CHAPTER 3

The Define and Delete Commands

Use the **define** and **delete** commands to create and delete network-wide constructs used in network management with LightStream 2020 multiservice ATM switches (LS2020 switches). These constructs include filter conditions (for bridge, IP, and IPX traffic), traffic profiles, and multicast groups. The commands are given in the following list, together with the titles of the sections of this chapter in which they are discussed:

define *filter-type ID expression*	Define Traffic Filters
define tprof *ID* **max-rate** *bits/sec* [*arguments*]	Define Traffic Profiles
define mcast *ID port-list*	Define Multicast Groups
delete *filter-type ID* **delete mcast-member** *ID port-list*	Delete Network-Wide Constructs

See also the following types of **set** and **show** commands:

- Traffic filter commands
- VLI workgroup commands
- Spanning-tree bridge commands

The **set** commands are discussed in the chapter entitled “The Set Command,” and the **show** commands are discussed in the chapter entitled “The Show Command.”“

Define Traffic Filters

Use the **define** command to create and modify filters for bridge traffic, IP traffic, or IPX traffic. A maximum of 512 filters can be defined for a node.

define bflt ID expression

Use the **define bflt** *ID expression* command to define a bridge filter. Assign the filter identification number *ID*, in the range 1 – 65535. If *ID* is already in use for a bridge filter, the command overwrites the old filter without warning. Use the **show bflt** command to display currently defined bridge filters. The *expression* argument specifies values for fields in an incoming frame header. If the contents of a field match the value specified for that field in a filter condition, then a specified action is taken with the frame. The *expression* can be a comparison expression or a boolean expression (see the section entitled “Filter Expressions” later in this chapter).

The broadcast rate limit, a port attribute, does not apply to broadcast traffic that matches a custom filter set on the port. Broadcast frames that are recognized by filters are handled by the LS2020 hardware, and are not subject to the broadcast rate limitation.

Note Bridge filter conditions are applied before IP and IPX filter conditions. After a bridge filter condition forwards a packet, an IP or IPX filter can block it.

If you define a filter that applies to an existing flow between a LAN interface and the NP, reset the LAN interface. When the interface returns to service, valid flows are re-established, but flows blocked by the filter are not.

define ipflt ID expression

Use the **define ipflt** *ID expression* command to define an IP filter. Assign a filter identification number *ID* in the range 1 – 65535. If *ID* is already in use for an IP filter, the command overwrites the old filter without warning.

Use the **show ipflt** command to display currently defined IP filters. The *expression* argument specifies values for fields in an incoming frame header. If the contents of a field match the value specified for that field in a filter condition, a specified action is taken with the frame. The *expression* can be a comparison expression or a boolean expression (see the section entitled “Filter Expressions” later in this chapter).

Note Bridge filter conditions are applied before IP and IPX filter conditions. After a bridge filter condition forwards a packet, an IP or IPX filter can block it.

802.3/SNAP encapsulated frames cannot be filtered.

If you define a filter that applies to an existing flow between a LAN interface and the NP, reset the LAN interface. When the interface returns to service, valid flows are re-established, but flows blocked by the filter are not.

define ipxflt ID expression

Use the **define ipxflt** *ID expression* command to define an IPX filter. Assign the filter identification number *ID*, in the range 1 – 65535. If *ID* is already in use for an IPX filter, the command overwrites the old filter without warning. Use the **show ipxflt** command to display currently defined IPX filters. The *expression* argument specifies values for fields in an incoming frame header. If the contents of a field match the value specified for that field in a filter condition, then a specified action is taken with the frame. The *expression* can be a comparison expression or a boolean expression (see the section entitled “Filter Expressions” later in this chapter).

Note Bridge filter conditions are applied before IP and IPX filter conditions. After a bridge filter condition forwards a packet, an IP or IPX filter can block it.

If you define a filter that applies to an existing flow between a LAN interface and the NP, reset the LAN interface. When the interface returns to service, valid flows are re-established, but flows blocked by the filter are not.

