endace

accelerated

dagixp_filter_loader User Guide

www.endace.com

Published by:

Endace Measurement Systems® Ltd

Building 7
17 Lambie Drive

PO Box 76802
Manukau City 1702
New Zealand

Phone: +64 9 262 7260
Fax: +64 9 262 7261

support@endace.com
www.endace.com

International Locations

New Zealand Endace Technology® Ltd	**Americas** Endace USA® Ltd	**Europe, Middle East & Africa** Endace Europe® Ltd
Level 9 85 Alexandra Street PO Box 19246 Hamilton 2001 New Zealand	Suite 220 11495 Sunset Hill Road Reston Virginia 20190 United States of America	Sheraton House Castle Park Cambridge CB3 0AX United Kingdom
Phone: +64 7 839 0540 Fax: +64 7 839 0543	Phone: ++1 703 382 0155 Fax: ++1 703 382 0155	Phone: ++44 1223 370 176 Fax: ++44 1223 370 040
support@endace.com www.endace.com	support@endace.com www.endace.com	support@endace.com www.endace.com

Table of Contents

Chapter 1: Purpose **2**

1.1 Revision History 2

Chapter2: Introduction to IXP Filtering **3**

2.1 IXP Filtering Overview 3

2.2 DAG7.1S Card Overview 3

2.3 IXP Filter Software Overview 4

2.4 Coloured Packet Records 5

2.5 IXP Filter Rulesets 5

2.5.1 Rulesets 5

2.5.2 Internet Protocol Version 4 Rules 5

2.5.3 Internet Protocol Version 6 Rules 6

2.5.4 TCP, UDP and SCTP Port Filtering....6

2.5.5 ICMP Type Filtering 7

2.5.6 PoS Frame Header Formats....7

2.5.7 Multi-Protocol Label Switching (MPLS) Support 7

2.6 Modes of Operation 7

2.6.1 Loopback Mode....7

Chapter 3: Loading Configurations **8**

3.1 Card Initialisation 8

3.1.1 Load the Latest PCI-Express FPGA Image....8

3.1.2 Load the Latest Packet Processing FPGA Images 8

3.1.3 Load the Latest IP Filtering Images 8

3.1.4 Configure the Card....8

3.2 `dagixp_filter_loader` Overview 9

3.3 `dagixp_filter_loader` Usage 9

Chapter 4: File Format Specification **10**

4.1 XML Element Specification 10

4.2 Example Filter Files 19

4.2.1 IPv4 Rule....19

4.2.2 IPv4 Rule with Port Filtering 19

4.2.3 Multiple IPv6 Rules 20

Appendix A ERF Record Format **21**

A.1 Type 10 Colored PoS HDLC Record 21

A.2 Type 17 Colored Multi-Channel PoS HDLC Record 22

Appendix B; Statistics **23**

B.1 Standard Output Statistics 23

B.2 Verbose Output Statistics 23

Chapter 1: Purpose

This document describes the usage of the `dagixp_filter_loader` application.

1.1 Revision History

Revision	**Data of Change**	**Description of Change**	**Revision Originator**
1.0	27/03/06	Initial revision	Ben Gray

Chapter2:
Introduction to IXP Filtering

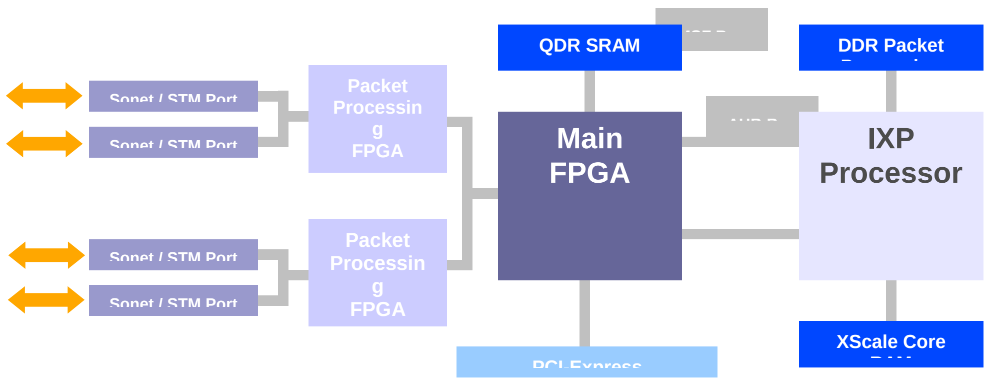
2.1 IXP Filtering Overview

The IXP Filter is a collection of host and embedded software that performs Internet Protocol (IPv4/IPv6) and layer 3 filtering on PoS packets. The IXP Filtering is specific to DAG7.1S cards, all the filtering is done internally on the card. A host API library is supplied to configure and initiate the filtering, however no host process is required to maintain the filtering once configured and running.

2.2 DAG7.1S Card Overview

The Endace DAG7.1S is a four port sonet/STM capture card that can support data rates of 4 × oc3 or 2 × oc12. To provide the extra functionality supported by the card (in this case IP filtering) a Intel IXP network processor has been added. The diagram in figure 2-1 illustrates the main components of a DAG7.1S card.

