

# Event-Based Monitoring Interface Description

Ericsson Service-Aware Policy Controller

## INTERWORK DESCRIPTION

**Copyright**

© Ericsson España, S.A. 2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

**Trademark List**

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



# Contents

<b>1</b>	<b>Event-Based Monitoring Interface Overview</b>	<b>1</b>
1.1	Typographic Conventions	1
<b>2</b>	<b>Description</b>	<b>1</b>
<b>3</b>	<b>Output Format Elements</b>	<b>2</b>
<b>4</b>	<b>XML File</b>	<b>2</b>
4.1	General	3
4.2	Parameter Types	4
4.3	Structure Types	7
4.4	Events	8
4.5	Records	9
<b>5</b>	<b>XML File Schema</b>	<b>11</b>
<b>6</b>	<b>Binary Stream Output Format</b>	<b>11</b>
6.1	Logical Format	11
6.2	Data Format	13
6.3	Record Padding	13
6.4	Record Descriptions	14
6.4.1	Stream Header Record	14
6.4.2	Stream Error Record	15
6.4.3	Event Record	15
<b>7</b>	<b>Directory Structure</b>	<b>17</b>





# 1 Event-Based Monitoring Interface Overview

This document describes the format of the Event-Based Monitoring (EBM) output produced by the Service Aware Policy Controller (SAPC).

The supported EBM output is as follows:

- Binary stream in real time to an external system over Transmission Control Protocol (TCP)

To help in the post processing of the output information generated by the EBM function, this document includes the following information:

- The binary stream
- The XML file that defines the format of the binary stream
- The schema file that defines the structure of the XML file
- The relationships between the binary stream, the schema file, and the XML file
- Information on using the schema file to interpret the XML file, and using the XML file to interpret the binary stream

**Note:** This document does not contain the events generated by the SAPC, their cause codes, or the event parameters.

## 1.1 Typographic Conventions

Typographic conventions can be found in the following document:

- [Typographic Conventions](#)

# 2 Description

The contents of the EBM output can be used for Business Intelligence. The EBM output can be streamed in real time to an external system over TCP. An XML file available on the SAPC describes the binary format of the generated streamed output.



## 3 Output Format Elements

The EBM output is composed of a binary stream that is formatted as defined in the XML file and the schema file.

The relationships between the schema file, XML file, and binary stream are shown in Figure 1.

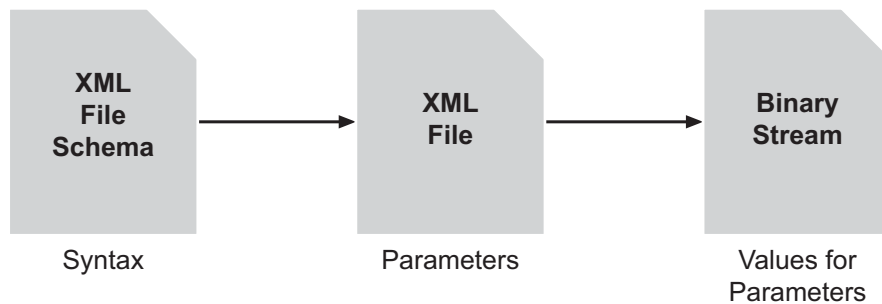


Figure 1 Format Element Relations

## 4 XML File

The XML file, `sapc_ebm.xml`, defines the data format of the binary stream, and is used to decode the binary stream. The structure of the XML file is shown in Figure 2.

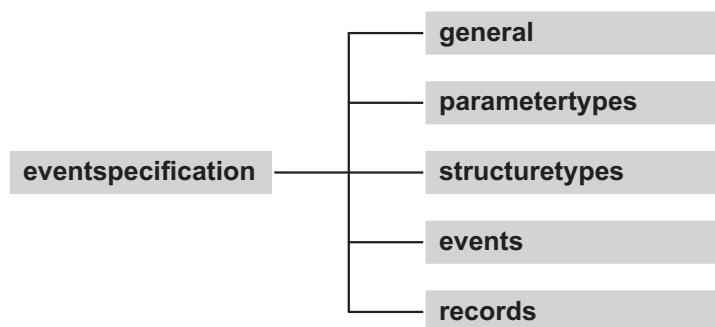


Figure 2 XML File Structure

The XML file contains a detailed description of each element of the binary stream. The following sections describe the individual parts of the XML file.

## 4.1 General

The **general** part of the XML file contains general information about the XML file. The structure of the **general** part is shown in Figure 3.