Related Commands

Use the following commands together with the **define bflt**, **define ipflt**, and **define ipxflt** commands to manage traffic filters:

- To specify the action to take if a filter is matched, use the **set port** command with a **bflt**, **ipflt**, or **ipxflt** argument. The *priority* argument determines the sequence in which filter conditions are considered for the specified port.
- To display currently defined filters, use the command **show** {**bflt**|**ipflt**|**ipxflt**} [*ID*].
- To display filters associated with a specified port, plus statistics about filtered traffic, use the command **show port** *c.p* {**bflt**|**ipflt**|**ipxflt**} [*ID*].
- To break the association of a filter with a port, use the **set port** *c.p* {**bflt**|**ipflt**|**ipxflt** *ID* **delete** command.
- To delete a traffic filter after all of its associations with ports have been broken (see previous item), use the command **delete** {**bflt**|**ipflt**|**ipxflt**} *ID*.

Filter Expressions

The arguments of a **define bflt**, **define ipflt**, or **define ipxflt** command include *expression*, a filter expression. This argument can be a comparison expression or a boolean expression.

Comparison Expressions

In a comparison expression, the value of an incoming header field is compared with a constant. A comparison expression has one of the following two forms:

```
field operator constant
(field & mask) operator constant
```

The syntax parameters *field*, *operator*, and *constant*, and the *mask* parameter with the bitwise AND operator **&**, are described in the following paragraphs.

Field

The *field* parameter is a built-in identifier for a field in incoming frame or packet headers. The field identifiers are not case sensitive (for example, macsrc is equivalent to macSrc).

The *field* identifiers for bridge filters are as follows:

macSrc	MAC source address (in canonical form)
macDst	MAC destination address (in canonical form)
macProto	MAC protocol type
llcSSAP	LLC source SAP
llcDSAP	LLC destination SAP

snapOUI	SNAP OUI
snapProto	SNAP protocol

MAC addresses must be specified in canonical form (least significant bit first) for FDDI ports as well as for Ethernet ports.

The *field* identifiers for IP filters are as follows:

ipSrc	IP source address
ipDst	IP destination address
ipTOS	IP type of service
portSrc	TCP/UDP source port
portDst	TCP/UDP destination port
ipProto	IP Protocol type

The *field* identifiers for IPX filters are as follows:

ipxDstNw	Destination net
ipxSrcNw	Source net
ipxDstNd	Destination node
ipxSrcNd	Source node
ipxDstSt	Destination socket
ipxSrcSt	Source socket
ipxType	Packet type

Operator

The *operator* parameter is a comparison operator. The comparison operators are as follows:

==	Equal
!=	Not equal
>	Greater than
>=	Greater than or equal
<	Less
<=	Less than or equal

Constant

The constant specified on the right side of a simple comparison expression must be of the appropriate form for the built-in packet header field named on the left side of the comparison expression. The field identifiers are not case sensitive (for example, macsrc is equivalent to macSrc). Table 3-1 lists constants for bridge filters, Table 3-2 lists constants for IP filters, and Table 3-3 lists constants for IPX filters.

Table 3-1 Constants Used for Bridge Filters

Field	Format		Description
macSrc	*xx*:*xx*:*xx*:*xx*:*xx*:*xx*		MAC source address (in canonical form: least significant bit first)
macDst	*xx*:*xx*:*xx*:*xx*:*xx*:*xx*		MAC destination address (in canonical form: least significant bit first)
macProto	0 – 65535	(0 – 0xffff)	MAC protocol type
llcSSAP	0 – 255	(0 – 0xff)	LLC source SAP
llcDSAP	0 – 255	(0 – 0xff)	LLC destination SAP
snapOUI	0 – 16777215 (0 – 0xffffff)		SNAP OUI
snapProto	0 – 65535 (0 – 0xffff)		SNAP Ethernet protocol

Colon-separated values in MAC addresses for the macSrc and macDst fields are hex digits without the leading 0x, but with leading zeros if necessary. The other constants can be entered either as sequences of decimal digits (the default) or as hex digits (with leading 0x).

At the CLI prompt, determine the MAC address with the following command:

```
getsnmp dot1dBaseBridgeAddress.0
```

At the NP O/S prompt, determine the MAC address with the following command:

sysver -fc -s *NP*

In this command, *NP* is the slot number of the NP. The MAC address is labelled “Ethernet address” in the command output.