Figure 2-1. DAG7.1S Card Components

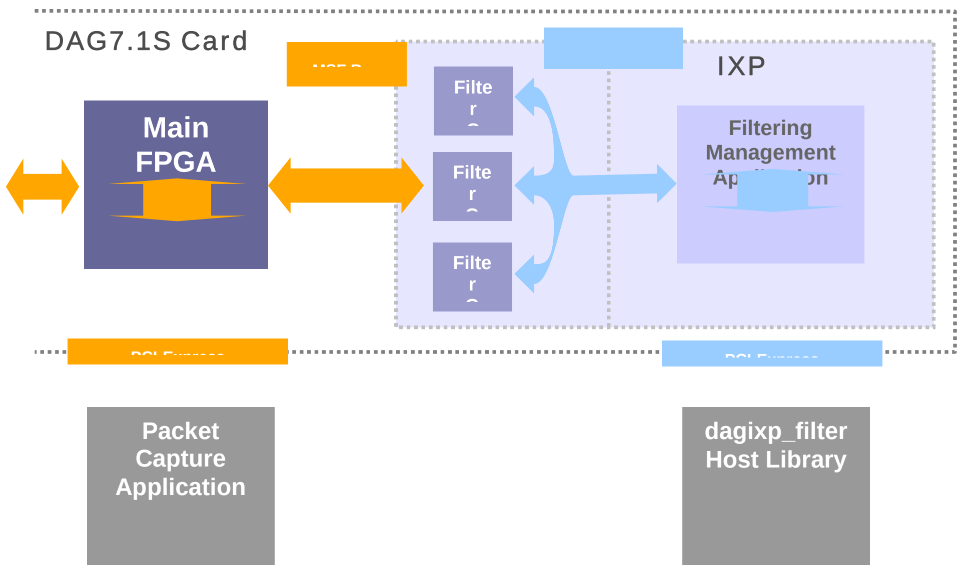


By default when a DAG7.1S card is reset (or powered up) the main FPGA is configured to route packets from the line to the host and from the host to the line, this is standard DAG card behaviour. In this way the DAG7.1S can be used as a standard Endace network capture card without IP filtering. Packet routing within the card is configurable, allowing for packets to be routed from anyone of the three sources (line, host or IXP) to any other including back to themselves. For IP filtering to work, it is required that packets be routed to the IXP Processor (from either the line or the host), the `dagixp_filter_loader` program configures the correct packet routing inside the card automatically.

2.3 IXP Filter Software Overview

The IXP filter software consists of a host library that interfaces with a DAG7.1S card, an embedded software management application that executes on the card and a collection of targeted packet processing programs. The diagram in figure 2-2 illustrates the logical layout of the different software components.

Figure 2-2. IXP Filter Software Diagram


DAG7.1S Card
IXP
Main FPGA
Filte r
Filte r
Filte r
Filtering Management Application
Packet Capture Application
dagixp_filter Host Library

🟧 Packet Paths

🟦 Configuration Message Paths

The *dagixp_filter Host Library* provides an API by which filter rules can be constructed and compiled into user defined ruleset. The host library also provides the functionality to load ruleset into the card and query the current filtering status. The `dagixp_filter_loader` application uses this library to load the rulesets contained in a configuration file to the card.

The *Filtering Management Application* is an embedded program that runs inside the IXP network processor and manages the Filtering microcode (µCode) programmes. It is also the program that the host library communicates with when loading the filter rulesets and querying the filtering status.

The *Filter Microcode* (µCode) performs the packet filtering, it accepts the packet records from the main FPGA and applies the filtering rules. Packets may be dropped by the microcode or routed back out the line or to the host depending on the rules installed.

All configuration of the IXP filter is done via the *dagixp_filter Host Library*, rom image files are provided that contain the embedded software for both the Filtering Management Application and the Filter MicroCode.

2.4 Coloured Packet Records

As packets pass through the filtering software the packet record is modified to change it's ERF type and to add a color field in the ERF record header (the length of the record might also be altered based on the snap length of the rules). The color field is a 14-bit wide value that will match the user defined tag of the rule that the packet hit, refer to Appendix A for more information.

2.5 IXP Filter Rulesets

The IXP filter can filter on Internet Protocol (IP) version 4 and version 6 packets encapsulated in PoS frames.

2.5.1 Rulesets

The IXP filter accepts a collection of rules packaged up within a logical ruleset, multiple rulesets can be loaded into the card but only one can be active at a time. A rule contains a set of fields that is compared with the packet, if all the fields of the rule match then the rule is said to *hit*, if one of more of the fields don't match the rule is said to *miss*. If the rule hits the *action* and *steering* attributes of the rule determines whether the packet is dropped or routed to the host/line. If none of the rules produce a hit then the packet is dropped.

Rules are priority ordered within a ruleset, rules with the highest priority are compared with the packet first, if a rule hits the rest of the rules are ignored. A rule can target either IPv4 or IPv6 packets, it is not possible to have a rule that targets both.

If no ruleset has been activated the filtering software goes into loopback mode, in this mode packets that are routed to the IXP for filtering are looped back out the IXP and to the host. As the packets pass through the filtering software the ERF record header is modified to the mark the with a

2.5.2 Internet Protocol Version 4 Rules

An IPv4 rule will match only if the layer 2 type of the packet being compared is IPv4, any other sort of layer 2 type (for example IPv6, ARP or IPX) will result in an automatic rule miss. Table 2-1 list IPv4 header fields that a rule can apply a bit masked filter to, multiple fields may be defined in a single rule.

Table 2-1. IPv4 header fields that can be filtered on

IPv4 Header Field	***Description***
IP Source Address	A 32-bit bit-masked filter may be applied to the IP source address , if the filter doesn't match the rule misses.
IP Destination Address	A 32-bit bit-masked filter may be applied to the IP destination address , if the filter doesn't match the rule misses.

A rule can also filter on the IP protocol field of a packet however this is not a bit masked type filter, instead a direct value match is performed on the field. If the IP protocol field is set to filter on TCP, UDP or SCTP type packets, up to 254 additional source and destination port filters can be added to the rule, refer to the section 2.5.4 for more information. ICMP packets can also be filtered in a similar way, with up to 254 ICMP type filters, refer to the section 2.5.5 for more information.

IP options are automatically skipped by the filtering process, however filter rules cannot target fields within the IP options.

2.5.3 Internet Protocol Version 6 Rules

IPv6 rules target packets with an IPv6 layer 2 type only, any other packet types automatically result in a rule miss. Table 2-2 list IPv6 header fields that a rule can apply a bit masked filter to, multiple fields may be defined in a single rule.