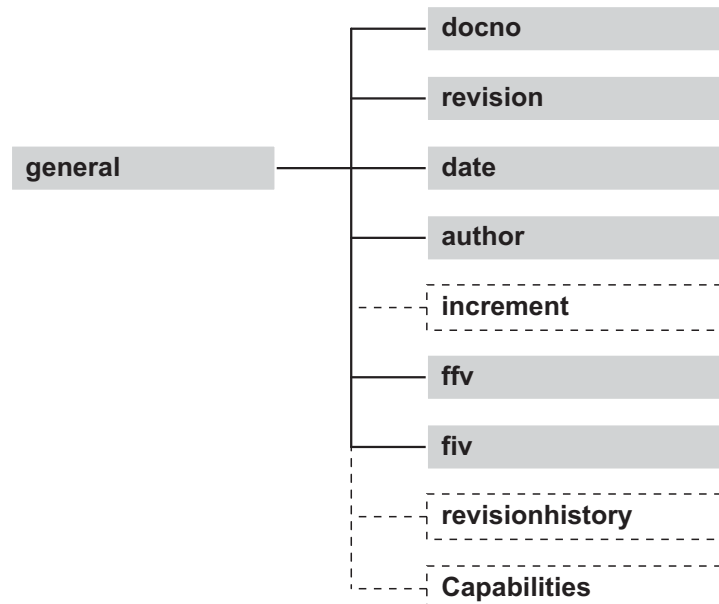


Figure 3 General Information Structure in the XML File

The following example shows how general information is presented in the actual XML file.

```
<general>
  <docno>14/155 19-CSH 109 215/7</docno>
  <revision>PA1</revision>
  <date>2018-02-08</date>
  <author>BDGSLFDG Almudena Mozas</author>
  <ffv>2</ffv>
  <fiv>10</fiv>
</general>
```

Example 1 General Information in the XML File

Element Name	Description
<docno>	This tag specifies the Ericsson identifier of the XML file.
<revision>	This tag specifies the revision identifier of the XML file.
<date>	This tag specifies the date the current revision was created.
<author>	This tag specifies the name and identifier of the person responsible for the current revision.
<increment>	This tag is not used for SAPC.



Element Name	Description
<code>&lt;ffv&gt;</code>	This tag specifies the File Format Version (FFV). The tag is incremented if there is a change in the file format which impacts the existing structure of an event. For example, changes in the file structure that requires updates of the controlling XML schema or new parameters added in the middle of an event. This information is sent in the STREAM HEADER RECORD.
<code>&lt;fiv&gt;</code>	This tag specifies the File Information Version (FIV). The tag is incremented if there is a change in the file information, for example, new events or parameters are added in the end of the events. This information is sent in the STREAM HEADER RECORD.
<code>&lt;revisionhistory&gt;</code>	This tag specifies the information on changes to the XML file over revisions.
<code>&lt;Capabilities&gt;</code>	This tag specifies the administrative information on attributes that are relevant to binary stream handling.

## 4.2 Parameter Types

The **parametertypes** part of the XML file defines parameters that are used as simple entities in the binary stream. Parameters compose the structures defined in the **structuretypes** part.

Parameters defined in the **parametertypes** part and structures defined in the **structuretypes** part are used to construct events and records. For more information on events and records, see Section 4.4 on page 8 and Section 4.5 on page 9.

The structure of the **parametertypes** part in the XML file is shown in Figure 4.