Table 3-2 Constants Used for IP Filters

Field	Format		Description
ipSrc	*nnn.nnn.nnn.nnn*		IP source address
ipDst	*nnn.nnn.nnn.nnn*		IP destination address
ipTOS	0 – 255	(0 – 0xff)	IP type of service
ipProto	0 – 255	(0 – 0xff)	IP protocol type
portSrc	0 – 65535	(0 – 0xffff)	TCP/UDP source port
portDst	0 – 65535	(0 – 0xffff)	TCP/UDP destination port

Dot-separated values in IP addresses are decimal digits without leading zeros. Other constants can be entered as sequences of decimal digits (the default) or hex digits (with leading **0x**), with leading zeroes if necessary.

Table 3-3 Constants for IPX Filters

Field	Format	Description
ipxDstNw	0 – 4294967295 (0 – 0xffffffff)	IPX destination network
ipxSrcNw	0 – 4294967295 (0 – 0xffffffff)	IPX source network
ipxDstNd	*xx*:*xx*:*xx*:*xx*:*xx*:*xx*	IPX destination node
ipxSrcNd	*xx*:*xx*:*xx*:*xx*:*xx*:*xx*	IPX source node
ipxDstSt	0 – 65535(0 – 0xffff)	IPX destination socket
ipxSrcSt	0 – 65535(0 – 0xffff)	IPX source socket
ipxType	0 – 255(0 – 0xff)	IPX packet type

In Table 3-3, *x* denotes a hex digit (with no leading **0x**). Colon-separated values in IPX addresses ipxDstNd and ipsSrcNd are hex digits without the leading 0x, but with leading zeros if necessary. Other constants can be entered as sequences of decimal digits (the default) or hex digits (with leading 0x).

The mask Parameter and the & Operator

You can use the *mask* parameter in a comparison expression to mask the *field* value in the incoming header field. This parameter is used in a C-style bitwise AND expression of the form (*field* **&** *mask*). For each bit in the field that you want to check, there should be a 1 in the mask, and for each "don't care" bit there should be a 0 in the mask. Under the == operator, described below, a 0 ("don't care") in the masked *field* value can be matched in the *constant* value only by a corresponding 0 ("don't care").

For example, with the mask 0xfff0 in the following expression, the operator == ignores the least significant digit of the macProto field, and matches a zero as the least significant digit of the constant 0x8130:

```
(macProto & 0xfff0) == 0x8130
```

Table 3-4 shows the different results of applying mask **0xfff0** or mask **0xfffe** to field value **0x8137** (the IPX value in the MAC or SNAP protocol field):

Table 3-4 Effects of Mask fff0 and Mask fffe on a Value

Mask	f	f	f	0	f	f	f	e
	1111	1111	1111	0000	1111	1111	1111	1110
Field	8	1	3	7	8	1	3	7
	1000	0001	0011	0111	1000	0001	0011	0111
Result	8	1	3	0	8	1	3	6
	1000	0001	0011	0000	1000	0001	0011	0110

Boolean Expressions

In a boolean expression, boolean operators are used to combine two or more expressions of any type. The syntax is as follows:

```
(expression) boolean-operator (expression) […]
```

Here, *boolean-operator* is **&&** (logical AND) or **||** (logical OR), and *expression* can be either a comparison expression or another boolean expression. Expressions are evaluated left to right, but because parenthesized expressions are resolved first you can sometimes force a different evaluation sequence.

Field Values

The following tables list the most commonly used values for various fields.

TCP/UDP Ports and IPX Sockets

Certain port numbers are established by Internet convention as well-known ports. The most commonly used well-known ports for the TCP/UDP source or destination port, and for the IPX source or destination socket, are as follows:

5	RJE	**23**	TELNET	**75**	Private dial
7	ECHO	**25**	SMTP	**77**	Private RJE
9	DISCARD	**37**	TIME	**79**	FINGER
11	USERS	**39**	RLP	**95**	SUPDUP
13	DAYTIME	**42**	NAMESERVER	**101**	HOSTNAME
15	NETSTAT	**43**	NICNAME	**102**	ISP-TSAP
17	QUOTE	**53**	DOMAIN	**113**	AUTH
19	CHARGEN	**67**	BOOTPS	**117**	UUCP-PATH
20	FTP-DATA	**68**	BOOTPC	**123**	NTP
21	FTP	**69**	TFTP		

IP Protocol Type

The expected values for the IP protocol type are as follows:

1	ICMP	**13**	ARGUS	**25**	LEAF1-1
2	IGMP	**14**	EMCON	**26**	LEAF1-2
3	GGP	**15**	XNET	**27**	RDP
4	—	**16**	CHAOS	**28**	IRTP
5	ST	**17**	UDP	**29**	ISO-TP4
6	TCP	**18**	MUX	**30**	NETBLT
7	UCL	**19**	DCN-MEAS	**31**	MFE-NSP
8	EGP	**20**	HMP	**32**	MERIT-INP
9	IGP	**21**	PRM	**33**	SEP

10	BBN-RC-MON	**22**	XNS-IDP	**34**	3PC
11	NVP-II	**23**	TRUNK-1	**61**	Host internal
12	PUP	**24**	TRUNK-2	**62**	CFTP

Bridge MAC and SNAP Protocol Type

Typical values for the MAC and SNAP protocol type fields for bridge filters are as follows:

0800	IP	**80f3**	Appletalk ARP
809B	Appletalk	**8137**	IPX

IP TOS and IPX Packet Type

The most common values for the IP type of service and the IPX packet type are as follows:

0x00	Unknown	**0x03**	Error packet
0x01	RIP	**0x04**	PEP
0x02	Echo packet	**0x05**	SPP

Examples

The following command defines bridge filter 20, which matches a value greater than or equal to 1 in the LLC source SAP field:

```
cli> define bflt 20 llcSSAP >= 1
```

The following command defines bridge filter 30, which matches any frame from the specified source MAC address, so long as the frame is not going to the specified MAC destination address:

```
cli> define bflt 30 (macSrc == 00:dd:00:00:00:12) && (macDst != \
00:dd:00::00:76)
```

The following command defines bridge filter 40, which matches a MAC source address whose first two fields are 00:dd, regardless of the values of the remaining four fields:

```
cli> define bflt 40 (macSrc & ff:ff:00:00:00:00) == 00:dd:00:00:00:00
```

The following command defines bridge filter 50, which matches a header whose MAC source address begins with the two fields 00:dd, and whose MAC destination address does not.

```
cli> define bflt 50 ((macSrc & ff:ff:00:00:00:00) == 00:dd:00:00:00:00) \
&& ((macDst & ff:ff:00:00:00:00) != 00:dd:00:00:00:00)
```

The following command defines IP filter 60, which matches any packet from IP network 186:

```
cli> define ipflt 60 (ipSrc & 255.0.0.0) == 186.0.0.0
```

The following command defines IPX filter 70 in the same way that bridge filter 50 was defined in the previous example; it matches a header whose MAC source address begins with the two fields 00:dd, and whose MAC destination address does not:

```
cli> define ipxflt 70 ((ipxDstNd & ff:ff:00:00:00:00) == 00:dd:00:00:00:00)\
&& ((ipxDstNd & ff:ff:00:00:00:00) != 00:dd:00:00:00:00)
```

Define Traffic Profiles

Use the **define tprof** command to create or modify a traffic profile. A traffic profile is a set of type-of-service attributes that can be associated with a traffic flow when the flow is created. A flow is created by assigning a filter to an input port, optionally with an associated traffic profile and/or multicast group (see the commands **set port** *c.p* **bflt**, **set port** *c.p* **ipflt**, and **set port** *c.p* **ipxflt** in the chapter entitled "The Set Command").

Note The **define tprof** command requires CLI protected mode for all of its arguments except **max-rate** and **insured-rate**. (See the **protected** command in the chapter entitled "CLI Control Commands.")

A traffic profile is a network-wide construct, that is, each profile should be unique across the network. For consistency and for clarity in network management, if you define a traffic profile on one node, you should define it identically on all nodes in the network; it does not matter whether or not it is currently used on a given node.

---

define tprof ID max-rate

Use the **define tprof** *ID* **max-rate** {*bits/sec* | **default**} command to define or modify a traffic profile whose identification number is *ID*.

All arguments of this command except for the **max-rate** argument are optional.

Note Because of the length and number of command arguments, this command can exceed the 80-column width assumed for the screen. See the *LightStream 2020 Network Operations Guide* for a description of how the CLI displays command lines that are too long for the screen.