Table 2-2. IPv6 header fields that can be filtered on

IPv6 Header Field	*Description*
IP Source Address	A 128-bit bit-masked filter may be applied to the IP source address , if the filter doesn't match the rule misses.
IP Destination Address	A 128-bit bit-masked filter may be applied to the IP destination address , if the filter doesn't match the rule misses.
Flow Label	A 20-bit bit-masked filter may be applied to the flow label field of an IPv6 header, if the filter doesn't match the rule misses.

As with an IPv4 rule, layer 3 protocols (TCP, UDP, SCTP or ICMP) can be filtered on, refer to the sections 2.5.4 and 2.5.5 for more information.

IPv6 extension headers are automatically skipped by the filtering process, table 2-3 illustrates which extension headers are supported, if an unknown extension header is found, the packet is dropped and a statistics counter is incremented. Currently filter rules can't target fields within extension headers.

Table 2-3 Supported IPv6 extension headers

Name	*Supported*	*Description*
Hop-by-Hop Options Header	yes	Contains options for hop-by-hop processing.
Routing Header	yes	Contains routing information for the packet.
Fragment Header	yes	Contains information for fragmenting and reassembling the packet.
Authentication Header	yes	Used for packet authentication.
Encapsulated Security Payload Header	no	Provides fields for packet encryption and security.
Destination Options Header	yes	Contains optional information for the destination node of the packet.

2.5.4 TCP, UDP and SCTP Port Filtering

As well as IP header filtering, a rule can filter on TCP, UDP or SCTP source and destination port numbers. Two types of port filters can be added to a rule; bit-masked or port range, bit-masked filters take a value and mask pair to compare against the packet, a port range filter takes a maximum and minimum port value to compare with the packet. Up to 254 source and destination port ranges filters can be added to a rule or a single bit-masked filter. If no port filters are added to the rule the port value in the packet is ignored by the rule. If multiple port filters are added to the rule, each is compared with the packet value and OR'ed together to produce the result, for example if a rule has two source port ranges 0-25 and 80-90 as well as two destination port ranges 25-50 and 110-120, then only packets with a source port value of between 0 and 25 or 80 and 90 and with a destination port value of between 25-50 or 110-120 will result in a rule hit.

The maximum and minimum values of a port range are inclusive values. If wanting to filter on a single port value, a range can be added to the rule with both the minimum and maximum values set to the same value.

2.5.5 ICMP Type Filtering

ICMP packets can be filtered on their type fields in much the same way as ports can be filtered on for TCP, UDP and SCTP packets. The only differences between port filter and ICMP type filtering is that the type values are only 8-bit rather than the 16-bit for port values and there is a single ICMP type filter list rather than the two source and destination port filter lists.

2.5.6 PoS Frame Header Formats

Both PPP with HDLC (RFC 1662) and Cisco type PoS encapsulation is supported, 16 or 32 bit CRCs are also supported. Table 2-4 lists the PoS header formats supported by the filtering software.

Table 2-4 Supported PoS header formats

PoS Header Values	***Format***	***Layer 2 Type***
0xFF030021	RFC 1662	IPv4
0x0F000800	Cisco	IPv4
0xFF030057	RFC 1662	IPv6
0x0F0086DD	Cisco	IPv6
0xFF030281	RFC 1662	IP with Unicast MPLS shims
0xFF030283	RFC 1662	IP with Multicast MPLS shims
0x0F008847	Cisco	IP with Unicast MPLS shims
0x0F008848	Cisco	IIP with Multicast MPLS shims

2.5.7 Multi-Protocol Label Switching (MPLS) Support

The filtering microcode can handle packets with up to 10 MPLS shim headers inserted between the HDLC and IP headers. If more than 10 shims are present, the packet is automatically dropped and a statistic counter updated. Currently filter rules cannot target fields within MPLS shims.

2.6 Modes of Operation

The filtering software has two possible modes of operation, the first is filter mode, where packets that are presented to the IXP network processor are filtered based on the activated ruleset, this is what is described above.

2.6.1 Loopback Mode

The second mode of operation is loopback mode, where incoming packets are looped back out of the IXP without being filtered. In loopback mode the ERF type of the packet record is still modified to have a color field, however the value in the color field is always 0x3FFF, this is used to distinguish loopback packets from packets that have been filtered.

Loopback mode is not directly controllable by the user, instead it is automatically entered when the filtering software

- is initially started and no rulesets have been activated.
- is activating a new ruleset (the process of deactivating the old ruleset and activating the new ruleset is not instantaneous, during this time the software is put in looopback mode so packets are not lost).

Chapter 3:
Loading Configurations

3.1 Card Initialisation

The DAG card will likely require configuration prior to loading a ruleset into the card. The following paragraphs illustrate the typically steps required for configuration, refer to the card manual for more detailed information.

3.1.1 Load the Latest PCI-Express FPGA Image

Run the `dagrom` program to load the latest image into the card.

```
dagrom -d dag0 -rvp -f filename.bit
```

Where `filename.bit` is the name of the latest FPGA image file to load. Replace `dag0` on the command line with the number of the DAG card if multiple cards are installed on your system.

3.1.2 Load the Latest Packet Processing FPGA Images

The `dagld` program is used to load the packet processing images into the card.

```
dagld -d dag0 -x pp_filename.bit:pp_filename.bit
```

Where `pp_filename.bit` is the name of the latest packet processing FPGA image file to load. Replace `dag0` on the command line with the number of the DAG card if multiple cards are installed on your system.

3.1.3 Load the Latest IP Filtering Images

An additional three images may need to be loaded into the card for performing the IP filtering. Run the `dagrom` program to load these additional images.

```
dagrom -d dag0 -rylv -cb -f d71s_bootloader.rom
dagrom -d dag0 -rylv -ck -f d71s_kernel.rom
dagrom -d dag0 -rylv -cf -f d71s_ipf_filesystem.rom
```

Where `d71s_bootloader.rom`, `d71s_kernel.rom` and `d71s_ipf_filesystem.rom` are the names of the three image files. Replace `dag0` on the command line with the number of the DAG card if multiple cards are installed on your system.

