ERF Types Reference Guide

EDM11-01 - Version 21

Website

www.endace.com

© Endace Technology Limited 2005 - 2016

Disclaimer

Contents

Introduction

Overview

This document identifies and explains the following:

- The Endace Extensible Record Format (ERF) (page 3).
- Extension Headers (EH) (page 17).
- Output Formats (page 27).

Refer to previous versions of this document for details of ERF and EH types associated with DAG cards which are no longer supported.

This document is applicable to DAG software release 5.6 or greater. See previous versions of this document for information on obsolete ERF types.

Extensible Record Format

Introduction

DAG cards produce trace files in their own native format, known as the Extensible Record Format (ERF).

An ERF file contains of a series of ERF records; there is no special file header. Each ERF record describes one packet. This allows concatenation and splitting to be performed arbitrarily on ERF record boundaries.

DAG Card Extensible Record Format Types

Current DAG cards and DAG software (dagconvert) support the following ERF Types:

Number	Type	Description
1:	TYPE_HDLC_POS	Packet over SONET / SDH frames, using either PPP or CISCO HDLC framing.
2:	TYPE_ETH	Ethernet
3:	TYPE_ATM	ATM cell
16:	TYPE_DSM_COLOR_ ETH	Ethernet format like TYPE_ETH, but with the LCNTR field reassigned as DSM COLOR
22:	TYPE_IPV4	IPV4 Variable Length Record
23:	TYPE_IPV6	IPV6 Variable Length Record
24:	TYPE_RAW_LINK	Raw link data, typically SONET or SDH Frame
27:	TYPE_META	Adds Provenance records
48:	TYPE_PAD	Pad Record type

ERF Types for Each DAG Card

The ERF types used by each DAG card are listed below.

DAG Card	ERF Type
DAG 7.4S	Type 1 - PoS HDLC Record Type 3 - ATM Cell Record Type 24 - Raw Link Record
DAG 7.5G2	Type 2 - Ethernet Record Type 16 - DSM Color Ethernet record
DAG 7.5G4	Type 2 - Ethernet Record
DAG 8.1SX	Type 1 - PoS HDLC Record Type 2 - Ethernet Record Type 24 - Raw Link Record
DAG 9.2X2	Type 2 - Ethernet Record
DAG 9.2SX2	Type 1 - PoS HDLC Record Type 2 - Ethernet Record Type 24 - Raw Link Record
DAG 10X2-S	Type 2 - Ethernet Record Type 27 - Provenance Record
DAG 10X2-P	Type 2 - Ethernet Record Type 27 - Provenance Record
DAG 10X4-S	Type 2 - Ethernet Record Type 27 - Provenance Record
DAG 10X4-P	Type 2 - Ethernet Record Type 27 - Provenance Record

ERF Types Associated with End of Life DAG Cards

The following table lists the ERF types associated with End of Life DAG cards. These ERF Types may still be produced by software:

Number	Type	Description
0:	TYPE_LEGACY	Old style record
4:	TYPE_AAL5	reassembled AAL5 frame
5:	TYPE_MC_HDLC	Multi-channel HDLC frame
6:	TYPE_MC_RAW	Multi-channel Raw time slot link data
7:	TYPE_MC_ATM	Multi-channel ATM Cell
8:	TYPE_MC_RAW_CHANNEL	Multi-channel Raw link data. Legacy ERF type - for DAG 3.7T and 7.1S only.
9:	TYPE_MC_AAL5	Multi-channel AAL5 frame
10:	TYPE_COLOR_HDLC_POS	HDLC format like TYPE_HDLC_POS, but with the LCNTR field reassigned as COLOR
11:	TYPE_COLOR_ETH	Ethernet format like TYPE_ETH, but with the LCNTR field reassigned as COLOR
12:	TYPE_MC_AAL2	Multi-channel AAL2 frame
15:	TYPE_DSM_COLOR_HDLC_POS	HDLC format like TYPE_HDLC_POS, but with the LCNTR field reassigned as DSM COLOR
17:	TYPE_COLOR_MC_HDLC_POS	Multi-channel HDLC like TYPE_MC_HDLC, but with the LCNTR field reassigned as COLOUR
18:	TYPE_AAL2	Reassembled AAL2 Frame Record
19:	TYPE_COLOR_HASH_POS	Colored PoS HDLC record with Hash load balancing
20:	TYPE_COLOR_HASH_ETH	Colored Ethernet variable length record with Hash load balancing
21:	TYPE_INFINIBAND	InfiniBand Variable Length Record
25:	TYPE_INFINIBAND_LINK	InfiniBand link data.
32-47:	-	Reserved for Co-Processor Development Kit (CDK) Users and Internal use

For details of these ERF Types, refer to version 16 of this document.

Non-Endace ERF Types

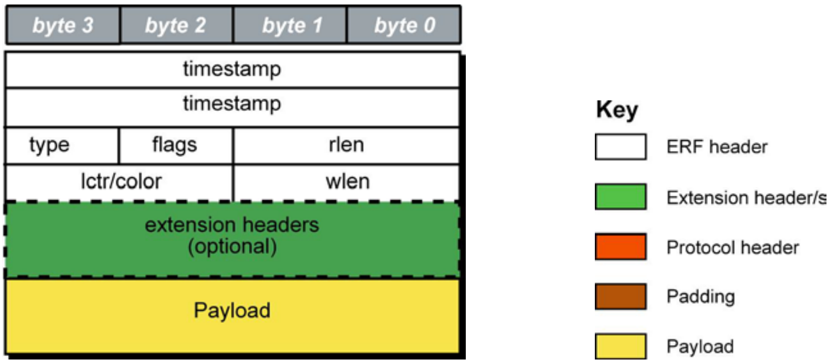
The following table lists the ERF types created by Intel for use with their Omni-Path Architecture (OPA). Endace does not support OPA capture, dissection, or filtering in DAG tools.

Number	Type	Description
28:	ERF_TYPE_OPA_SNC	Intel Omni-Path "Snoop and Capture". Non-Endace ERF type.
29:	ERF_TYPE_OPA_9B	Intel Omni-Path Port link format 9B. Non-Endace ERF type.

Generic ERF header

All ERF records share some common fields. Timestamps are in little-endian (x86 native) byte order. All other fields are in big-endian (network) byte order. All payload data is captured as a byte stream in network order, no byte or re-ordering is applied.

The generic ERF header is shown below:



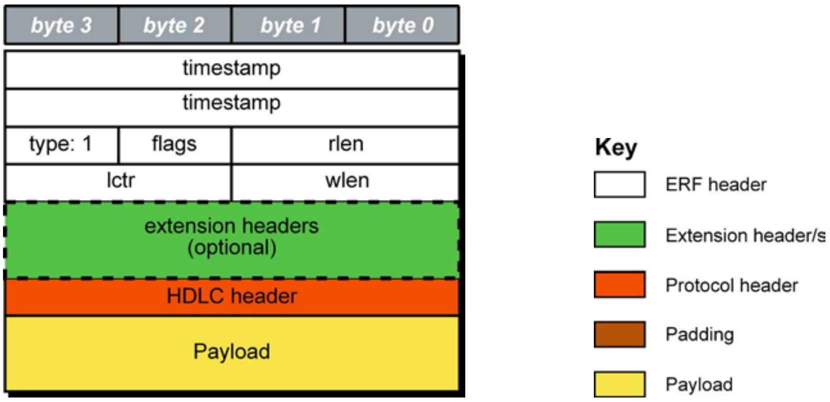
The fields are described below:

<table>
<tr><td>timestamp</td><td></td><td>The time of arrival of the cell, an ERF 64-bit timestamp.</td></tr>
<tr><td rowspan="2">type</td><td>Bit 7</td><td>Extension header present.</td></tr>
<tr><td>Bit 6:0</td><td>ERF type. See table below:</td></tr>
<tr><td>flags</td><td></td><td>This byte is divided into several fields as follows:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>1-0:</td><td>Binary enumeration of capture interface:
11 Interface 3 or D
10 Interface 2 or C
01 Interface 1 or B
00 Interface 0 or A
Cards with more than four interfaces typically use Multichannel ERF types (type 5 to 9, 12 and 17) which provide a separate larger interface field.</td></tr>
<tr><td>2:</td><td>Varying length record (vlen). When set, packets shorter than the snap length are not padded and rlen resembles wlen.
When clear, longer packets are snapped off at snap length and shorter packets are padded up to the snap length. rlen resembles snap length. Setting novarlen and slen greater than 256 bytes is wasteful of bandwidth</td></tr>
<tr><td>3:</td><td>Truncated record - insufficient buffer space.
• <i>wlen</i> is still correct for the packet on the wire.
• <i>rlen</i> is still correct for the resulting record. But, rlen is shorter than expected from snap length or wlen values.
<i>Note: Truncation is depreciated and this bit is unlikely to be set in an ERF record.</i></td></tr>
<tr><td>4:</td><td>RX error. The number of packets received on this port with an error. Errors can be: FCS, short packet, truncated packet, length field, length type, frame code, or frame invalid errors. Present on the wire.
Reported as <code>RX_Protocol_Errors</code> in the universal counter (dagconfig -u).</td></tr>
<tr><td>5:</td><td>DS error. An internal error generated inside the DAG card annotator.
Not present on the wire.
On the DAG 9.2SX2, 9.2X2 and 10X series cards, this error represents a port drop inside the internal data path.</td></tr>
<tr><td>6:</td><td>Reserved</td></tr>
<tr><td>7:</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>rlen</td><td></td><td>Record length in bytes. Total length of the record transferred over the PCI bus to storage.
The timestamp of the next ERF record starts exactly rlen bytes after the start of the timestamp of the current ERF record.</td></tr>
<tr><td>lctr / Color</td><td></td><td>Depending upon the ERF type this 16 bit field is either a loss counter or color field. The <i>loss counter</i> records the number of packets lost between the DAG card and the stream buffer due to overloading on the PCI bus. The loss is recorded between the current record and the previous record captured on the same stream/interface. The <i>color field</i> is explained under the appropriate ERF type details.</td></tr>
<tr><td>wlen</td><td></td><td>Wire length. Packet length "on the wire" including some protocol overhead. The exact interpretation of this quantity depends on physical medium. This may contain padding.</td></tr>
<tr><td>extension headers</td><td></td><td>Extension headers in an ERF record allow extra data relating to each packet to be transported to the host. Extension header/s are present if bit 7 of the type field is '1'. If bit 7 is '0', no extension headers are present (ensures backwards compatibility).
<i>Note: There can be more than one Extension header attached to a ERF record.</i></td></tr>
<tr><td>Payload</td><td></td><td>Payload is the actual data in the record. It can be calculated by either :
• Payload = rlen - ERF header - Extension headers (optional) - Protocol header - Padding</td></tr>
</table>

ERF 1. TYPE_POS_HDLC

Type	Bit 7	1 = Extension header present. See Extension Headers (page 17).
	Bits 6:0	Type 1
Short description	TYPE_POS_HDLC	
Long description	Type 1 PoS HDLC Record	
Use	This record format is for HDLC data links. For example: • Packet over SONET • Point-to-Point Protocol [PPP] over SONET • Frame Relay • MTP2 (SS7) May be used with EH 12. Channelization (page 20) when records have been reconstructed from Channelized TYPE_RAW_LINK records by software. May be used with EH 14. Packet Signature (page 23).	

The *TYPE_POS HDLC* record is shown below:



The following is a description of the *TYPE_POS_HDLC* record format:

Field	Description
HDLC Header (4 bytes)	Protocol Header. Length may vary depending on protocol, typically 4 bytes.
Payload (bytes of record)	Payload = rlen - ERF header (16 bytes) - Extension headers (optional) - Protocol header (4 bytes)

ERF 2. TYPE_ETH

Type	Bit 7	1 = Extension header present. See Extension Headers (page 17).
	Bits 6:0	Type 2
Short description	TYPE_ETH	
Long description	Type 2 Ethernet Record	
Use	This record format is for Ethernet [802.3] data links. May be used with the following Extension Headers: • EH 14. Packet Signature (page 23) • EH 16. Flow ID (page 24)	

The *TYPE_ETH* record is shown below:

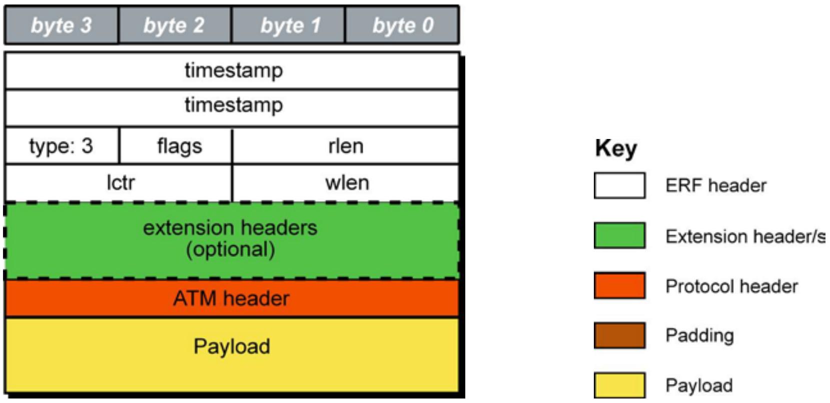
The following is a description of the *TYPE_ETH* record format:

Field	Description
Offset (1 byte)	This field is currently not implemented, contents should be disregarded.
Pad (1 byte)	The Ethernet frame begins immediately after the pad byte so that the layer 3 [IP] header is 32-bit aligned.
Payload (bytes of record)	Payload = rlen - ERF header (16 bytes) - Extension headers (optional) - Padding (2 bytes)

ERF 3. TYPE_ATM

Type	Bit 7	1 = Extension header present. See Extension Headers (page 17).
	Bits 6:0	Type 3
Short description	TYPE_ATM	
Long description	Type 3 ATM Cell Record	
Use	This record format is for ATM cell capture.	

The *TYPE_ATM* record is shown below:



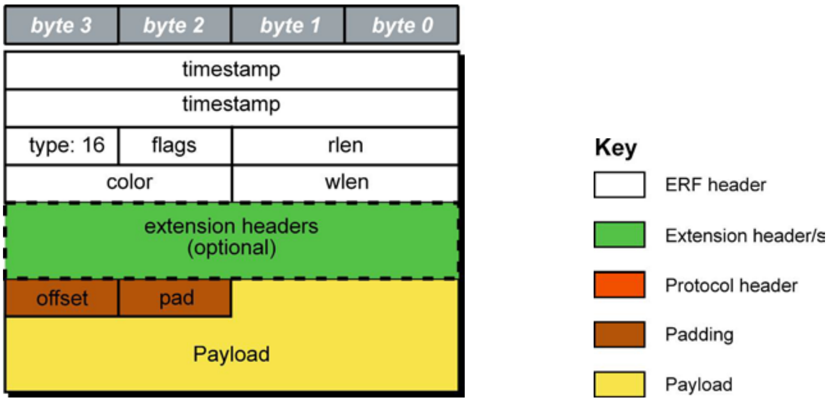
The following is a description of the *TYPE_ATM* record format:

Field	Description
ATM Header (4 bytes)	Protocol header. Does not include the 8-bit HEC.
Flags (1 byte)	ATM cells should not have the variable length flag set.
Payload (bytes of cell)	Payload = 48 bytes of cell

ERF 16. TYPE_DSM_COLOR_ETH

Type	Bit 7	1 = Extension header present. See Extension Headers (page 17).
	Bits 6:0	Type 16
Short description	TYPE_DSM_COLOR_ETH	
Long description	Type 16 DSM Color Ethernet Record	
Use	This record format is for Ethernet [802.3] data links, incorporating filter results. The record format is the same type as the Type 2 TYPE_ETH (page 8) record, with the exception that the *lctr* field reassigned as *DSM* type *color*.	