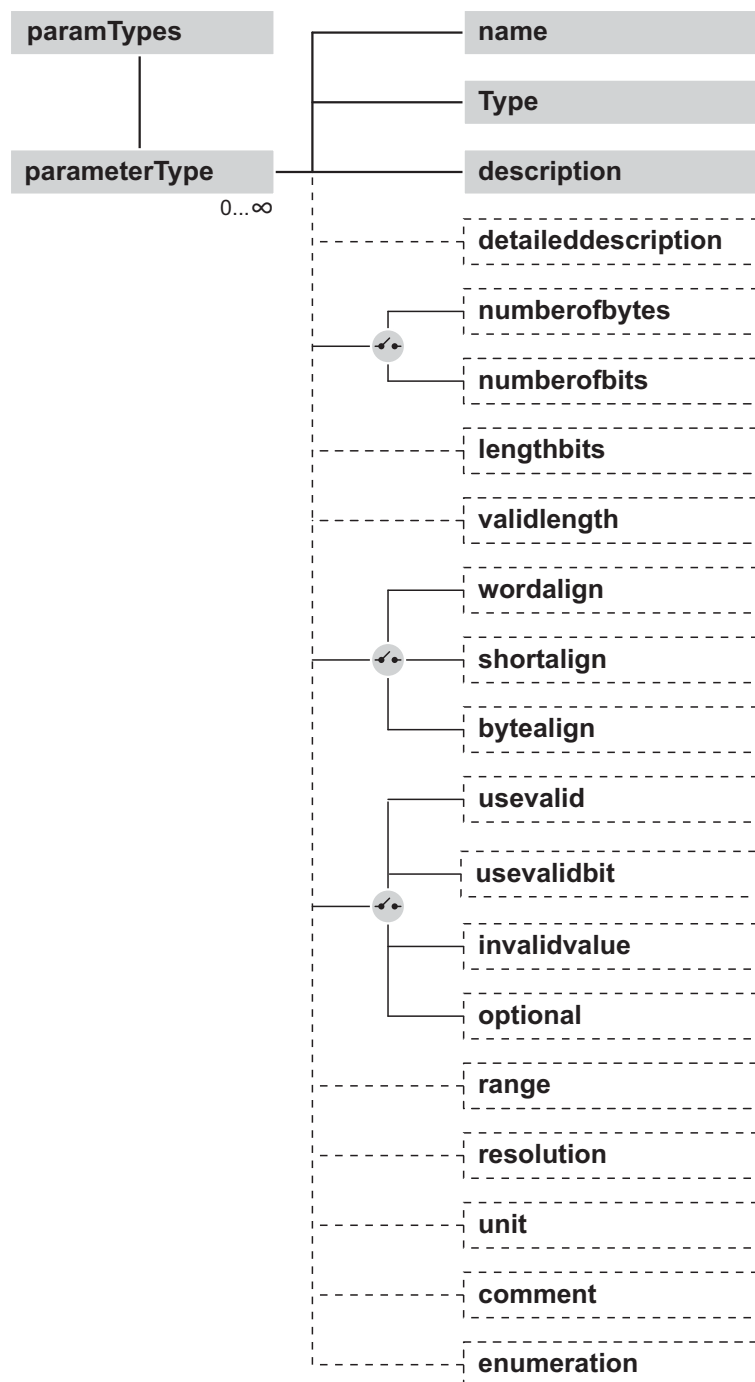


Figure 4 Parameter Types in the XML File

The XML file details the individual parameters in a readable format. Table 1 shows the elements used to describe the parameters.



Table 1 Parameter Type Tag Elements

Position	Elements	Description
0	name	This tag specifies the name of the affected parameter.
1	type	This tag specifies the type of the affected parameter.
2	description	This tag specifies the description of the affected parameter.
3	detaileddescription	This tag specifies the detailed information of the affected parameter.
4	numberofbytes or numberofbits	This tag specifies the length (measured in number of bytes or bits) of the affected parameter.
5	lengthbits	This tag specifies the maximum length (measured in number of bits) required to specify the length of the affected parameter. This tag is used for parameters which can have a variable length. All parameters with lengthbits shall be bytealign. The length field should not be byte aligned but the content should.
6	validlength	This tag specifies the length interval of the valid parameter.
7	wordalign, shortalign or bytealign	This tag specifies the alignment (measured in word, short, or bytes) of the affected parameter.
8	usevalid, usevalidbit, invalidvalue or optional	This tag specifies the validity of the parameter.
9	range	This tag specifies the applicable value range for the parameter.
10	resolution	This tag specifies the resolution of the parameter.
11	unit	This tag specifies the applicable unit of the parameter.
12	comment	This tag specifies the comment of the parameter.
13	enumeration	This tag specifies the possible examples of the parameter.

Some parameters have an optional flag set to **TRUE** in the XML file, indicating that these parameters are optional. The most significant bit of these parameters determines their presence in the event stream. If it is set to 0, the parameter is present in the event stream. If it is set to 1, the parameter is not present in the event stream.

The following example shows how the parameter type is described in the actual XML file.

```
<parametertype>
  <name>QOS_PROFILE_ID</name>
  <type>UINT</type>
  <description>Provides the ID of the QoS Profile.</description>
  <numberofbits>16</numberofbits>
  <range>
    <low>0</low>
    <high>65535</high>
  </range>
</parametertype>
```

## Example 2 Parameter Type Description

In the case of parameters with type STRING, DNSNAME, BYTEARRAY, the actual parameter value is always a multiple of 8-bit words. This is achieved by adding a 0 bit padding as needed between lengthbits XML tag value and the start of the actual parameter value.