Note

The IP Filtering Images are stored in non-volatile memory on the card, therefore once they are loaded the first time there is no need to reload them unless they have been overwritten by other images.

3.1.4 Configure the Card

Refer to the DAG card user manual for details on how to configure the card for your particular network settings. Care should be taken when setting the snap length during card configuration, because packets are trimmed to the snap length prior to being present to the IP filtering software.

3.2 `dagixp_filter_loader` Overview

The command line `dagixp_filter_loader` application is a tool that accepts a ruleset file and loads the details into the DAG card. It utilises the DAG IXP Filtering software API library to configure the card.

3.3 `dagixp_filter_loader` Usage

The following arguments can be used with `dagixp_filter_loader`.

```
dagixp_filter_loader - loads a ruleset into a DAG7.1S card
Usage : dagixp_filter_loader -d <device> [options] -f <ruleset file>
Options:
    -?,--usage
    -h,--help                   display help (this page)
    -v,--verbose                increase verbosity
    -d,--device <device>        DAG device to use
    -f,--rule_file <filename>   file containing the ruleset to load
    -y,--do_not_download        don't download the ruleset to the card, if
                                not specified the ruleset is always
                                downloaded to the card
    -s,--stats <seconds>        display filtering statistics every
                                <seconds>
    -u,--do_not_clear_stats     don't clear the statistics, if not
                                specified all statistics are cleared when
                                a ruleset is loaded.
    -x,--restart                restart the IXP processor on the card,
                                ignored if -y is specified
    -m,--cpp_memtest            run CPP memory tests on the card, ignored
                                if -x is not also specified
    -n,--xsi_memtest            run XSI memory tests on the card, ignored
                                if -x is not also specified
    -V,--version                display version information.
```

The use of a file containing the ruleset details is mandatory, the format of the file is given in chapter 4 of this document.

The `-y` *don't download* option is useful if you want to simply parse a configuration file and verify the syntax is correct.

Statistics can be displayed periodically using the `-s` option, the statistics consist of the packet and error counters maintained by the IXP Filtering software. Additional statistics will be displayed if the verbose option is also specified. Refer to Appendix B for a description of the statistics.

Chapter 4:
File Format Specification

The filter loader file is written in xml format, it must satisfy the format by having a single root node `<ruleset>`. Detailed in the following sections are the possible elements that may be used inside the root element.

4.1 XML Element Specification

4.1.1 <ruleset>

This must be the root element of the xml document. This element allows only one possible child element, the `<rule>` element.

Table 4-1. <ruleset> Attributes

Attribute	***Use***	***Type***	***Default Value***	***Description***
version	Mandatory	float	-	Defines the version of the file, currently the only version is 1.0 and this is the only valid value. This document describes version 1.0 only.

Table 4-2. <ruleset> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		
rule	0	Unlimited	Rule element that contains the details of a single rule inside the ruleset. The order of the `<rule>` elements doesn't effect the order the rules are processed during filtering.	Section 4.1.2

4.1.2 <rule>

This element defines all the details about a rule within a ruleset. The only two possible child elements are `<ipv4>` and `<ipv6>`, these two elements are mutually exclusive, it is not possible to define a rule that has both `<ipv4>` and `<ipv6>` elements.

The `<rule>` element is required to have either an `<ipv4>` or `<ipv6>` child tag, if the element is empty the rule is ignored.

Table 4-3. <rule> Attributes

Attribute	***Use***	***Type***	***Default Value***	***Description***
action	Optional	enumeration	“accept”	Defines the action to perform if the rule produces a hit. Possible values are `accept` or `reject`. If `reject` is specified the packet will be dropped if the rule hits, if `accept` is specified the

				packet will be routed to either the host or back out the line depending on the value of the `steering` attribute.
tag	Optional	integer	0	User defined value that is inserted into the color field of the packet record upon a filter hit. The color field is 14-bits wide, therefore the tag value should be a value in the range of 0-16383. Refer to Appendix A for information on the format of packet records including the color field.
priority	Optional	integer	0	Defines the priority of the rule, the lower the number, the higher the priority of the rule. Rules with higher priority are compared with the packet first. Multiple rules can have the same priority, however the order in which they are compared with a packet are indeterminable. ***Note*** To increase the performance of the filtering process, set the priority of the rules that are likely to hit the most often to the lowest possible value.
snap	Optional	integer	65528	Defines the snap length applied to the packet if the rule hits and the `action` is set to `accept`. The snap length only applies if the `snap` value is less than the size of the packet record. The snap length should be a multiple of 8 (this ensures 64-bit alignment of packet records), if not a multiple of 8 the snap length will be rounded down to the nearest multiple. ***Note*** The snap length defined for a rule is independent of the snap length specified when configuring the DAG card for packet reception.
steering	Optional	enumeration	“host”	Defines how a packet that matches the rule is steered, either to the host or back out the line. The two possible values are `line` or `host`. By default packets that match a rule are routed to the host.

Table 4-4. <rule> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		
ipv4	0	1	Defines IPv4 fields to filter on.	Section 4.1.3
ipv6	0	1	Defines IPv6 fields to filter on.	Section 4.1.3

4.1.3 <ipv4> & <ipv6>

Defines the parent rule element as being of type IPv4 or IPv6, packets that don't match either type are automatically discarded. For example if a rule is defined as an IPv6 type, any IPv4 packets that arrive will always generate a rule miss and vise-versa for IPv4 rules.

The `<ip-source>` and `<ip-dest>` child elements have different formats depending on the whether the parent element is `<ipv4>` or `<ipv6>`.