The *TYPE_DSM_COLOR_ETH* record is shown below:



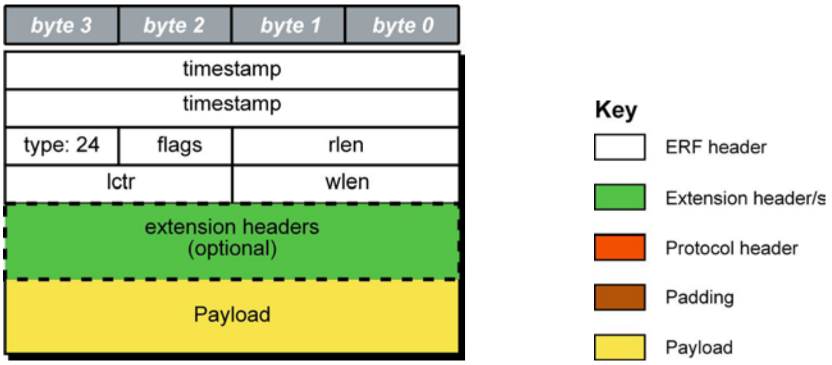
The following is a description of the *TYPE_DSM_COLOR_ETH* record format:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Color
(2 bytes)</td><td>The color field is a hardware generated tag indicating the result of a filtering or classification operation.
This field is divided into the following:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0-5</td><td>Receive stream number (0-63)</td></tr>
<tr><td>6-13</td><td>Filter match bits (bit6 = filter0, bit7 = filter1 and so on).</td></tr>
<tr><td>14</td><td>hlb0 (CRC calculation) output bit.</td></tr>
<tr><td>15</td><td>hlb1 (parity calculation) output bit.</td></tr>
</table></td></tr>
<tr><td>Offset
(1 byte)</td><td>Number of bytes not captured from the start of the frame. This is typically used to skip link layer headers when they are not required in order to save bandwidth and space.
<i>Note:</i>
<i>This field is currently not implemented; contents should be disregarded.</i></td></tr>
<tr><td>Pad
(1 byte)</td><td>The Ethernet frame begins immediately after the pad byte so that the layer 3 [IP] header is 32-bit aligned. This is typically used to skip link layer headers when they are not required in order to save bandwidth and space.</td></tr>
<tr><td>Payload
(bytes of record)</td><td>Payload = rlen - ERF header (16 bytes) - Extension headers (optional) - Padding (2 bytes)</td></tr>
</table>

ERF 24. TYPE_RAW_LINK

Type	Bit 7	1 = Extension header present. See Extension Headers (page 17).
	Bits 6:0	Type 24
Short description	TYPE_RAW_LINK	
Long description	Type 24 Raw link data, typically SONET or SDH Frame	
Use	Used in Raw SONET/SDH capture. Used with EH 5. Raw_Link (page 19) and EH 12. Channelization (page 20).	

The *TYPE_RAW_LINK* record is shown below:



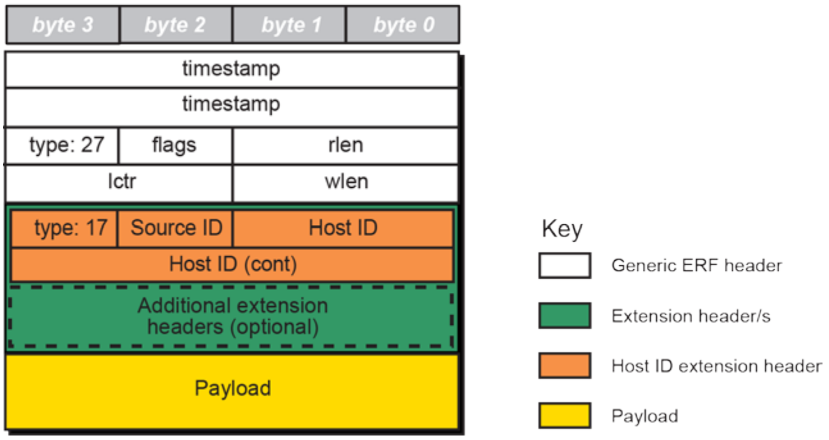
The following is a description of the *TYPE_RAW_LINK* record format:

Field	Description
Payload (bytes of record)	Payload = rlen - ERF header (16 bytes) - Extension headers (optional)

ERF 27. TYPE_META

Type	Bit 7	1 = Host ID extension header present.
	Bits 6:0	Type 27 (0x1b)
Short description	TYPE_META	
Long description	Type 27 Provenance Record	
Use	This record format contains metadata about the environment in which the original packets were captured - it does not contains any captured packet data. Provenance Records are ignored by all Snap length settings - these records are never "snapped". May be used in conjunction with the following Extension Headers: • EH 16. Flow ID (page 24) • EH 17. Host ID (page 26) (enables the matching of metadata records with packet records (page 13)) *Note:* *This ERF Record Type is not used to encapsulate captured packets like other ERF Record Types.*	

The *TYPE_META* record format is shown below:



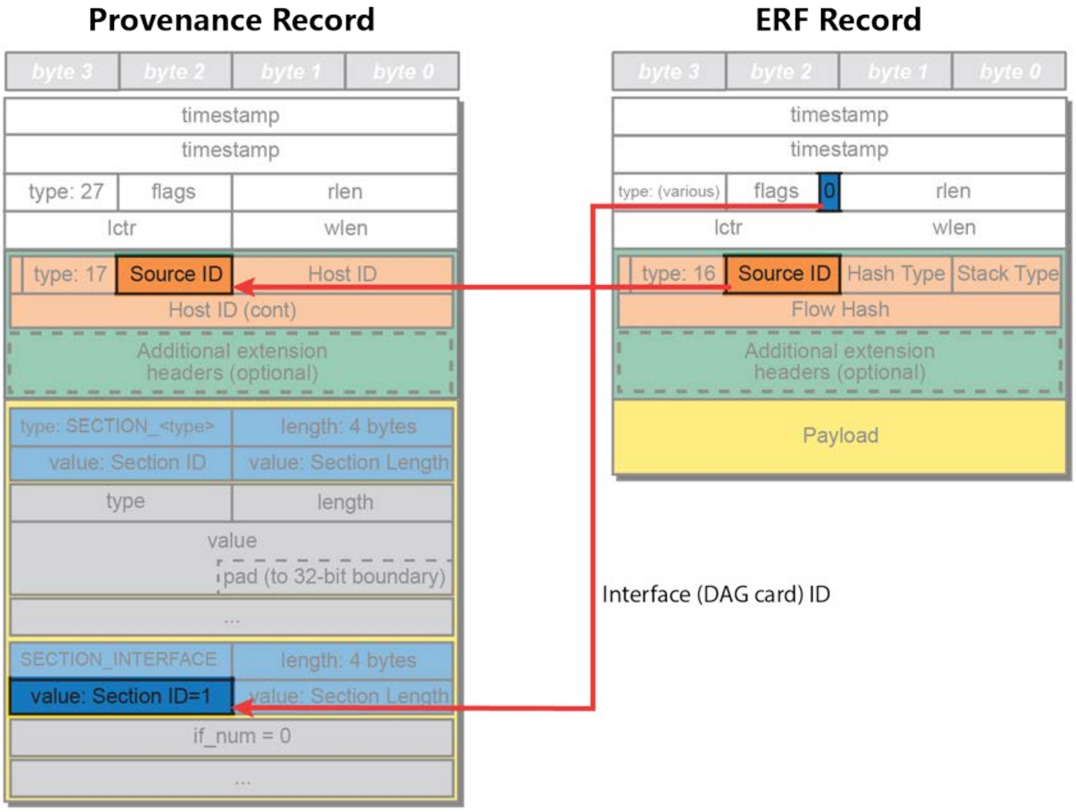
TYPE_META Record Matching

By including:

- a *Host ID* extension header (EH. 17) in the Provenance record, and
- a *Flow ID* extension header (EH. 16) in the ERF packet Records,

ERF records can be paired to the matching Provenance record using the common *Source ID* value.