Default Traffic Profile

The default traffic profile has the following parameter values:

insured-burst	0 bytes
insured-rate	0 bits/sec
max-burst	32000 bytes
max-rate	Multicast: 500,000 bps Unicast: 1.2 * smallest bottleneck in path
principal-service-type	Insured
secondary-scale	1%
transmit-priority	0

max-rate

The **max-rate** *bits/sec* argument is required; it does not require protected mode. The maximum rate (in bits per second) is the upper bound on the rate of all traffic (insured and non-insured) allowed to enter the LightStream 2020 network, congestion permitting. The range is 64,000 – 100,000,000 *bits/sec*. It must be greater than the insured rate

With the string **default** as the value, the software determines the maximum rate at the time that the profile is assigned to a port. If no profile is specified, the system begins this calculation with the maximum rate allowed by the network.

- For a unicast (point-to-point) flow, the default is 1.2 times the slowest bottleneck in the circuit (including source and destination LAN ports), up to a limit as follows:
  - 12,000,000 bps If the circuit originates at an Ethernet port.
  - 120,000,000 bps If the circuit originates at an FDDI port.
- For a multicast (point-to-multipoint) flow, the default maximum rate is 500,000 bps.

Note Use a value appropriate for the slowest link in the connection. For example, for an Ethernet-to-FDDI flow, set the maximum rate to 10,000. If an incorrect value is set, the software blocks the flow and issues a trap.

---

insured-burst

Use the **insured-burst** *bytes* parameter to set the upper bound on the nonsharable bandwidth that the LAN flow is permitted to use in bursts, that is, the amount by which traffic on the LAN flow is permitted to exceed the insured rate (see **insured-rate**). The range is 0 – 64,000. In the default profile, this parameter is set to 0 bytes. This value cannot exceed the value of the max-burst parameter.

insured-rate

Use the **insured-rate** *bits/sec* parameter to set the upper bound on the nonsharable bandwidth that the LAN flow is permitted to use in a sustained way. The range is 0 – 100,000,000 bps. This parameter is set to 0 bps in the default profile. It must be less than the max-rate parameter. This parameter does not require protected mode.

max-burst

Use the **max-burst** *bytes* parameter to set the upper bound on bursts of traffic allowed to enter the network from the LAN interface. This bound determines the amount by which this traffic is permitted to exceed the maximum rate (see **max-rate**). The range is 0 – 64,000. In the default profile, this parameter is set to 32,000 bytes. This value must be at least as great as the value of the insured-burst parameter.

principal-service-type

Use the **principal-service-type** {**guaranteed** | **insured**} parameter to set the relative importance of the LAN flow in the face of local congestion (cell-drop eligibility). This value indicates priority order for selective cell discard of best-effort traffic. In the default profile, this parameter is set to **insured**.

secondary-scale

Use the **secondary-scale** *value* parameter to set the scaling factor for secondary bandwidth. This value is a fraction of the secondary portion of a VC's bandwidth, also known as the excess rate. LS2020 software uses this fraction to apportion traffic on trunk lines according to available bandwidth. The fraction is multiplied by the excess rate, and the resultant number of bits per second is deducted from the available bandwidth on the trunk line. The result is a new value for available bandwidth on that trunk line.

The range is 0 – 109. A value in the range 0 – 100 is interpreted as a percent (*x*/100). A value in the range 101 – 109 yields tenths of a percent according to the formula (*x* – 100)/1000. For example, 2 means 2%, and 102 means 0.2%. In the default profile, this parameter is set to 1, yielding a 1% scaling factor.

transmit-priority

Use the **transmit-priority** {**0** | **1**} parameter to set a value indicating the relative priority that this traffic has across the VC, end to end. This value is a factor in determining how cells are queued at each node along the VC. It also contributes to cell loss calculations. A value of 0 is lower priority, and 1 is higher. The default is 0.

Related Commands

Use the following commands with the **define** command to manage traffic filters:

- Use the **set port** *c.p* {**bflt** | **ipfilt** | **ipxflt**} commands to associate one and only one profile with a filter while assigning the filter to a port. The action of the filter must be `forward`.
- Use the **show tprof** [*ID*] command to display traffic profiles.
- Use the **show tprof default** command to display the default traffic profiles.
- Use the **delete tprof** *ID* command to delete traffic profiles.

Examples

In the following example, the **define tprof** command is used to define traffic profile 7:

```
*cli> define tprof 7 max-rate 77000 principal-service-type insured
*cli>
```

Traffic profile 7 is defined here with 77,000 bps as the maximum rate and insured as the principal service type. It uses the defaults for insured rate, maximum burst, insured burst, secondary scale, and transmit priority.