Table 4-5. <ipv4> & <ipv6> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		
ip-source	0	1	Defines the IPv4 or IPv6 source address to filter on.	Section 4.1.4
ip-dest	0	1	Defines the IPv4 or IPv6 destination address to filter on.	Section 4.1.4
ipv6-flow	0	1	This element is only valid if the parent item is `<ipv6>`. The `<ipv6-flow>` element defines a value and mask pair to use for filtering on the flow label field of an IPv6 header.	Section 4.1.5
udp	0	1	Defines the layer 3 protocol for the rule as being UDP, this rule will miss for all non-UDP packets. This element allows for addition layer 3 filter fields to be defined.	Section 4.1.6
tcp	0	1	Defines the layer 3 protocol for the rule as being TCP, this rule will miss for all non- TCP packets. This element allows for addition layer 3 filter fields to be defined.	Section 4.1.6
sctp	0	1	Defines the layer 3 protocol for the rule as being SCTP, this rule will miss for all non-SCTP packets. This element allows for addition layer 3 filter fields to be defined.	Section 4.1.6
icmp	0	1	Defines the layer 3 protocol for the rule as being ICMP, this rule will miss for all non-ICMP packets. This element allows for addition layer 3 filter fields to be defined.	Section 4.1.7

4.1.4 <ip-source> & <ip-dest>

These elements define the IPv4 or IPv6 source/destination addresses to filter on. This element typically has `<addr>` and `<mask>` child elements, the `<mask>` element defines a value that is AND'ed with the packet to produce a masked value that is compared with the contents of the `<addr>` element. If the `<addr>` child element is not present the value defaults to all zero bits, if the `<mask>` child element is not present the mask defaults to all ones bits. The format of the `<addr>` and `<mask>` child elements is different for IPv4 and IPv6 rules, refer to table 4-6 for information on the two formats.

Table 4-6. <ip-source> & <ip-dest> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		

addr	0	1	Defines the IPv4/IPv6 address to filter on, the format of the contents of this element is dependant on the type of the rule (either IPv4 or IPv6).
			
			IPv4 Format
			For IPv4 rules the contents should be formatted like `X.X.X.X` where `X` is an 8-bit number represented by either a decimal (0-255) or hexadecimal (0-FF) value . The `hex` attribute determines the format of the address, by default decimal is used. Add `hex="true"` to the element to enable hexadecimal parsing.
			
			IPv4 Examples:
			`<ip-source>`
			`<addr>192.168.0.0</addr>`
			`</ip-source>`
			
			`<ip-dest>`
			`<addr hex="true">C0.A8.0.0</addr>`
			`<mask hex="true">FF.FF.0.0</mask>`
			`</ip-dest>`
			
			IPv6 Format
			For IPv6 rules the contents should be formatted like x:x:x:x:x:x:x:x where x is a 16-bit number represented by either a decimal (0-65535) or hexadecimal (0-FFFF) value . As with the IPv4 style, the `hex` attribute can be used to change between decimal format (the default) and hexadecimal. It is important to note that if `hex="true"` is not specified as an attribute the format of the address is in decimal, which is not the standard way IPv6 addresses are written.
			
			IPv6 Note: The symbol "::" is a special syntax that can be used as a shorthand way of representing multiple 16-bit groups of contiguous 0's (zeros). The "::" can appear anywhere in the address; however it can only appear once.
			
			IPv6 Examples:
			`<ip-source>`
			`<addr>61024:0:0:0:0:0:102:12</addr>`
			`<mask>65520:0:0:0:0:0:255:15</mask>`
			`</ip-source>`
			
			`<ip-dest>`
			`<addr hex="true">EE60::66:C</addr>`
			`<mask hex="true">FFF0::FF:F</mask>`
			`</ip-dest>`
				-

mask	0	1	Defines the mask to use for the IPv4 or IPv6 address, the contents have the same format as the `<addr>` element.	-

4.1.5 <ipv6-flow>

Defines a bit-masked filter to use in a rule to filter out IPv6 packets that don't have a matching flow label field. This element typically has `<flow>` and `<mask>` child elements, the `<mask>` element defines a value that is AND'ed with the packet to produce a bit-masked value that is compared with the contents of the `<flow>` element. If the `<flow>` child element is not present the value defaults to all zero bits, if the `<mask>` child element is not present the mask defaults to all ones bits.

Table 4-7. <ipv6-flow> Elements

Element	*Occurrences*		*Description*	*Section*
	Min	*Max*		

flow	0	1	Defines the value to compare with the packets flow label after the mask has been applied. The contents of this field should be a 20-bit number represented as either a decimal (0-1048575) or hexadecimal (0-FFFFF) value. The `hex` attribute determines the format of the value, by default decimal is used. Add `hex=”true”` to the element to enable hexadecimal parsing.
			
			Examples:
			`<ipv6-flow>`
			`<flow>1034</addr>`
			`<mask>2047</mask>`
			`</ipv6-flow>`
			
			`<ipv6-flow>`
			`<flow hex=”true”>40A</flow>`
			`<mask hex=”true”>7FF</mask>`
			`</ipv6-flow>`
				-

mask	0	1	Defines the mask to use for the flow label, the contents have the same format as the `<flow>` element.	-

4.1.6 <udp>, <tcp> & <sctp>

These elements define the layer 3 protocol used by the rule to filter the packets. By defining any one of these elements, a rule can also define a list of source and destination port filters.

Packets that are compared with the rule and don't match the layer 3 type represent by the `<udp>`, `<tcp>` or `<sctp>` elements, automatically generate a rule miss.

Table 4-8. <udp>, <tcp> & <sctp> Elements

Element	*Occurrences*		*Description*	*Section*
	Min	*Max*		
source-port-list	0	Unlimited	Contains a list of source port values/ranges to compare with a packet. If this element is not specified the source port field of the packets TCP, UDP or SCTP header is ignored by the rule,	Section 4.1.8
dest-port-list	0	Unlimited	Contains a list of destination port values/ranges to compare with a packet. If this element is not specified the destination port field of the packets TCP, UDP or SCTP header is ignored by the rule,	Section 4.1.8

4.1.7 <icmp>

This elements defines the layer 3 protocol used by the rule to filter the packets. Packets that are compared with the rule and don't contain an ICMP payload, automatically produce a filter miss.