Additionally, the Provenance record's *Interface Section* contains the metadata information for the interface (port) on the DAG card used for packet capture. The *Interface Section* matches the *Interface Flag* in the ERF Record.



For further information on the extension headers, see:

- EH 16. Flow ID (page 24)
- EH 17. Host ID (page 26)

Note:

If an ERF file is exported to another system or contains data from multiple Host IDs, then Host ID extension headers should be added to all ERF records in order to identify which Host ID they are associated with. This can be done using the dagconvert hostidxthdr *module.*

TYPE_META Payload

This section explains the technical details of the Provenance record payload.

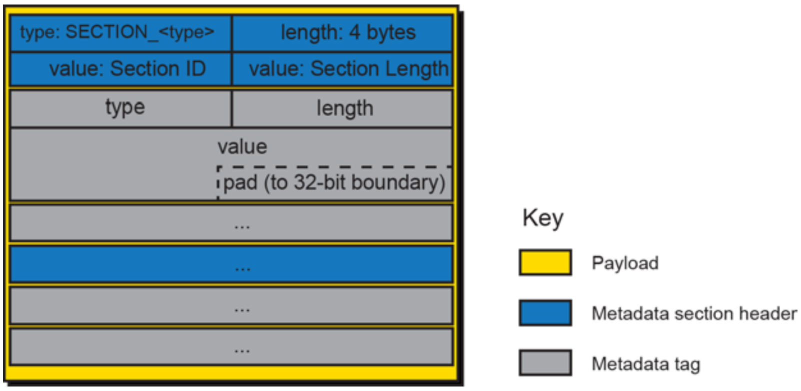
The Provenance record payload must be encoded using a system of 32-bit aligned type-length-value tags which represent metadata information. One or more Section Header tags should be included, describing the type of information in each section.

Section header and metadata tags are Endace-defined.

Notes:

- *Provenance records should not be modified.*
- *Provenance records should include the gen_time tag (metadata generation time).*
- *Endace recommends that the first Host ID extension header contains the Host ID for the system used to capture the ERF records. If the host used to generate the Provenance record is different to the host used to capture the ERF records, the host_id metadata tag should be included in the payload to maintain traceability of the metadata generation.*

The Provenance record payload is shown in the diagram below:



The metadata information is configured using the *dagmeta* configuration file, for more information refer the *dagmeta* section of *EDM04-39 DAG Software Tools Reference Guide* and *EDM04-42 Provenance Guide*.

The following is a description of the Provenance section header tags:

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>type (2 bytes)</td><td colspan="2">SECTION_<type> (0xFFXX)</td></tr>
<tr><td>length (2 bytes)</td><td colspan="2">Set to 0 or 4. If set to 0 no Section ID is used and an unspecified length is assumed.
Additional bytes reserved for future use.</td></tr>
<tr><td>value (4 bytes)</td><td>Section ID
(2 bytes)</td><td>Identifier for Section, local to Source Channel and Section Type. 0 if unset.
0x7FFF for overflow.
Values beginning with 1 in the most significant bit indicate Section ID local to current record and references. 0x8000 if unset, 0xFFFF for overflow.</td></tr>
<tr><td></td><td>Section Length
(2 bytes)</td><td>Section Length in bytes including entire Section Header. If non-zero, tag at Section Header tag offset plus Section Length must be next Section Header tag (or end of record padding, if any). Must be a multiple of 4. May be set to 0, unspecified.</td></tr>
</table>

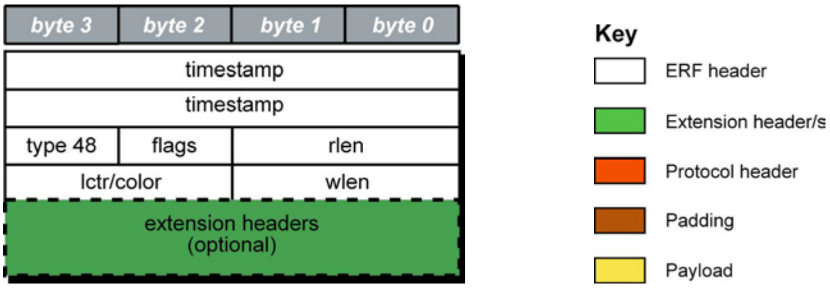
The following is a description of the Provenance metadata tag:

Field	Description
type (2 bytes)	Metadata tag code, for example: `gen_time` (code 2), `comment` (code 1), `hostname` (code 18), etc. For more information on configuring Provenance record payload tags refer to *EDM04-42 Provenance Guide*.
length (2 bytes)	The Length of the *value* field, in bytes. Note that all tags must be 32-bit aligned.
value	Padded to 32-bit (4 bytes) aligned boundary.

ERF 48. TYPE_PAD

Type	Bit 7	1 = Extension header present. See Extension Headers (page 17).
	Bits 6:0	Type 48
Short description	TYPE_PAD	
Long description	Type 48 Pad record	
Use	The pad record type is a record type that does not contain packet data. It has historically been used in situations where records have been required in an ERF stream, and records carrying packet data have not been available.	

The *TYPE_PAD* record is shown below:



The following is a description of the *TYPE_PAD* record format:

Field	Description
timestamp (4 bytes)	All zeros
type (1 byte)	48 (0x30)
flags (1 byte)	A value of 0
rlen (2 bytes)	Record length in bytes. Total length of the record transferred over the PCI bus to storage. The timestamp of the next ERF record starts exactly rlen bytes after the start of the timestamp of the current ERF record.
loss counter/color (2 bytes)	A value of 0
wlen (2 bytes)	A value of 0

Extensible Record Format Timestamps

Overview

The Extensible Record Format (ERF) incorporates a hardware generated timestamp of the packet's arrival.

The format of this timestamp is a single little-endian 64-bit fixed point number, representing whole and fractional seconds since midnight on the first of January 1970.

The high 32-bits contain the integer number of seconds, while the lower 32-bits contain the binary fraction of the second. This allows an ultimate resolution of 2^{-32} seconds, or approximately 233 picoseconds.

Another advantage of the ERF timestamp format is that a difference between two timestamps can be found with a single 64-bit subtraction.

It is not necessary to check for overflows between the two halves of the structure as is needed when comparing UNIX time structures, which are also available to Windows users in the Winsock library.

DAG Card Resolutions

Different DAG cards have different actual resolutions. This is accommodated by the lowermost bits that are not active being set to zero. In this way the interpretation of the timestamp does not need to change when higher resolution clock hardware is available.

Example Code

The following is example code showing how a 64-bit ERF timestamp (erfts) can be converted into a struct timeval representation (tv).

```
unsigned long long lts;
struct timeval tv;
 lts = erfts;
 tv.tv_sec = lts >> 32;
 lts = ((lts & 0xffffffffULL) * 1000 * 1000);
 lts += (lts & 0x80000000ULL) << 1;      /* rounding */
 tv.tv_usec = lts >> 32;
 if(tv.tv_usec >= 1000000) {
   tv.tv_usec -= 1000000;
   tv.tv_sec += 1;
 }
```

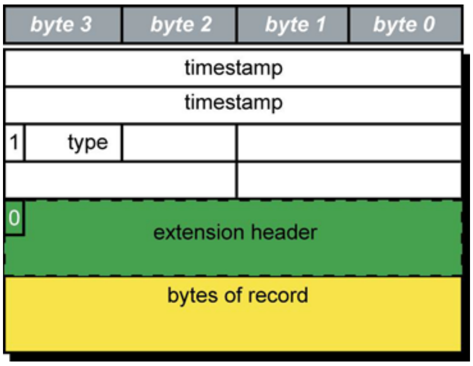
Extension Headers (EH)

Introduction

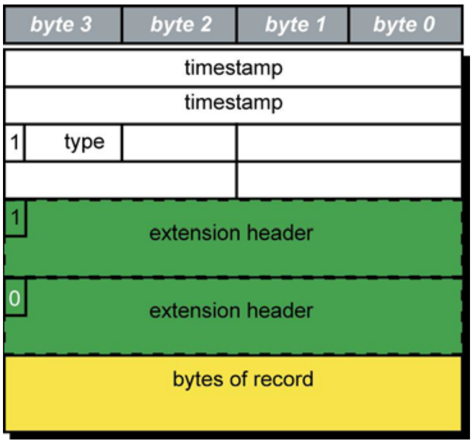
The addition of an Extension Header into the ERF record allows extra data relating to the packet to be transported to the host. The extension header allows certain features to be added independently of ERF types, for example, features shared by different ERF records do not have to be implemented separately. This results in automatic support across ERF types.