The following example shows a 0 bit padding to an optional BYTEARRAY parameter.

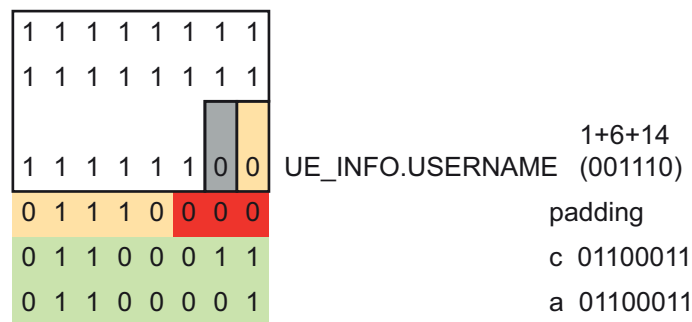


Figure 5 Encode Carlos Valencia as Username

Parameters with type STRING are encoded as UTF-8.

## 4.3 Structure Types

The **structuretypes** part of the XML file defines structures that are used as complex entities in the binary stream.

Some structures can appear several times in the same event, which is referred to as repeated parameter sets. The **seqmaxlen** octet specifies the number of times the structure is repeated for the specific event. In the XML file, these structures have the **seqmaxlen** flag set to the maximum number of times the structure can be repeated for the event.

The structure of the **structuretypes** part in the XML file is shown in Figure 6.

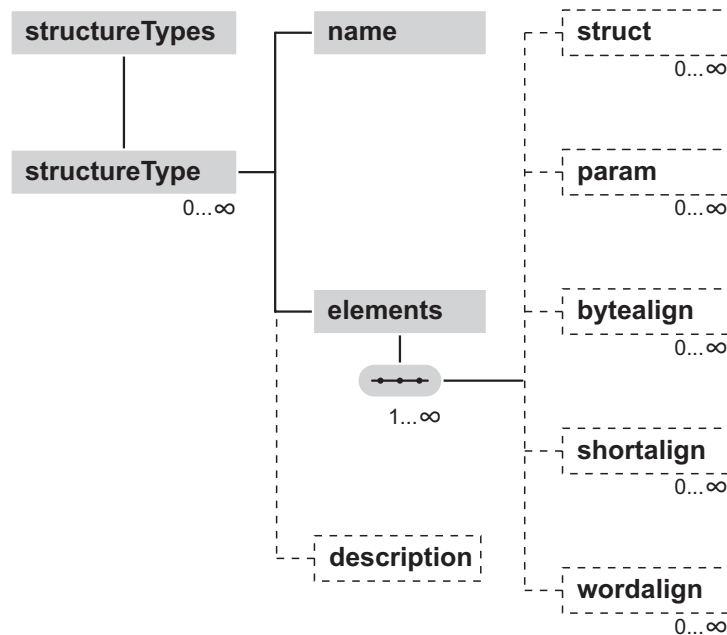


Figure 6 Structure Types in the XML File

The following example shows how PDN structure type is described in the actual XML file.

```

<structuretype>
  <name>PDN_INFO</name>
  <elements>
    <param type="BEARER ID" optional="true" >DEFAULT_BEARER_ID</param>
    <param type="APN" optional="true" >APN</param>
    <struct type="IPADDRESS_STRUCT">ALLOCATED_UE_ADDRESS</struct>
    <param type="RULE SPACE" optional="true">RULE_SPACE</param>
  </elements>
</structuretype>

```

Example 3 PDN Structure Type

## 4.4 Events

The **events** part of the XML file defines the events that are monitored in the binary stream.

Events consist of parameters, defined in the **parametertypes** part of the XML file, and structures, defined in the **structuretype** part of the XML file.

The structure of the events in the XML file is shown in Figure 7.