Table 4-9. <icmp> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		
icmp-type-list	0	Unlimited	Contains a list of ICMP type values/ranges to compare with a packet. If this element is not specified the ICMP type field of the packets ICMP header is ignored by the rule,	Section 4.1.9

4.1.8 <source-port-list> & <dest-port-list>

This element is a container for one or more `<port>`, `<bitmask>` or `<range>` elements, each element is a member of a port filter list. If any member of the list matches the source/destination port of the TCP, UDP or SCTP field of a packet the rule hits. If no elements are defined inside the list then the rule will ignore the port value when determining if the rule hits or misses.

Table 4-10. <source-port-list> & <dest-port-list> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		

port	0	Unlimited	Defines a single port value entry in the list. The contents of this field should be a 16-bit number represented by either a decimal (0-65535) or hexadecimal (0-FFFF) value. The `hex` attribute determines the format of the value, by default decimal is used. Add `hex=”true”` to the element to enable hexadecimal parsing.
			
			Example:
			`<source-port-list>`
			`<port>80</port>`
			`<port hex=”true”>50</port>`
			`</source-port-list>`
				-

bitmask	0	Unlimited	Defines a bitmasked port value entry in the list.	Section 4.1.10
range	0	Unlimited	Defines a range of port values in the list.	Section 4.1.11

IMPORTANT

Due to the design of the IP filter software, mixing bitmask port filters with a single port value or a range of port values inside a list, leads to a decline in filter performance. Internally the filtering software can compare up to 254 port values or a single bit masked port value per filter rule, therefore if multiple bit masked values are defined in a list, duplicated rules are generated with one bit masked port value per rule. This is worsen if source and destination port list are used, because port filters are OR'ed together, leading to a situation where multiple duplicated rules have to be generated to cover the different combinations of source and destination port values, the following example illustrates the situation. The rule described below will be converted to four rules by the IXP Filtering API when the ruleset is loaded into a card, table 4-11 shows the port values generated by the rule.

```
<rule>
<ipv4>
<tcp>
        <source-port-list>
                <port>80</port>
                <range>
                        <min-port>0</min-port>
                        <max-port>10</max-port>
                </range>
                <bitmask>
                        <port>25</port>
                        <mask hex=”true”>FF</mask>
                </bitmask>
        </source-port-list>

        <dest-port-list>
                <port>90</port>
                <range>
                        <min-port>10</min-port>
                        <max-port>30</max-port>
                </range>
                <bitmask>
                        <port hex=”true”>60</port>
                        <mask hex=”true”>F0</mask>
                </bitmask>
        </dest-port-list>
</tcp>
</ipv4>
</rule>
```

Table 4-11. Rules Downloaded to the Card

Rule	*Source Port Filters*	*Destination Port Rules*
Rule 1	Value [80], Range [0-10]	Value [90], Range [10-30]
Rule 2	Value [80], Range [0-10]	Bitmasked [0x60,0xF0]
Rule 3	Bitmasked [25,0xFF]	Value [90], Range [10-30]
Rule 4	Bitmasked [25,0xFF]	Bitmasked [0x60,0xF0]

4.1.9 <icmp-type-list>

This element is a container for one or more `<type>`, `<bitmask>` or `<range>` elements, each element is a member of a list. If any member of the list matches the type field inside the ICMP header of a packet then the rule hits. If no elements are defined inside the list then the rule will ignore the ICMP type values when determining if the packet hits or misses.

As with the `<source-port-list>` and `<dest-port-list>` elements, care should be taken when adding `<bitmask>` list elements, refer to the comments in section 4.1.8 for more information.

Table 4-12. <icmp-type-list> Elements

Element	*Occurrences*		*Description*	*Section*
	Min	*Max*		

type	0	Unlimited	Defines a single ICMP type value entry in the list. The contents of this field should be an 8-bit number represented by either a decimal (0-255) or hexadecimal (0-FF) value. The `hex` attribute determines the format of the value, by default decimal is used. Add `hex="true"` to the element to enable hexadecimal parsing.
			
			Example:
			`<icmp-type-list>`
			`<type>80</type>`
			`<type hex="true">50</type>`
			`</icmp-type-list>`
				-

bitmask	0	Unlimited	Defines a bit masked port value entry in the list.	Section 4.1.10
range	0	Unlimited	Defines a range of port values in the list.	Section 4.1.11

4.1.10 <bitmask>

This element is a child element of either a `<source-port-list>`, `<dest-port-list>` or `<icmp-type-list>` element. It defines a bit masked UDP, TCP or SCTP port value or a bit masked ICMP type value depending on the lineage of the element.

The bit masked element should be used with caution, because a list of ports or ICMP types with multiple different bit masked elements, will typically lead to severely degraded filter performance, see the comments in section 4.1.8 for more information.

Table 4-13. <bitmask> Elements

Element	*Occurrences*		*Description*	*Section*
	Min	*Max*		

value	0	1	Defines the value to compare with the packet once the mask has been applied. The format of this values depends on the parent of the `<bitmask>` element. If `<source-port-list>` or `<dest-port-list>` is the parent then the value should be a 16-bit number. If `<icmp-type-list>` is the parent element then the value should be a 8-bit number.
			
			An optional `hex` attribute can be added to the element, this determines the format of the value. Add `hex="true"` to the element to enable hexadecimal parsing. If the `hex` attribute is not specified or `hex="false"`, the value is parsed as a decimal number.
			
			Examples:
			`<bitmask>`
			`<value>12</value>`
			`<mask>255</mask>`
			`</bitmask>`
			
			`<bitmask>`
			`<value hex="true">0C</value>`
			`<mask hex="false">255</mask>`
			`</bitmask>`
				-

mask	0	1	The mask to apply to the packet field before comparing the value. This element has the same format as the `<value>` element.	-

4.1.11 <range>

This element defines a range of TCP, UDP or SCTP ports or ICMP types, it can be a child of `<source-port-list>`, `<dest-port-list>` or `<icmp-type-list>` elements.