Bit 7 of the ERF type field is used to indicate that Extension Headers are present. If set to '1' Extension Headers are present. The Extension Header type field indicates the type and format of the Extension Header. It also indicates whether further Extension Headers are present. If bit 7 of the Extension Header is set to '1' further Extension Headers exist in the record. The Extension Headers are 8 bytes in length.

The following diagram shows presence of an Extension Header in addition to the ERF record.



The following diagram shows presence of two Extension Headers with Bit 7 of the first Extension Header set to '1'.



Extension Header Types

Number	Type	Description
0	Reserved	Reserved.
1	Reserved	Reserved.
2	Reserved	Reserved.
5	Raw_Link	Used in Raw SONET/SDH capture. Additional information for ERF 24. TYPE_RAW_LINK (page 11) records.
12	Channelized	Used in Raw SONET/SDH capture of channelized links. It describes the origin channel, fragmentation and, type of traffic captured.
14	Packet Signature	Used with Enhanced Packet Processing v2.
16	Flow ID	The *Flow ID* extension header allows software acceleration by providing a hardware-based hash performed on header fields from the Ethernet packet.
17	Host ID	Used to distinguish ERF records from multiple capture hosts and sources in the same file.

Extension Headers per DAG Card

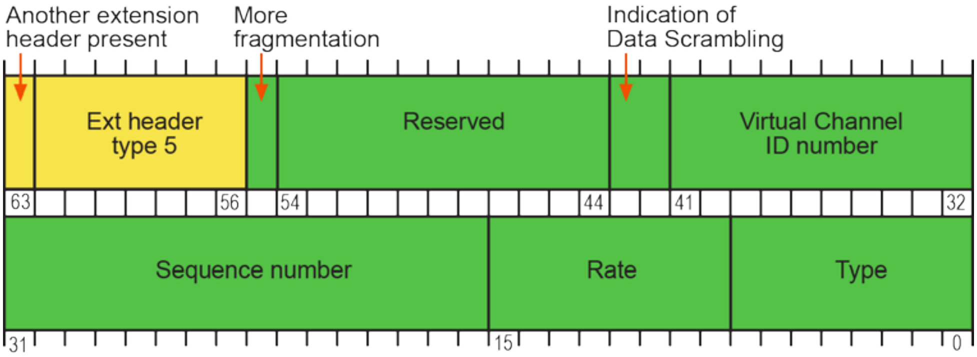
The extension headers available to each DAG card type:

DAG Card	Extension Header		DAG Card	Extension Header
DAG 7.4S	EH.5 - Raw_link EH.14 - Packet Signature		DAG 10X2-S	EH.14 - Packet Signature EH.16 - Flow ID EH.17 - HostID
DAG 7.5G2	none		DAG 10X2-P	EH.14 - Packet Signature EH.16 - Flow ID EH.17 - HostID
DAG 7.5G4	EH.14 - Packet Signature EH.16 - Flow ID		DAG 10X4-S	EH.14 - Packet Signature EH.16 - Flow ID EH.17 - HostID
DAG 8.1SX	EH.5 - Raw_link EH.14 - Packet Signature EH.16 - Flow ID		DAG 10X4-P	EH.14 - Packet Signature EH.16 - Flow ID EH.17 - HostID
DAG 9.2X2	EH.14 - Packet Signature EH.16 - Flow ID			
DAG 9.2SX2	EH.5 - Raw_link EH.12 - Channelized EH.14 - Packet Signature			

EH 5. Raw_Link

Type	Bit 7	Extension header present
	Bits 6:0	Type 5
Short description	Raw_Link	
Long description	Extra information for TYPE_RAW_LINK records	
Use	Used in Raw SONET/SDH capture. Used with ERF 24. TYPE_RAW_LINK (page 11).	

The *Raw_Link* extension header is shown:



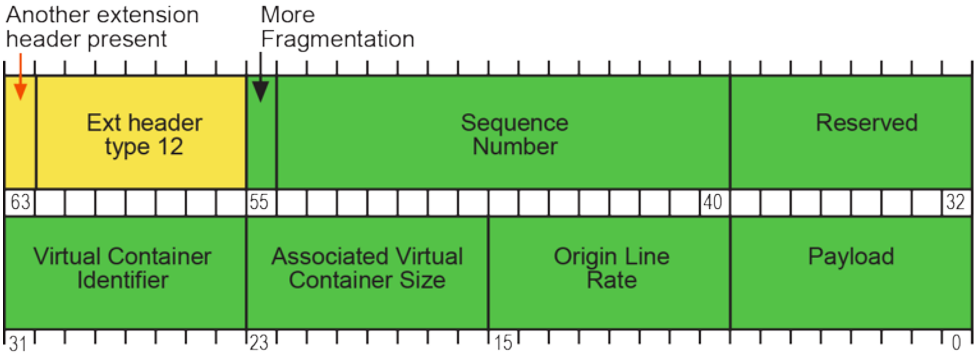
The following details the format of the *Raw_Link* Extension Header:

Bit	Length	Meaning
63	1	More Extension Headers bit (1=more headers)
62:56	7	Extension header type (0x05 / 5).
55	1	More fragmentation. • 1 = More Fragment Expected • 0 = End of Frame (no more fragment expected))
54:44	11	Reserved
43:42		Scrambling of data indication • 00b = Unknown, • 01b = Data has not been descrambled (Frame level) • 10b = Data has been descrambled (Frame level))
41:32	10	Virtual channel identification number, set to 0 if unused.
31:16	16	Sequence number (starting at 0)
15:8	8	Rate. • 0x00 = reserved • 0x01 = OC3 • 0x02 = OC12 • 0x03 = OC48 • 0x04 = OC192 • 0x05 = OC768 • 0x06 = ds3 As defined in the SONET control register.
7:0	8	Type. • 0x00 = raw SONET • 0x01 = raw SDH • 0x02 = SONET spe • 0x03 = SDH spe • 0x04 = ds3 (c-bit) • 0x05 = SONET spe w/o POH • 0x06 = SDH spe w/o POH • 0x07 = SONET line mode 2 • 0x08 = SDH line mode 2 • 0x09 = bit-level raw (no alignment) • 0x0A = raw 10GbE 66b • 0x0B = XGMII Symbols • Others are reserved for future use.

EH 12. Channelization

Type	Bit 7	Extension header present
	Bits 6:0	Type 12
Short description	Channelization	
Long description	Channelization and fragmentation information for TYPE_RAW_LINK records and derived records ERF 1, ERF 3 and ERF24.	
Use	Used in RAW SONET/SDH capture of channelization links. It describes the origin channel, fragmentation and type of traffic captured. Used with ERF 1 TYPE_POS_HDLC (page 7), ERF 3. TYPE_ATM (page 9) and ERF 24. TYPE_RAW_LINK (page 11).	