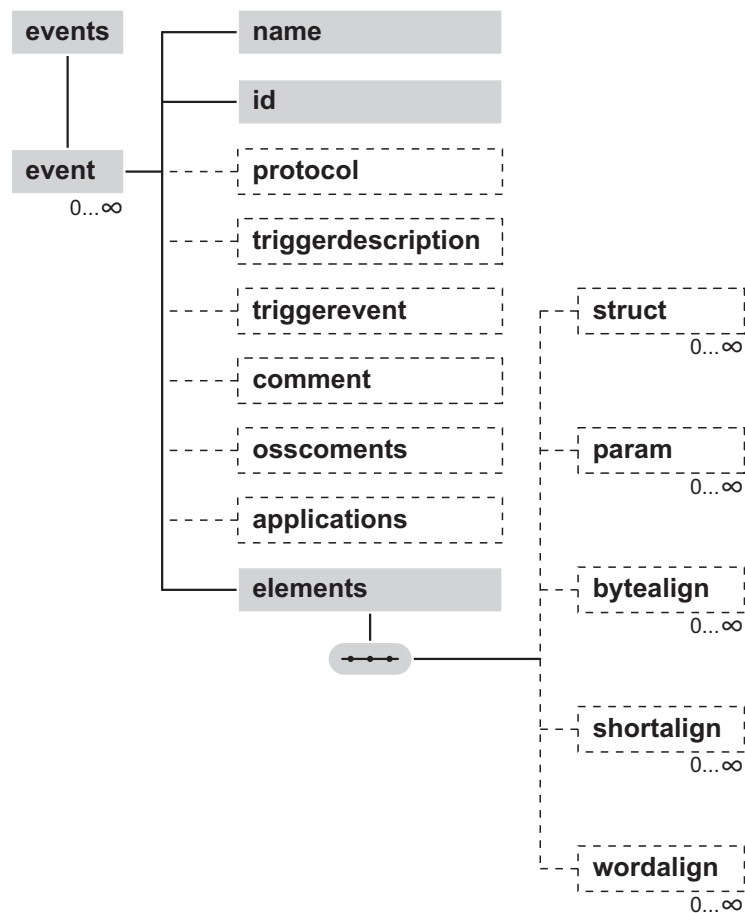


Figure 7 Events Structure in the XML File

The following example shows how the reported usage event is described in the actual XML file.

```

<event>
  <name>REPORTED_USAGE</name>
  <id>7</id>
  <triggerdescription></triggerdescription>
  <comment>Provides usage information</comment>
  <elements>
    <struct type="HEADER">HEADER</struct>
    <param type="CAUSE_PROTOCOL">CAUSE_PROTOCOL</param>
    <param type="CAUSE_CODE">CAUSE_CODE</param>
    <struct type="PDN_INFO">PDN_INFO</struct>
    <struct type="UE_INFO">UE_INFO</struct>
    <param type="SESSION_ID">SESSION_ID</param>
    <struct seqmaxlen="50" type="REPORTING_GROUP_USAGE_INFO">REPORTING_GROUP_USAGE_INFO</struct>
  </elements>
</event>

```

Example 4 Reported Usage Event

## 4.5 Records

The **records** part of the XML file defines the events that are created in the binary stream.

Records consist of parameters, defined in the **parametertypes** part of the XML file, and structures, defined in the **structuretype** part of the XML file.

The structure of the records in the XML file is shown in Figure 8.

