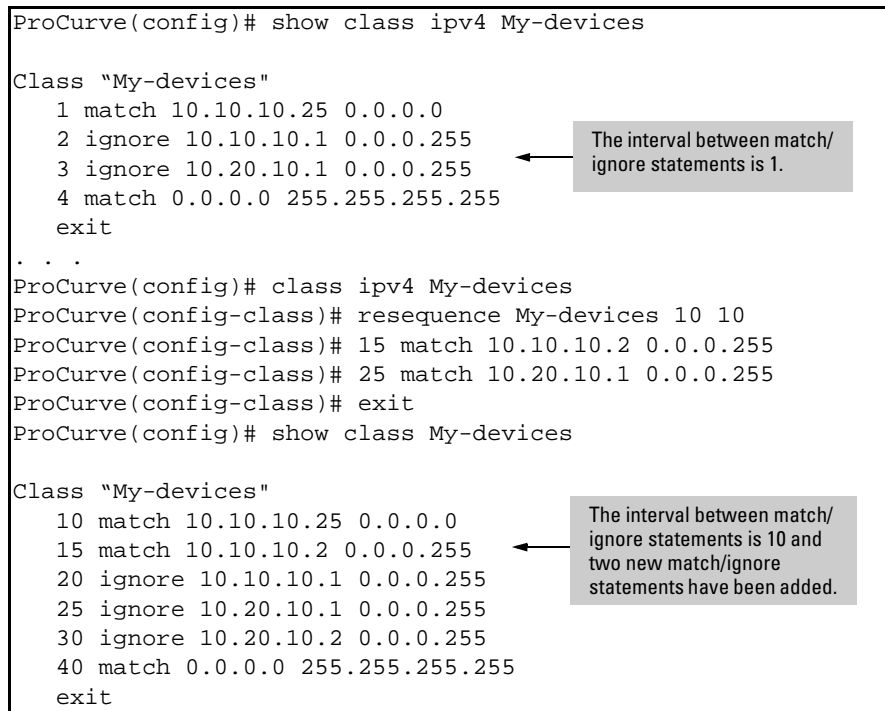


Figure 9-7. Example of Resequencing a Class Configuration

Creating a Service Policy

In the *classifier-based* configuration model, the service policy you create for one or more traffic classes is always relative to a software feature, such as QoS or port mirroring. The software feature must support class and policy configuration. Each feature supports different actions for managing selected packets.

For example, QoS policies support QoS-specific actions, such as rate-limiting, 802.1p-priority, IP-precedence, and DSCP-codepoint assignment. Port mirroring policies support mirror-destination assignment for matching packets.

1. To create a service policy that performs feature-specific actions on selected packets, enter the **policy < feature-name >** command from the global configuration context.

Context: Global configuration

Syntax: [no] policy <feature-name > <policy-name >

Defines the name of a service policy and enters the policy configuration context, where:

<feature-name > is a keyword that identifies a ProCurve software feature that supports classifier-based configuration (for example, qos or mirror).

<policy-name > is a text string (64 characters maximum).

A traffic policy consists of one or more actions that are configured for each class of traffic. The configured actions are executed on packets that match a **match** statement in a class. No policy action is performed on packets that match an **ignore** statement. You can configure multiple classes in a policy.

2. To configure the actions that you want to execute on packets that match the **match** criteria in a specified class, enter one or more **class action** commands from the policy configuration context.

Context: Policy configuration

Syntax: [no] [seq-number] class < ipv4 | ipv6 > <classname >
action <action-name > [action <action-name > ...]

Defines the action(s) to be applied on a pre-configured IPv4 or IPv6 traffic class when a packet matches the match criteria in the class.

You can enter multiple class-action statements for the same class. Note that the actions supported for a class command differ according to the feature-specific policy (for example, QoS or mirroring) configured with the policy command in Step 1.

[no] [seq-number] class < ipv4 | ipv6 > <classname >

- **[seq-number]** — *The (optional) seq-number parameter sequentially orders the class-action statements in a policy configuration. Actions are executed on matching packets in numerical order. Default: Class-action statements are numbered in increments of 10, starting at 10.*

- **class < ipv4 | ipv6 > <classname >** — Defines the preconfigured class on which the actions in a class-action statement are executed, and specifies whether the class consists of IPv4 or IPv6 traffic. The class name is a text string (64 characters maximum).