The *Channelization* extension header is shown below:



The following details the format of the *Channelization* Extension header:

Bit	Length	Meaning
63	1	More Extension Headers present (1 = more).
62:56	7	Extension header type.
55	1	More fragments. • 1 = More Fragments of this frame part expected • 0 = Last fragment of this frame part.
54:40	15	Sequence Number The sequence number identifies this record in the sequence of fragments belonging to a frame part. It is indexed starting at 0 from a fixed point. The fixed point is defined for each part type as follows: **Type** / **Fixed Point** TOH: Start of the SDH Frame – i.e. A1 A2 bytes. POH: Start of the Virtual Container – i.e. the J1 byte associated with the given VC. Container: Start of the Container – i.e. the first byte of the Container (or TUG) occurring after the start of the POH associated with the given VC. POS Packet: Start of the POS packet. ATM Cell: Start of the ATM cell. RAW: Start of the SDH Frame – i.e. A1 A2 bytes The value 0 is given to the first fragment of the given part type that occurs begins at the fixed point, and each subsequent fragment has a incrementing sequence number – i.e. 1,2,3,4. For example, the TOH is associated with the SDH frame – hence, the TOH part containing the A1 A2 bytes will be labeled zero, and each TOH part beyond this will be labeled 1,2,3 etc.
39:32	8	Reserved.

<table>
<tr>
<td>31:24</td>
<td>8</td>
<td>
<p>Virtual Container Identifier</p>
<p>This value identifies the Virtual Container associated with the frame part.</p>
<p>This value is defined as a bitfield, representing the AU-n numbering scheme defined in ITU-T G.707, barring that each number shall range from 0-3 (Or 0-2 for AU-3s), rather than 1-4. In addition, where a bitfield is unused, the field shall be set to zero, meaning that only four values are used (0-3), rather than the five values in ITU-T G.707 (0-4, where 0 means ‘unused’), as whether the value is unused or not can be determined from the Associated Virtual Container Size field.</p>
<p>For later extension, the highest order AUG will be placed in the highest bitfield position.</p>
<p>The bitfield is assigned as such:</p>
<table>
<tr><th>Bits</th><th>AU</th><th>ITU-T Address letter</th></tr>
<tr><td>7:6</td><td>AU-4-16c</td><td>D</td></tr>
<tr><td>5:4</td><td>AU-4-4c</td><td>C</td></tr>
<tr><td>3:2</td><td>AU-4</td><td>B</td></tr>
<tr><td>0:1</td><td>AU-3</td><td>A</td></tr>
</table>
<p>This value hence unambiguously identifies the position of the Virtual Container within the frame.</p>
<p>If this field is unused, then it shall be set to a value of zero.</p>
<p>For example, the VC-4 channel in STS192 /STM-64 defined in G.707 as VC (1,2,3,0) is given the virtual container identifier 0b0110_0000 = 0x60. This is distinguished from the the VC-3 channel VC (1,2,3,1) by the Associated Virtual Container Size field of the extension header.</p>
<p><i>Note:</i>
<i>Please consult respective DAG Card User Guide for supported configurations.</i></p>
</td>
</tr>
<tr>
<td>23:16</td>
<td>8</td>
<td>
<p>Associated Virtual Container Size</p>
<p>Set to one of the following values to indicate the Virtual Container size associated with the frame part:</p>

0x00 - Indicates field is unused
0x01 - VC-3 (STS-1)
0x02 - VC-4 (STS-3)
0x03 - VC-4-4c (STS-12)
0x04 - VC-4-16c (STS-48)
0x05 - VC-4-64c (STS-192)

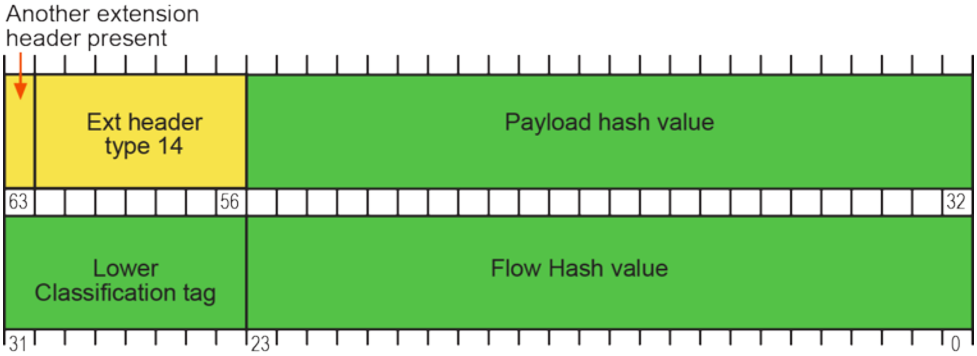
<p>Other values are reserved.</p>
<p><i>Note:</i>
<i>Please consult respective DAG Card User Guide for supported configurations.</i></p>
</td>
</tr>
<tr>
<td>15:8</td>
<td>8</td>
<td>
<p>Origin Line Rate</p>
<p>Set to one of the following values to indicate what physical line type and speed this frame part was captured from:</p>
<table>
<tr><th>Value</th><th>SONET</th><th>SDH</th></tr>
<tr><td>0x00</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>0x01</td><td>STS-1 / OC-1</td><td>STM-0</td></tr>
<tr><td>0x02</td><td>STS-1 / OC-3</td><td>STM-1</td></tr>
<tr><td>0x03</td><td>STS-12 / OC-12</td><td>STM-4</td></tr>
<tr><td>0x04</td><td>STS-48 / OC-48</td><td>STM-16</td></tr>
<tr><td>0x05</td><td>STS-192 / OC-192</td><td>STM-64</td></tr>
</table>
<p>Other values are reserved.</p>
</td>
</tr>
</table>

7:0	8	Payload Set to one of the following values to indicate the content of the record: • 0x00 - TOH (de-multiplexed). • 0x01 - POH • 0x02 - Container • 0x03 - POS Packet • 0x04 - ATM Cell • 0x05 - RAW(de-multiplexed) Other values are reserved.

EH 14. Packet Signature

Type	Bit 7	Extension header present
	Bits 6:0	Type 14
Short description	Packet Signature	
Long description	Packet Signature header contains both Payload hash and Flow hash information for acceleration, and short color for classification.	
Use	Used with Enhanced Packet Processing v2 to enhance packet processing.	

The *Packet Signature* extension header is shown below:



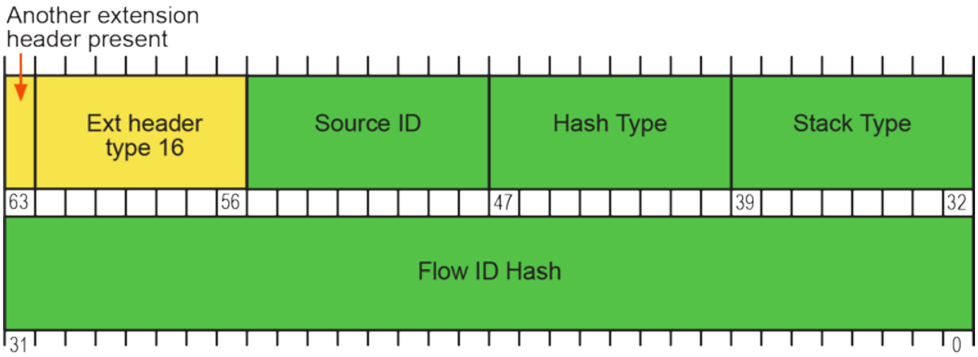
The following details the format of the *Packet Signature* Extension Header:

Bit	Length	Meaning
63	1	More Extension Headers present (1 = more).
62:56	7	Extension header type (0x0e / 14).
55:32	24	Payload hash value. Low 24-bits of the CRC32 payload hash. The Payload Hash Value always excludes all recognized packet headers, and calculates the hash over the highest level payload found. The n_tuple_value attribute (configured via EPPv2) **has no effect** on this calculation.
31:24	8	Lower 8 bits of classification tag (color).
23:0	24	Flow hash value. Low 24-bits of the Ethernet CRC hash calculated by the Hash Load Balancer engine. The set of fields used depends on the n_tuple_value configured as per EPPv2.