In the following example, the **define tprof** command is used to define traffic profile 16. (The command is shown wrapped to two lines. For more information about how the CLI displays lines that are too long for the screen, see the *LightStream 2020 Network Operations Guide*.)

The first attempt to use the command fails:

```
*cli> define tprof 16 max-rate 64000 insured-rate 32000 insured-burst 40000
principal-service-type guaranteed
Insured burst cannot be greater than 32,000 when no max-burst is entered
(because max-burst DEFAULT value is 32,000, when none is entered, and
insured-burst value can never be greater than max-burst value)
*cli>
```

The error message indicates that we must either reduce the insured burst value or increase the max-burst value. We choose to reduce the insured burst value to the default maximum burst value of 32000:

```
*cli> define tprof 16 max-rate 64000 insured-rate 32000 insured-burst 32000
principal-service-type guaranteed
*cli>
```

The following example illustrates use of the **show tprof** command to verify that traffic profiles 7 and 16 have been created:

```
*cli> show tprof

Traffic
Profile
ID      Service-Type    Mx R      Mx B      In R      In B  S Scl  Xmt Pri
-----   ------------    ----      ----      ----      ----  -----  -------
1       Insured        77000     32000     66000        0    1      0
2       Insured       122000     32000         0        0    1      0
7       Insured        77000     32000         0        0    1      0
16   Guaranteed        64000     32000     32000    30000    1      0
*cli>
```

We may also use the **show tprof** command to verify that the traffic profile parameters have been set as we intended. The display in the example shows non-default values for 4 parameters: insured rate for traffic profiles 1 and 16, and insured burst and principal service type for traffic profile 16. It shows the configured maximum rate, whose default value is calculated by software for unicast connections.

Define Multicast Groups

Use the **define** command to create or modify a multicast group.

Note A multicast group is a network-wide construct, that is, each multicast group must be unique across the network. To minimize confusion in network management, if a multicast group is defined on one node, it should be defined consistently on all nodes in the network, whether (immediately) used there or not.

---

define mcast

Use the **define mcast** *ID* [*node*:]*c.p*[[*node*:]*c.p* …] command to define a multicast group. A multicast group is a list of LAN ports on nodes in the network. Traffic that matches an associated filter condition is sent to each member of the group. Only one multicast group can be associated with any given filter on a given port, and the action of the filter must be `forward`.

The arguments are as follows:

- *ID*
  The identification number by which the filter is identified, in the range 1 – 255. If *ID* is already in use for a multicast group on the current node, the command adds the specified ports to the old multicast group definition. (A repeated port specification is ignored.) Use the **show mcast** command to display currently defined multicast groups.
- `[`*node*`:]`*c*`.`*p*`[[`*node*`:]`*c*`.`*p*`…]`
  A list of one or more LAN ports. Different LAN media (for example, Ethernet and FDDI) can be combined in one list. The value of *node* is the node name (alias) or chassis ID of a node, and *c.p* is port *p* on card *c* on that node. If *node* is not specified, the port is on the current node. Up to 48

members of a given multicast group can be defined with each use of the **define** command. The **define** command can be used more than once with the same *ID*, to add members to an existing multicast group.

Note To configure a large multicast group, you must understand its topology. Currently, the sum of endpoint ports plus trunks traversed must be less than 100. If you exceed this limit, a message tells you that the connection cannot be established because of failure to get a path.

It is possible to include non-LAN ports in a multicast group, but of course multicast traffic cannot be delivered to such ports.

Because of the length and number of command arguments, this command may exceed the 80-column width assumed for the screen. See the *LightStream 2020 Network Operations Guide* for a description of how the CLI displays long command lines.

Related Commands

You can use the following commands together with the **define mcast** command to manage multicast groups:

- Use the **set port** *c.p* {**bflt** | **ipfilt** | **ipxflt**} commands to associate a multicast group and its traffic profile with a filter in the course of assigning the filter to a port. The filter action must be `forward`.
- Before you can modify (redefine) or delete a multicast group, you must use a **set port** command to break its association with a filter on every port for which that association has been made. There are two ways to do this. To delete the filter, use the following command:

  set port *c.p* {**bflt**|**ipfilt**|**ipxflt**} *ID* **delete**

  To retain the filter on that port but delete the multicast group, you need not delete the filter first. Just repeat the same **set port** *c.p filter-type* command that made the association of the filter to the port, omitting the multicast group arguments, as follows:

  set port *c.p* {**bflt**|**ipfilt**|**ipxflt**} *ID action priority*

- Use the **show mcast** command to display multicast groups.
- Use the **delete mcast** command to delete a multicast group. Use the **delete mcast-member** command to delete member ports from a multicast group.
- If the specified *ID* has already been defined, the effect of the **define mcast** command is to add the specified list of ports to the already defined multicast group.