Note: You can configure multiple class-action statements to include different classes in a policy. The execution of actions is performed in the order in which the class-actions are numerically listed.

action <action-name > [action <action-name > ...]

The **action** keyword configures the action specified by the **action-name** parameter. The action is executed on any packet that matches the **match** criteria in the class. The action is not executed on packets that match **ignore** criteria. You can configure more than one action for a class.

The complete **no** form of the **class action** command or the **no <seq-number >** command removes an action from the policy configuration.

For information on the exact actions supported by each classifier-based software feature, refer to the appropriate chapter in the ProCurve documentation set, as described in [“”](#) on page 9-33.

Be sure to enter a class and its associated actions in the *precise order* in which you want packets to be checked and handled by **class action** commands.

3. (Optional) To configure a default class, enter the **default-class** command and specify one or more actions to be executed on packets that are not matched and not ignored.

Context: Policy configuration

Syntax: [no] default-class action <action-name > [action <action-name > ...]

Configures a default class to be used to execute one or more actions on packets that are not matched nor ignored in any of the class configurations in a policy.

*The **default-class action** command supports only the feature-specific commands supported in the **class action** command.*

The default class manages packets that do not match the **match** or **ignore** criteria in all classes in a policy, and otherwise would have no actions performed on them.

The default class differs from other classes because it contains no match/ignore statements and uses implicit **match ipv4 any any** and **match ipv6 any any** statements to manage all unmatched packets. If you do not configure a default class, unmatched and unignored packets are transmitted without an action performed on them.

4. Enter the **exit** command to exit the policy configuration context.

To display a policy configuration, enter the **show policy < feature-name > < policy-name >** command (see Figure 9-9), where **< feature-name >** is a software feature (such as QoS or port mirroring) that supports classifier-based configuration.

To edit a policy configuration, re-enter the policy context (**policy** command) and modify class-action statements as described in “Modifying Classes in a Policy” on page 9-28.

To resequence the order in which class-action statements are listed, enter the **resequence** command (see “Resequencing Classes in a Policy” on page 9-29).

Example. In the following QoS policy configuration, matching HTTP packets are rate-limited to 10000 kbps and assigned an 802.1p (CoS) priority of 3 in their Layer 2 VLAN headers. All unmatched packets are managed by the default class, which assigns a slightly higher 802.1p priority (4) and a new DSCP codepoint (5)

```
ProCurve(config)# class ipv4 http
ProCurve(class-config)# match tcp any any eq 80
ProCurve(class-config)# match tcp any any eq 8080
ProCurve(class-config)# exit
ProCurve(config)# policy qos RateLimitPrioritizeSuspectTraffic
ProCurve(policy-config)# class ipv4 http action rate-limit kbps 10000
ProCurve(policy-config)# class ipv4 http action priority 3
ProCurve(policy-config)# default-class action priority 4 dscp 5
ProCurve(policy-config)# exit
```

Figure 9-8. Example of a Policy Configuration

As shown in Figure 9-8, a policy configuration requires a feature-specific **policy** command to identify the software feature used to manage one or more traffic classes:

- To configure a QoS policy, use the **policy qos** command as described in the “Quality of Service” chapter in the *Advanced Traffic Management Guide*.

- To configure a mirroring policy, use the **policy mirror** command as described in the *Monitoring and Analyzing Switch Operation* appendix in the *Management and Configuration Guide*.

Modifying Classes in a Policy

You can modify the classes and class-action statements in a policy configuration without removing them from the policy:

- To modify the match/ignore statements in a class, enter the class-configuration context with the **class <ipv4 | ipv6> <classname>** command, and make the necessary changes by removing, resequencing, or replacing existing statements. (To display a class configuration, enter the **show class <ipv4 | ipv6> <classname>** command as shown in Figure 9-7.)

When you exit class configuration context, the changes are automatically saved and applied to existing policy configurations on the switch that use the class *if the policies have not been applied to an interface*. If a policy has already been applied to an interface, the editing changes are not accepted and an error message is displayed.