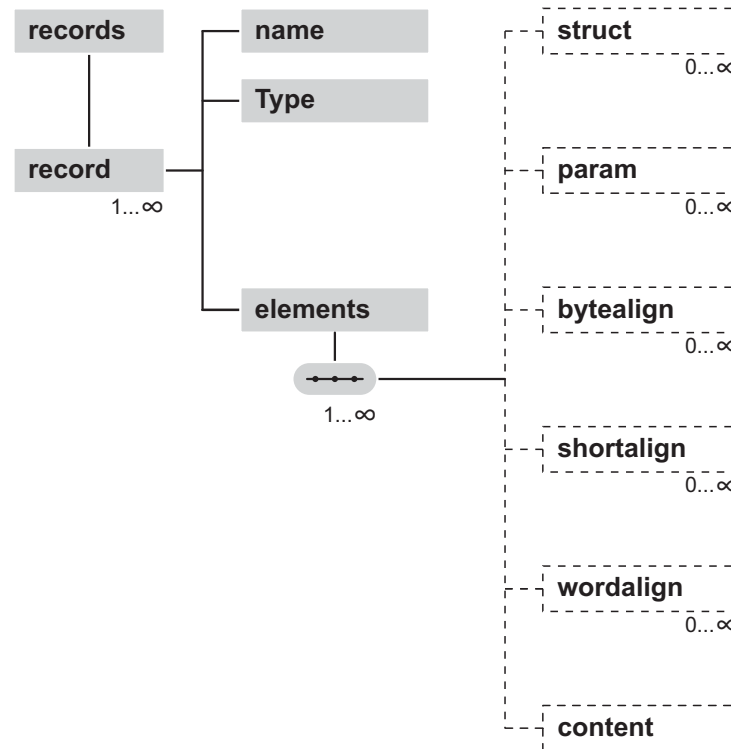


Figure 8 Record Structure in the XML File

The following example shows how the header record is described in the actual XML file.

```

<record>
  <name>HEADER_RECORD</name>
  <type>0</type>
  <elements>
    <param type="RECORD_LENGTH">RECORD_LENGTH</param>
    <param type="RECORD_TYPE">RECORD_TYPE</param>
    <param type="FILE_FORMAT_VERSION">FILE_FORMAT_VERSION</param>
    <param type="FILE_INFORMATION_VERSION">FILE_INFORMATION_VERSION</param>
    <param type="TIME_YEAR">TIME_YEAR</param>
    <param type="TIME_MONTH">TIME_MONTH</param>
    <param type="TIME_DAY">TIME_DAY</param>
    <param type="BYTE_HOUR">BYTE_HOUR</param>
    <param type="BYTE_MINUTE">BYTE_MINUTE</param>
    <param type="BYTE_SECOND">BYTE_SECOND</param>
  </elements>
</record>

```

Example 5 Header Record



## 5 XML File Schema

The `Event_Format_description.xsd` schema file defines the syntax of the XML file, and is used to validate the XML file structure.

## 6 Binary Stream Output Format

The following subsections detail the structure of the binary stream.

### 6.1 Logical Format

The format of the EBM streamed data is shown in Figure 9.

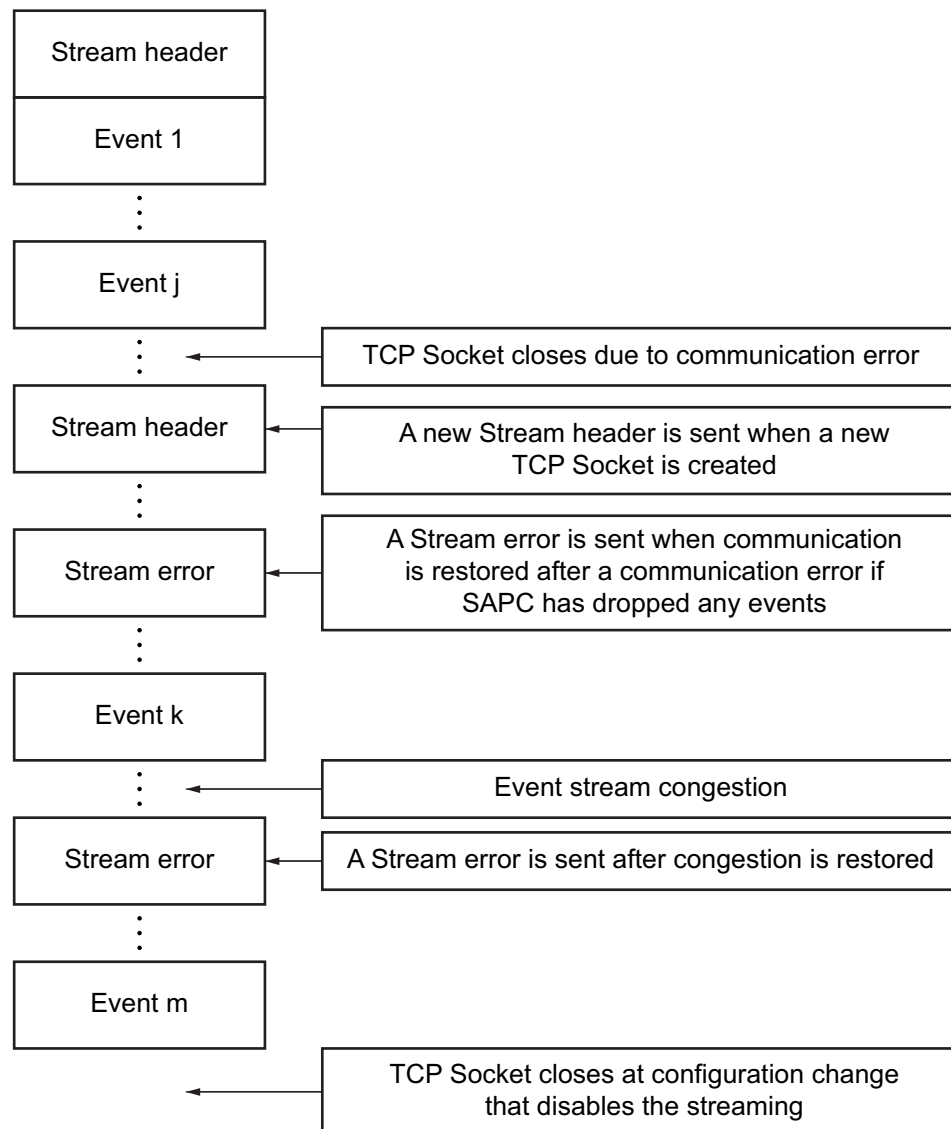


Figure 9 Binary Stream Format

The binary stream is composed of a series of records, which are broken up into shorter segments. Each segment consists of:

- A stream header record
- Conditional stream error records
- A sequence of event records

For descriptions of the stream header record, the stream error record, and the event record, see Section 6.4 on page 13.