EH 16. Flow ID

Type	Bit 7	Indicates if another extension header is present. • 1 indicates another extension header is present. • 0 indicates this is the final extension header.
	Bits 6:0	Type 16
Short description	*Flow ID* Extension Header	
Long description	*Flow ID* extension header contains flow hash, stack type, hash type for acceleration and the Source ID. The *Source ID* identifies the source of the ERF record, such as which DAG card it was captured on. This *Source ID* is then matched with the Source ID and Host ID in the EH 17. Host ID (page 26) extension header attached to Provenance records (ERF 27. TYPE_META (page 12)), enabling ERF records to be matched to Provenance records.	
Use	The *Flow ID* extension header allows software acceleration by providing a hardware-based hash performed on header fields from the Ethernet packet. Typical uses include flow state hash tables or software-based load balancing. The Flow ID extension header replaces the Packet Signature extension header for use with Enhanced Packet Processing v2. For more information on Enhanced Packet Processing v2, refer to the *EDM04-31 Enhanced Packet Processing v2*.	

The *Flow ID* extension header is shown below:



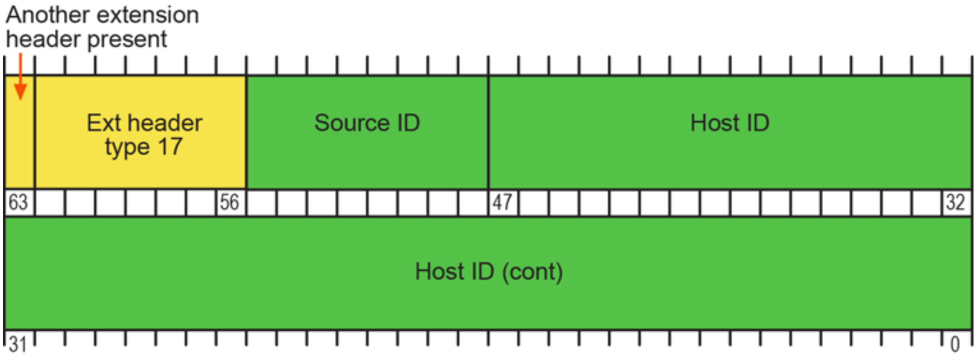
The following details the format of the *Flow ID* Extension Header:

<table>
<tr><th>Bit</th><th>Length</th><th>Description</th></tr>
<tr><td>63</td><td>1</td><td>More Extension Headers present (1 = more).</td></tr>
<tr><td>62:56</td><td>7</td><td>Extension header type (0x10 / 16).</td></tr>
<tr><td>55:48</td><td>8</td><td><i>Source ID</i>. This field is configurable per DAG card. When configured per DAG card, it enables records received on multiple DAG cards to be distinguished from each other.
If this extension header is included in an ERF record (rather than a Provenance record), the Source ID must match the Source ID in the <i>Host ID</i> extension header for the same ERF record.
If the Source ID in the Flow ID extension header is zero this does not apply.
Additional extension headers can be added to contain further source information.
If set to:
0x00 = unset.
0xff = Id overflow.</td></tr>
<tr><td>47:40</td><td>8</td><td><i>Hash Type</i>. An enumerated field indicating which packet fields were used to generate the Flow Hash.
The values <code>0x02</code> to <code>0x06</code> are equivalent to EPPv2 n_tuple_mode settings.
<table>
<tr><th>Value</th><th>Name</th><th>Fields</th></tr>
<tr><td><code>0x00</code></td><td>Not Set</td><td>None</td></tr>
<tr><td><code>0x01</code></td><td>Non-IP</td><td>Src/Dst MACs, EtherType</td></tr>
<tr><td><code>0x02</code></td><td>2-tuple</td><td>Src/Dst IPv4/6 Addresses</td></tr>
<tr><td><code>0x03</code></td><td>3-tuple</td><td>Src/Dst IPv4/6 Addresses, IP Protocol</td></tr>
<tr><td><code>0x04</code></td><td>4-tuple</td><td>Src/Dst IPv4/6 Addresses, IP Protocol, Interface Id</td></tr>
<tr><td><code>0x05</code></td><td>5-tuple</td><td>Src/Dst IPv4/6 Addresses, IP Protocol, Src/Dst L4 Ports</td></tr>
<tr><td><code>0x06</code></td><td>6-tuple</td><td>Src/Dst IPv4/6 Addresses, IP Protocol, Src/Dst L4 Ports, Interface Id</td></tr>
</table></td></tr>
<tr><td>39:32</td><td>8</td><td><i>Stack Type</i>. Enumerated type field of common protocol stacks for packet decode acceleration. Used to implement a look up table of static protocol offsets and/or function handlers.
<table>
<tr><th>Value</th><th>Description</th><th></th><th>Value</th><th>Description</th></tr>
<tr><td>0x00</td><td>Not set</td><td></td><td>0x04</td><td>one VLAN, IPv4</td></tr>
<tr><td>0x01</td><td>Non-IP</td><td></td><td>0x05</td><td>one VLAN, IPv6</td></tr>
<tr><td>0x02</td><td>no VLAN, IPv4</td><td></td><td>0x06</td><td>two VLANs, IPv4</td></tr>
<tr><td>0x03</td><td>no VLAN, IPv6</td><td></td><td>0x07</td><td>two VLANs, IPv6</td></tr>
</table></td></tr>
<tr><td>31:0</td><td>32</td><td><i>Flow ID hash</i>. CRC-32c hash calculated over a linear tuple construction. The set of fields used depends on the <i>n_tuple_value</i> configured as per EPPv2. This hash is bi-directionally flow-safe.
Note: <i>This value is not the same as the ERF BFS extension header hash.</i></td></tr>
</table>

EH 17. Host ID

Type	Bit 7	Indicates if another extension header is present. • 1 indicates another extension header is present. • 0 indicates this is the final extension header.
	Bits 6:0	Type 17
Short description	*Host ID* Extension Header	
Long description	*Host ID* extension header contains Source ID and Host ID. The *Source ID* is used to identify the source of the ERF record, such as which DAG card it was captured on. The Source ID is matched against the Source ID in the *Flow ID* extension header (EH. 16 Host ID (page 24)) contained in the captured packets.	
Use	Used to distinguish ERF records from multiple capture hosts and sources in the same file. The *Host ID* is used to provide an organizationally unique identifier for the capture system.	

The *Host ID* extension header is shown below:



The following details the format of the *Host ID* Extension Header:

Bit	Length	Description
63	1	More Extension Headers present (1 = more).
62:56	7	Extension header type (0x11 / 17).
55:48	8	*Source ID*. This field is configurable per DAG card. When configured per DAG card, it enables records received on multiple DAG cards to be distinguished from each other. If this extension header is included in an ERF record (rather than a Provenance record), the Source ID must match the Source ID in the *Flow ID* extension header for the same ERF record. If the Source ID in the Flow ID extension header is zero this does not apply. Additional extension headers can be added to contain further source information. If set to: • 0x00 = unset. • 0xff = Id overflow.
47:0	32	*Host ID*. Organizationally unique Host Identifier. 0 represents an unset/unknown value.

Output Formats

Endace products including DAG cards, EndaceProbes and EndaceAccess output records in the following formats:

- ERF
  See Extensible Record Format (page 3)
- Endace_ETH
  See Endace_ETH Format (page 28)
- E3 - Endace Ethernet Encapsulation
  See E3 (Endace Ethernet Encapsulation) Format (page 29).