- To modify the class-action statements in a policy, enter the policy-configuration context with the **policy <feature-name> <policy-name>** command. (To display a policy configuration, enter the **show policy <feature-name> <classname>** command as shown in Figure 9-9.) Then do one of the following:
 - You can enter a new class-action statement. If you do not enter a sequence number, the new class-action statement is inserted at the end of the policy configuration.
 - To remove a class-action statement from a policy configuration, enter the **no <sequence-number>** command or the complete form of the **no class ... action** command.
 - To resequence the order in which class-action statements are listed, enter the **resequence** command (see “Resequencing Classes in a Policy” on page 9-29).
 - To replace an existing class-action statement, enter the **no <sequence-number>** command to delete the entry, and re-enter a complete **class <ipv4 | ipv6> <classname> action <action-name>** or **default-class action <action-name>** command.

When you exit the policy-configuration context, the changes are automatically saved and applied to the policy configuration *if the policy has not been applied to an interface*. If the policy has already been applied to an interface, the editing changes are not accepted and an error message is displayed.

Resequencing Classes in a Policy

In policy configuration mode, you can use the **resequence** command to reconfigure the number at which the first class-action statement starts, and reset the interval used to number other class-actions.

Resequencing classes is useful when you want to insert a new class (with its associated actions) between two numbered entries.

Context: Policy configuration

Syntax: resequence < seq-number > < interval >

Resets the sequence numbers for all classes in the policy.

< seq-number >: *Specifies the sequence number of the first class in the policy. Default: 10.*

< interval >: *Specifies the interval between sequence numbers of classes in the policy to allow additional match/ignore statements to be inserted. Default: 10.*

Note

When you resequence classes in a policy, the default class always remains as the last class.

To view the current class numbering in a policy, enter the **show policy < feature-name > < policy-name >** command.

The following example shows how to resequence a policy configuration after displaying its contents. The resequenced policy allows you to add a new class-action statement between entries 100 and 200.

```
ProCurve(config)# show policy My-devices
Policy "My-devices"
  10 Class "My-devices" priority 7
  11 Class "http" rate-limit 1000
...
ProCurve(config)# policy My-devices
ProCurve(policy-config)# resequence My-devices 100 10
ProCurve(policy-config)# 150 class ipv4 voice priority 3
ProCurve(policy-config)# exit
ProCurve(config)# show policy My-devices

Policy "My-devices"
  100 Class "My-devices" priority 7
  150 Class "voice" priority 3
  200 Class "http" rate-limit 1000
```

The interval between class-action statements is 1.

The interval between class-action statements is 100, and a new statement has been added.

Figure 9-9. Example of Resequencing a Policy Configuration

Applying a Service Policy to an Interface

To apply the feature-specific service policies you create to an inbound port or VLAN interface, use the **interface service-policy in** or **vlan service-policy in** command.

The following service-policy restrictions apply to all software features:

- A service policy is supported only on inbound traffic.
- Only one feature-specific policy (for example, QoS or mirroring) is supported on a port or VLAN interface.
- If you apply a policy to a port or VLAN interface on which a policy of the same type (for example, QoS) is already configured, an error message is displayed. The new policy does not overwrite the existing one.

Before you can apply a new policy, you must first remove the existing policy with the **no interface service-policy in** or **no vlan service-policy in** command.

Because only one policy of each type is supported on a port or VLAN interface, ensure that the policy you want to apply contains all the required classes and actions for your configuration.

Note

If ICMP rate-limiting is already configured on a port, a service policy cannot be applied to the port until you disable the ICMP rate-limiting configuration.

If you want to apply a service policy to the port, you can maintain ICMP rate-limiting by configuring a QoS policy in which you add the necessary **match** statements for ICMP packets to a class configuration and configure a **rate-limit** action for the class in the policy configuration.

For information on globally-configured ICMP, refer to the “Configuring ICMP” section in the “Configuring IP Parameters for Routing Switches” chapter in the *Multicast and Routing Guide*.

To apply a service policy on a port or VLAN interface, enter one of the following commands from the global configuration context.

Context: Global configuration

Syntax: interface <port-list> service-policy <policy-name> in

Configures the specified ports with a policy that is applied to inbound traffic on each interface.