Delete Network-Wide Constructs

Use the **delete** command to delete a filter condition, traffic profile, or multicast group that was previously created with a **define** command. If the filter, multicast group, or traffic profile has been assigned to a port with a **set port** *c.p* {**bflt** | **ipflt** | **ipxflt**} *ID* command, it cannot be deleted until the association is broken with the **set port** *c.p* {**bflt** | **ipflt** | **ipxflt**} *ID* **delete** command.

delete bflt

Use the **delete bflt** *ID* command to delete a bridge filter. *ID* is the identifier of a bridge filter that was previously created with the **define** command.

delete ipflt

Use the **delete ipflt** *ID* command to delete an IP filter. *ID* is the identifier of an IP filter that was previously created with the **define** command.

delete ipxflt

Use the **delete ipxflt** *ID* command to delete an IPX filter. *ID* is the identifier of an IPX filter that was previously created with the **define** command.

delete mcast

Use the **delete mcast** *ID* command to delete a multicast group. *ID* is the identifier of a multicast group that was previously created with the **define** command.

delete tprof

Use the **delete tprof** *ID* command to delete a traffic profile. *ID* is the identifier of a traffic profile that was previously created with the **define** command.

delete mcast-member

Use the **delete mcast-member** *ID* [*node*:]*c*.*p*[[*node*:]*c*.*p* …] command to delete a member or a list of members from a multicast group without deleting the group itself. *ID* is the identifier of a multicast group previously created with the **define** command. The port list is specified as [*node*:]*c*.*p*[[*node*:]*c*.*p* …]. If *node* is specified, the port is on a node in the network, identified by its node name (alias) *node* or its chassis ID *node*. Each port ***c*.*p*** is port *p* on card *c* on that node. If *node***:** is not specified, the port is on the current node.

Related Commands

You can use the following commands together with the **delete** command:

- Use the **define** command to define a traffic filter, a traffic profile, or a multicast group.
- Use one of the **show** {**bflt** | **ipfilt** | **ipxflt**} [*ID*] commands to display all currently defined filters of a given type.
- Use one of the **show port** *c.p* {**bflt** | **ipfilt** | **ipxflt**} [*ID*] commands to display currently defined filters on a per-port basis.

- Use one of the **set port** *c.p* {**bflt** | **ipfilt** | **ipxflt**}*ID priority action* commands to assign a filter to a port, to specify the sequence in which filter conditions are considered for the specified port, and to specify the action to take if a filter is matched.
- Use one of the **set port** *c.p* {**bflt** | **ipfilt** | **ipxflt**} *ID* **delete** commands to break the association of a filter with a port. You must perform his step before deleting the filter itself with the **delete** command.

Examples

The following examples illustrate the effect of deleting a traffic profile. The first display includes traffic profile number 1:

```
cli> show tprof

Traffic
Profile
ID      Service-Type     Max R     Max B      Ins R    Ins B  S Scl  Xmt Pri
-----   ------------     -----     -----      -----    -----  -----  -------
1          Insured      77000     32000      66000        0     1      0
2          Insured     122000     32000          0        0     1      0
3          Insured    Default     32000          0        0     1      0
7          Insured      77000     32000          0        0     1      0
16      Guaranteed      64000     32000      32000    30000     1      0
cli>
```

The following example illustrates use of the **delete tprof** command to delete traffic profile number 1:

```
cli> delete tprof 1
cli>
```

In the following example, traffic profile number 1 is not included in the output of the **show tprof** command:

```
cli> show tprof

 Traffic
 Profile
ID      Service-Type     Max R     Max B      Ins R    Ins B  S Scl  Xmt Pri
-----   ------------     -----     -----      -----    -----  -----  -------
2          Insured     122000     32000          0        0     1      0
3          Insured    Default     32000          0        0     1      0
7          Insured      77000     32000          0        0     1      0
16      Guaranteed      64000     32000      32000    30000     1      0
cli>
```