Endace_ETH Format

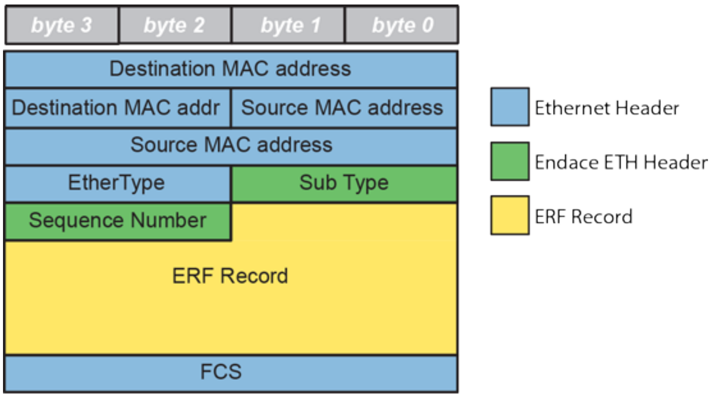
The Endace_ETH format is an Endace proprietary Ethernet frame format.

The Endace_ETH format is used to wrap a standard Endace ERF record within an Ethernet frame. This preserves the original packets time stamp and meta information. The encapsulated ERF record can be sent to another DAG card on another EndaceProbe without the receiving DAG card applying a new time stamp or otherwise modifying the original ERF record.

Note:
Packet decapsulation is performed automatically by the EndaceProbe software - but not by DAG software.

Format

The Endace_ETH packet format is:



Field	Description
Ethernet Header	
Destination MAC Address (6 bytes)	The Destination MAC address of the Endace_ETH frame.
Source MAC Address (6 bytes)	The Source MAC address of the Endace_ETH frame.
Ethertype (2 bytes)	0x88b5 Currently using Local Experimental EtherType.
Endace_ETH Header	
Sub Type (2 bytes)	Indicates the encapsulation type of Endace_ETH. The currently value is 0.
Sequence Number (2 bytes)	The sequence number of the frame. Increments by 1 for each frame on each interface. Used to keep track of dropped packets.
ERF Record	
ERF Record (Variable length)	Contains a single ERF record. The ERF record can be of any valid ERF type.
FCS	
FCS (4 bytes)	Frame Check Sequence.

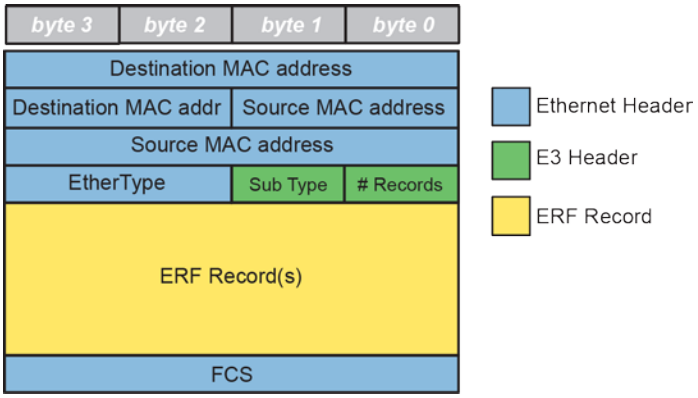
E3 (Endace Ethernet Encapsulation) Format

The E3 (Endace Ethernet Encapsulation) format is an Endace proprietary Ethernet frame format.

It is used by the EndaceAccess to wrap an Endace ERF record within an Ethernet frame, thus preserving the original packets time stamp and meta information. The Ethernet Frame has a specific EtherType which can be recognized automatically by the DAG 9.2X2 and de-encapsulated. For further details, refer to *EDM09-90 EndaceAccess User Guide*.

Format

The format of the E3 (Endace Ethernet Encapsulation) Ethernet frame is:



<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Ethernet Header</td><td></td></tr>
<tr><td>Destination MAC Address
(6 bytes)</td><td>The Destination MAC address of the E3 Ethernet frame.</td></tr>
<tr><td>Source MAC Address
(6 bytes)</td><td>The Source MAC address of the E3 Ethernet frame.</td></tr>
<tr><td>EtherType
(2 bytes)</td><td>The EtherType of the E3 Ethernet frame is
0x894a (IEEE allocated)
This EtherType is automatically recognized by the DAG 9.2X2. The EtherType is user configurable.</td></tr>
<tr><td>E3 Header</td><td></td></tr>
<tr><td>Sub Type
(1 byte)</td><td>Indicates which E3 format is used in the Ethernet frame. The current format is:<table><tr><th>Option</th><th>Description</th></tr><tr><td>0x1</td><td>E3 is on.</td></tr></table>If the number of records field is greater than one, then multiple ERF records are packed into the Ethernet frame. Otherwise only one ERF record is included.
In Multi ERF mode, the packet is transmitted when the either 8000 Bytes of payload is available or 4 microseconds of time has past since the first ERF record arrived.</td></tr>
<tr><td># Records (1 byte)</td><td>The number of records included with in this E3 frame.</td></tr>
<tr><td>ERF Record</td><td></td></tr>
<tr><td>ERF Record
(Variable length)</td><td>Contains a one or more ERF record(s) with no extension headers. The ERF record can be of Type-2 only (Ethernet traffic) and must be 64-bit aligned.
The total number of bits in the encapsulated ERF records shall not exceed 9624 Bytes (ERF-Records with Jumbo Ethernet Frames 9600B).</td></tr>
<tr><td>FCS</td><td></td></tr>
<tr><td>FCS (4 bytes)</td><td>Frame Check Sequence.</td></tr>
</table>

Note:

E3 Frames shall:

- *not exceed 9644 bytes for subtype 1.*
- *contain a minimum of 1 and a maximum of 127 ERF-Records*

All devices receiving E3 frames must support the maximum frame size of 9644.

Version History

Version	Date	Reason
1 - 2	-	Previous versions
3	October 2005	
4	August 2007	Added new data formats and updated existing data formats.
5	November 2007	Added Extension Headers 3,4 and records 19,20,22,23.
6	December 2007	Added ERF Type 21 and updated ERF types per DAG card
7	February 2008	Added ERF type 24 and EH 5. Defined Payload field in ERF types.
8	June 2008	Corrected ERF types per card information for the 5.4 and 5.4A DAG cards.
9	August 2009	Added DAG 8.5IF, ERF Type 25 and EH 6, updated ERF 6. Updated for DAG software release 3.4.1.
10	September 2010	4.0.1 Release. Rebrand. Added 7.5G2/G4 and 9.2X2 information. Changed name of 7.4S and 8.5I.
11	August 2011	Updated title. Added EH 12. Updated ERF types per DAG card list and added DAG 9.2SX2 to list.
12	October 2011	Updated EH 5. Raw_link section with new parameters for DAG 4.6 Bit-level-raw additions.
13	May 2012	DAG 4.2.2 release. Updated table line styles. Updated generic ERF header description.
14	December 2012	OSM 5.1 release. Added EH 14. Added details about Endace_ETH. Updated EH 5 and EH 12.
15	August 2013	EA 1.2 release. Added E3 (Endace Ethernet Encapsulation) details. Update Endace ETH format information. Correct details in EH 14.
16	January 2014	DAG 5.0 release. Updated EH 5.
17	September 2014	DAG 5.2 release. Renamed DAG cards to EndaceDAG cards. Remove ERF types and Extension Headers associated with End Of Life EndaceDAG cards.
18	July 2015	DAG 5.4.0 release. Added EH 16 Flow ID. Updated front and back pages.
19	March 2016	DAG 5.5.0 release. Rebrand back to Endace. Added ERF Type 27 Metadata and EH 17 Host ID. Moved ERF Types 9 and 32-47 to ERF Types Associated with EOL DAG Cards. Updated Use descriptions in ERF Types 1 and 2, and EH 16. Changed Copyright to use the Creative Commons Attribution-NoDerivatives 4.0 International (CC BY-ND 4.0) License
20	November 2016	DAG 5.5.1 release. Added: Non-Endace ERF types. Updated: Type 27 description, now Provenance Record, EH. 16 and EH. 17. Updated wording to Provenance, ERF Type 48 usage description updated. Removed: DAG 9.2SX2 from Provenance.
21	May 2017	DAG 5.6. Add reference to the DAG 10X4-S card. Updated Extension header 16 and 17 descriptions; Generic ERF type description - RX error, DS error; Reordered Extension header types table. Removed Support section.

endace