*Separate individual port numbers in a series with a comma; for example, **a1, b4, d3**. Enter a range of ports by using a dash; for example, **a1-a5**.*

*The policy name you enter must be the same as the policy name you configured with the **policy** command (see “Creating a Service Policy” on page 9-24).*

Context: Global configuration

Syntax: vlan <vlan-id> service-policy <policy-name> in

Configures a policy on the specified VLAN that is applied to inbound traffic on the VLAN interface.

*Valid VLAN ID numbers range from **1** to **4094**.*

*The policy name you enter must be the same as the policy name you configured with the **policy** command (see “Creating a Service Policy” on page 9-24).*

The following example shows how to apply a QoS policy to a port range and a VLAN interface:

```
ProCurve# interface a4-a5 service-policy RateLimitPrioritizeSuspectTraffic in
ProCurve# vlan 10 service-policy RateLimitPrioritizeSuspectTraffic in
```

Figure 9-4. Example of How to Configure an Interface with a Service Policy

Checking Resource Usage. After you apply a service policy to an interface, use the **show policy resources** command to verify the amount of additional resources used and the amount of resources that are still available on the switch. Classifier-based service policies (such as QoS or mirroring) share the same hardware resources with other software features, such as ACLs, virus throttling, management VLAN, globally configured QoS policies, MAC-based mirroring policies, and so on.

Use the displayed information to decide if you need to re-prioritize current resource usage by reconfiguring or disabling software features to free the resources reserved for less important features. For a detailed explanation of

Classifier-Based Software Configuration Applying a Service Policy to an Interface

the information displayed with the **show policy resources** command, refer to the “Monitoring Resources” appendix in the *Management and Configuration Guide*.

In Figure 9-5, the **show policy resources** command output displays the number of hardware resources (rules, meters, and application port ranges) used by classifier-based QoS and mirroring policies that are currently applied to interfaces on the switch, as well as other software features.

```
ProCurve# show policy resources
```

Resource usage in Policy Enforcement Engine								
Ports	Rules		Rules Used			VT	Mirror	Other
	Available	ACL	QoS	IDM				
1-24	3014	15	11	0	1	0	3	
25-48	3005	15	10	10	1	0	3	
A	3017	15	8	0	1	0	3	

Meters								
Ports	Meters		Meters Used			VT	Mirror	Other
	Available	ACL	QoS	IDM				
1-24	250		5	0			0	
25-48	251		4	0			0	
A	253		2	0			0	

Application Port Ranges								
Ports	Application		Application Port Ranges Used			VT	Mirror	Other
	Available	ACL	QoS	IDM				
1-24	3014	2	0	0			0	
25-48	3005	2	0	0			0	
A	3017	2	0	0			0	

0 of 8 Policy Engine management resources used.

Key:
 ACL = Access Control Lists
 QoS = Device & Application Port Priority, QoS Policies, ICMP rate limits
 IDM = Identity Driven Management
 VT = Virus Throttling blocks
 Mirror = Mirror Policies, Remote Intelligent Mirror endpoints
 Other = Management VLAN, DHCP Snooping, ARP Protection, Jumbo IP-MTU.

Resource usage includes resources actually in use, or reserved for future use by the listed feature. Internal dedicated-purpose resources, such as port bandwidth limits or VLAN QoS priority, are not included.

Figure 9-5. Displaying Policy Resources

Where to Go From Here

Classifier-based service policies are designed to work with your existing globally-configured software settings. While existing software features allow you to globally manage *all* network traffic on a switch or port, classifier-based service policies allow you to zoom in on subsets of network traffic to further manage it on a per-port or per-VLAN basis.

You can use the match criteria described in this chapter across software features to configure classes of traffic for use in feature-specific service policies.

After you decide on the IPv4 and IPv6 network traffic you want to manage, refer to the following chapters for more information about how to configure and use classifier-based quality-of-service and mirroring policies:

- *Quality of Service (QoS)* chapter in the *Advanced Configuration Guide*
- “Traffic Mirroring” section in the *Monitoring and Analyzing Switch Operation* appendix in the *Management and Configuration Guide*

Classifier-Based Software Configuration
Where to Go From Here