## 6.2 Data Format

The EBM records generated by the SAPC are encoded in an Ericsson proprietary bit-packed binary format. The following notations are used in the descriptions:

<b>byte</b>	A sequence of 8 bits. The format of bytes follows the network byte order standard known as big endian.
<b>unsigned char</b>	An integer represented by 8 bits (1 byte). The value range of an unsigned char is 0...255.
<b>unsigned short</b>	An integer represented by 16 bits (2 bytes). The value range of an unsigned short is 0...65535.
<b>binary integer</b>	An integer represented by a number of bits.

## 6.3 Record Padding

The length of each record is always a multiple of 32-bit (4 byte) words. If the last 32-bit word of a record is less than 32 bits long, record padding is performed. Record padding sets all unused bits in the last 32-bit word of the record to 0.

Figure 10 shows an example of record padding.

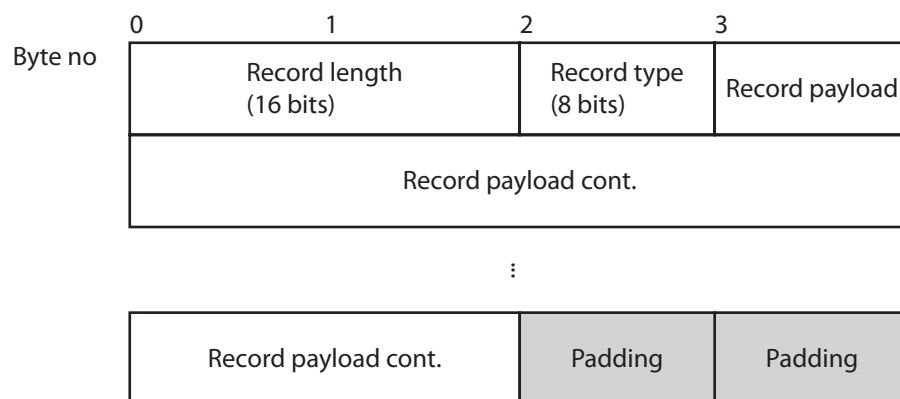


Figure 10 Record Padding



## 6.4 Record Descriptions

The following subsections describe the individual records that are included in the binary stream.

### 6.4.1 Stream Header Record

The stream header record is displayed in the stream to provide administrative information about the streamed data. The date and time indicate when the stream was started or restarted in UTC time.

Table 2 shows the contents of a stream header record.

Table 2 Stream Header Record Contents

Position	Description	Byte	Type	Value / Range
0	Record length	2	Unsigned short	[20...40]
1	Record type	1	Unsigned char	4
2	FFV	1	Unsigned char	[0...255] <sup>(1)</sup>
3	FIV	1	Unsigned char	[0...255] <sup>(2)</sup>
4	Year (4 digits)	2	Unsigned short	[2000...]
5	Month	1	Unsigned char	[1...12]
6	Day	1	Unsigned char	[1...31]
7	UTC hour	1	Unsigned char	[0...23]
8	UTC minute	1	Unsigned char	[0...59]
9	Second	1	Unsigned char	[0...60] <sup>(3)</sup>
10	UTC sign	1	Unsigned char	[0...1] <sup>(4)</sup>
11	UTC hour offset	1	Unsigned char	[0...14]
12	UTC minute offset	1	Unsigned char	[0...59]
13	Cause of header	1	Unsigned char	[0...4] <sup>(5)</sup>
14	Node ID	0-20	Unsigned char	ASCII string <sup>(6)</sup>
15	Padding	0-3	Unsigned char	0

(1) The FFV is incremented if there is a change in the file format.

(2) The FIV is incremented if there is a change in the file information.

(3) The 60 second value indicates a leap second.

(4) The UTC sign with the UTC hour and minute offset indicate the time difference between the local time and the UTC time. The value 0 stands for the sign +, and the value 1 stands for the sign -.