It is mandatory to define a `<min>` and `<max>` child item, they define the minimum and maximum values of the range respectively.

Table 4-14. <range> Elements

Element	***Occurrences***		***Description***	***Section***
	Min	***Max***		

min	1	1	Defines the minimum value of either a port value or an ICMP type. The format of this values depends on the parent of the `<range>` element. If `<source-port-list>` or `<dest-port-list>` is the parent then the value should be a 16-bit number. If `<icmp-type-list>` is the parent element then the value should be a 8-bit number.
			
			An optional `hex` attribute can be added to the element, this determines the format of the value. Add `hex=”true”` to the element to enable hexadecimal parsing. If the `hex` attribute is not specified or `hex=”false”` is specified, the value is parsed as a decimal number.
			
			Examples:
			`<range>`
			`<min>12</min>`
			`<max>100</max>`
			`</range>`
			
			`<range>`
			`<min hex=”true”>0C</min>`
			`<max hex=”false”>100</max>`
			`</range>`
				-

max	1	1	Defines the maximum value of the range. This element has the same format as the `<min>` element.	-

4.2 Example Filter Files

This section provides example IXP Filtering ruleset files.

4.2.1 IPv4 Rule

The following example defines a single IPv4 rule that accepts all packets with source IP addresses of 192.168.1.0/16 and a destination address of 192.168.64.0/255.

Example file:

```
<?xml version="1.0"?>

<ruleset version="1.0">

<rule action=”accept” tag=”1”>
      <ipv4>
            <ip-source>
                  <addr>192.168.1.0</addr>
                  <mask>255.255.255.240</mask>
            </ip-source>
            <ip-dest>
                  <addr>192.168.64.0</addr>
                  <mask>255.255.255.0</mask>
            </ip-dest>
      </ipv4>
</rule>

</ruleset>
```

4.2.2 IPv4 Rule with Port Filtering

The following example defines a single IPv4 rule that accepts all packets with source IP addresses of 192.168.1.0/16 and has a TCP payload with a source port value between 0 and 25 or 80. In the case of a rule hit the packet is routed to the host and it's length is snapped to 64 bytes.

Example file:

```
<?xml version="1.0"?>

<ruleset version="1.0">

<rule action=”accept” tag=”1” steering=”host” snap=”64”>
      <ipv4>
            <ip-source>
                  <addr>192.168.1.0</addr>
                  <mask>255.255.255.240</mask>
            </ip-source>
            <tcp>
                  <source-port-list>
                        <port>80</port>
                        <range>
                              <min>0</min>
                              <max>25</max>
                        </range>
                  </source-port-list>

            </tcp>
      </ipv4>
</rule>

</ruleset>
```

4.2.3 Multiple IPv6 Rules

In this example every packet is routed to the host except ones that are IPv6 and have either an ICMP payload or source address of `0000:0000:0000:0000:0000:0000:C613:0102`.

Note

In the second rule the `<mask>` element in the `<ip-source>` element is not necessary because if a mask is not specified it defaults to all ones values.

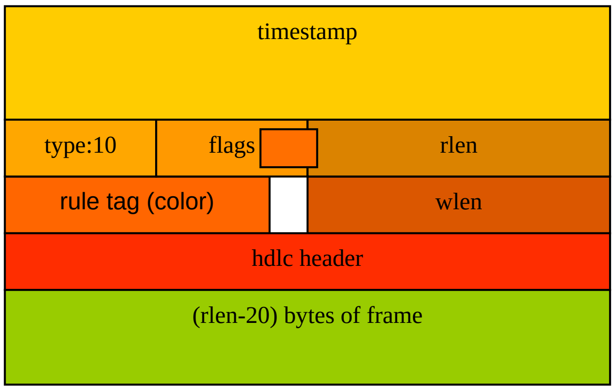
Example file:

```
<?xml version="1.0"?>

<ruleset version="1.0">

<!-- Rule to discard all IPv6 packets with ICMP payloads -->
<rule action=”reject” priority=”0”>
      <ipv6>
            <icmp/>
      </ipv6>
</rule>

<!-- Discard all IPv6 packets with a source address ::C613:0102 -->
<rule action=”reject” priority=”1”>
      <ipv6>
            <ip-source>
                  <addr hex=”true”>::C613:0102</addr>
                  <mask hex=”true”>
                        FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
                  </mask>
            </ip-source>
      </ipv6>
</rule>

<!-- Accept everything else rule for IPv6 -->
<rule action=”accept” priority=”2”>
      <ipv6/>
</rule>

<!-- Accept all IPv4 packets rule -->
<rule action=”accept” priority=”3”>
      <ipv4/>
</rule>

</ruleset>
```

Appendix A
ERF Record Format

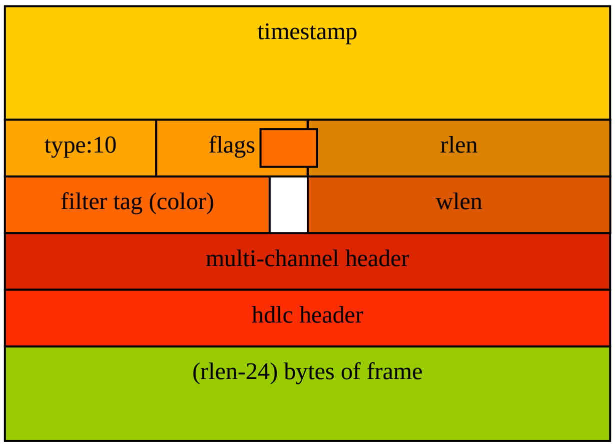
The following ERF record formats are generated by the IXP filtering software, refer to the *EDM11-01 Endace Extensible Record Format* document for more detailed information.

A.1 Type 10 Colored PoS HDLC Record



Data Format	***Size***	***Description***
timestamp	64 bits	The time of arrival of the cell, an ERF 64-bit timestamp, described in more detail in the *EDM11-01* document.
type	8 bits	This field contains an enumeration of the frame subtype. For DSM colored PoS records this value should be 15.
flags	8 bits	This byte is divided into 2 parts, the interface identifier, and a set of 1-bit flags.
rlen	16 bits	Record length. Total length of the record transferred over PCI bus to storage.
rule tag (color)	14 bits	The 14-bit unsigned integer (0-16383) that corresponds to the filter rule (rule tag) this packet matched.
stream	2 bits	The target receive stream for the packet record, this will always be 0.
wlen	16 bits	Wire length. The length of the packet that was received from the line.
hdlc header	32 bits	The 32-bit PoS frame HDLC header that was received from the line.

A.2 Type 17 Colored Multi-Channel PoS HDLC Record



Data Format	***Size***	***Description***
timestamp	64 bits	The time of arrival of the cell, an ERF 64-bit timestamp, described in more detail in the *EDM11-01* document.
type	8 bits	This field contains an enumeration of the frame subtype. For DSM colored PoS records this value should be 15.
flags	8 bits	This byte is divided into 2 parts, the interface identifier, and a set of 1-bit flags.
rlen	16 bits	Record length. Total length of the record transferred over PCI bus to storage.
rule tag (color)	14 bits	The 14-bit unsigned integer (0-16383) that corresponds to the filter rule (rule tag) this packet matched.
stream	2 bits	The target receive stream for the packet record, this will always be 0.
wlen	16 bits	Wire length. The length of the packet that was received from the line.
mulit-channel header	32 bits	This header is divided into several bit fields: 0-9 Connection number (0-511) 10-15 Reserved 16-23 Reserved 24 FCS Error 25 Short Record Error (<5 Bytes) 26 Long Record Error (>2047 Bytes) 27 Aborted Frame Error 28 Octet Error 29 Lost Byte Error 30 1st Rec. This is the first record received since the connection was configured. 31 Reserved
hdlc header	32 bits	The 32-bit PoS frame HDLC header that was received from the line.

Appendix B; Statistics

B.1 Standard Output Statistics

The standard statistics display the number of packets that have been received and the number of packets that have been accepted by the filter rules. The standard statistics are displayed with the `-s` command line option.

Example B-1. Standard statistics output

```
Packets Recv: 16740735200      Packets Accepted: 16738672716      In Play: 0
Packets Recv: 16740735200      Packets Accepted: 16738672716      In Play: 0
Packets Recv: 16740735200      Packets Accepted: 16738672716      In Play: 0
.
.
.
```

Table B-1. Standard statistics

Caption	***Description***
Packets Recv	The number of packets that have been received by the IXP, this is a 64-bit number that will roll over to 0 (after 18446744073709551615 packets have been received).
Packets Accepted	The number of packets that have been accepted and sent back out the IXP chip, this statistic doesn't cover the number of packets that have been accepted but were dropped because of buffer overflows or transmit errors.
In Play	Contains the number of packets that are being filtered at that instant, this statistic will contain a value between 0 and 8192. When the number of packets in play reaches 8192 packets will be dropped, if this statistic is regularly up around 4000 you may want to consider reducing the number of filter rules to avoid packets being dropped due to buffer overrun.

B.2 Verbose Output Statistics

The extended statistics are displayed when the verbose option is issued on the command line. All the extended statistics are drop counters, indicating the number of packets dropped due to their respective errors. Each statistic counts up from 0 to 0xFFFFFFFF, the counters don't roll over to 0 once they reach 0xFFFFFFFF.

Example B-2. Extended statistics output

```
 Invalid ERF type:        0          Invalid/unknown HDLC header: 0
 Invalid ERF size:        0          Buffer limit reached:        0
 RX Fifo Full:            0          TX Fifo Full:                0
 Unknown IPv6 Ext Header: 0          MPLS Stack Overflow:         0
 Sequence Buffer Full:    0          Packet to Large:             0
 MSF Error SOP:           0          MSF Error EOP:               0
 MSF Error BOP:           0          MSF Error NULL Packet:       0
 Invalid MSF Status Word: 0
--------------------------------------------------------------------------------
```

Table B-2. Extended statistics

Caption	***Description***
Invalid ERF type	Number of packets dropped because the ERF type received from the FPGA was not a PoS type record.
Invalid/unknown HDLC header	Number of packets dropped because the HDLC header of the PoS frame was not recognised, the filtering software only accepts a small number of standard HDLC headers, refer to table 2-4.
Invalid ERF size	Number of packets dropped because of a mismatch in the record length field of the ERF header and the length of the actual record.
Buffer limit reached	Number of packets dropped due to the internal packet buffers reaching their limit, the filtering software can buffer up to 8192 packets internally when this limit is reached packets are dropped. This statistic will increase if the host machine can't process the packets fast enough, in such cases reverse flow control will be asserted back to the filtering software and packets will be dropped.
RX Fifo Full	This statistic is reserved for future use and is currently not implemented.
TX Fifo Full	This statistic is reserved for future use and is currently not implemented.
Unknown IPv6 Ext Header	Number of IPv6 packets dropped because an unknown extension header was found in the packet, refer to table 2-3 for a list of supported IPv6 extension headers.
MPLS Stack Overflow	Number of IPv6 packets dropped because more than 10 MPLS shim headers were found in the packet, the IP filtering software supports a maximum of 10 MPLS shims only.
Sequence Buffer Full	Indicates the number of packets dropped because the filtering process delayed the packet to long, this is caused by complex filter rules that lead to varied packet latencies within the filtering software.
Packet To Large	Number of packets dropped because they were too large to be processed, the maximum packet size supported is 4096 bytes including the ERF header.
MSF Error SOP	Number of packets dropped because of an MSF bus error.
MSF Error EOP	Number of packets dropped because of an MSF bus error.
MSF Error BOP	Number of packets dropped because of an MSF bus error.
MSF Error NULL Packet	Number of packets dropped because of an MSF bus error.
Invalid MSF Status Word	Number of packets dropped because of an MSF bus error.