(5) The cause of header indicates the reason why the Stream Header Record was sent. Values: 0 = Stream enabled. 1 = Stream restart after communication failure. 2 = Stream restart after reconfiguration. 3 = Local time is changed. 4 = Other reason.

(6) ASCII characters with variable length.



## 6.4.2 Stream Error Record

The stream error record is only included if an error occurs which might have resulted in some event records being discarded. It contains information about the reason for the error, the number of discarded events, and the time of the error situation recovery.

Table 3 shows the contents of a stream error record.

Table 3 Stream Error Record Contents

Position	Description	Bytes	Type	Value / Range
0	Record length	2	Unsigned char	12
1	Record type	1	Unsigned char	5
2	UTC hour	1	Unsigned char	[0...23]
3	UTC minute	1	Unsigned char	[0...59]
4	UTC second	1	Unsigned char	[0...60] <sup>(1)</sup>
5	Error type	1	Unsigned char	[0...2] <sup>(2)</sup>
6	Number of dropped events	4	Unsigned char	[1...4294967295]
7	Padding	1	Unsigned char	0

(1) The 60 second value indicates a leap second.

(2) Values: 0 = Other. 1 = Communication error. No TCP connections available to the EBM server. 2 = Congestion. The SAPC generates more EBM events than the number of EBM events that it can send to the EBM server. The SAPC sends the number of dropped events in a Stream Error Record when new events can be sent.

Page 15 shows an example of Stream Error Record due to congestion.

Length	Type	Hour	Minute	Second	Error Type	Dropped event #	Padding
000C	05	0E	18	1C	02	0000020C	00

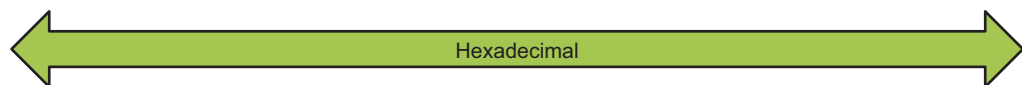


Figure 11 Stream Error Record

## 6.4.3 Event Record

Event records follow stream header records or stream error records in segments.

Event records contain event data, including event identity and event result.

Table 4 shows the contents of an event record.



Table 4 Event Record Contents

Position	Description	Bytes	Type	Value / Range
0	Record length <sup>(1)</sup>	2	Unsigned short	[4..65535]
1	Record type	1	Unsigned char	1
2	Bit-packed part	Variable	Unsigned char	Variable <sup>(2)</sup>
-	Padding	1	Unsigned char	0

(1) Record length requires 2 bytes, even though 1 byte would be sufficient. The maximum size of EBM record is 65535 bytes. If the event record maximum size is surpassed, the SAPC drops the event record.

(2) The bit-packed part of the event records is composed of a sequence of variable length fields. The length of the bit-packed part can be a multiple of 32-bit (4 byte) words. The Most Significant Bit (MSB) is displayed first in each field. If the last 32-bit (4 byte) word of a record is not exactly 32 bits (4 byte) long, padding is performed. Padding is setting all unused bits in the last 32-bit (4 byte) word of the record to 0.

Each event record contains a bit-packed part. Its content is included in Table 5.

Table 5 Event Record Common Contents

Position	Description	Bits	Type	Value / Range
0	Event ID	8	Binary integer	[0...22] <sup>(1)</sup>
1	Event Result	2	Binary integer	[0...3]
2	Hour	5	Binary integer	[0...23]
3	Minute	6	Binary integer	[0...59]
4	Second	6	Binary integer	[0...60] <sup>(2)</sup>
5	Millisecond	10	Binary integer	[0...999]
6	Duration	24	Binary integer	[0...16777215] <sup>(3)</sup>

(1) Event-specific field in bit-packed event records.

(2) Value 60 indicates a leap second.

(3) Duration provides timing information with millisecond granularity for the signaling procedure. The duration value is calculated from the time when the signaling procedure starts and the time when the signaling procedure ends. The signaling procedure starts when the first request is received or sent. The signaling procedure ends when the first accept or reject response is sent or received. If the duration of the event exceeds 16777215 ms, the value of the DURATION parameter starts counting from 0 again.

The common part of event records is followed by event-specific parameters. These event-specific parameters are defined in the XML file.

For further information on the XML file, see Section 5 on page 10.



## 7 Directory Structure

The schema file `Event_Format_description.xsd`, the `sapc_ebm.xml` and the `ebm_cause_codes.xml` files can be copied from the SAPC to an external post processing system using SFTP protocol. The files are stored in the EBM directory on the SAPC, using the following path: `/storage/no-backup/ebm`.