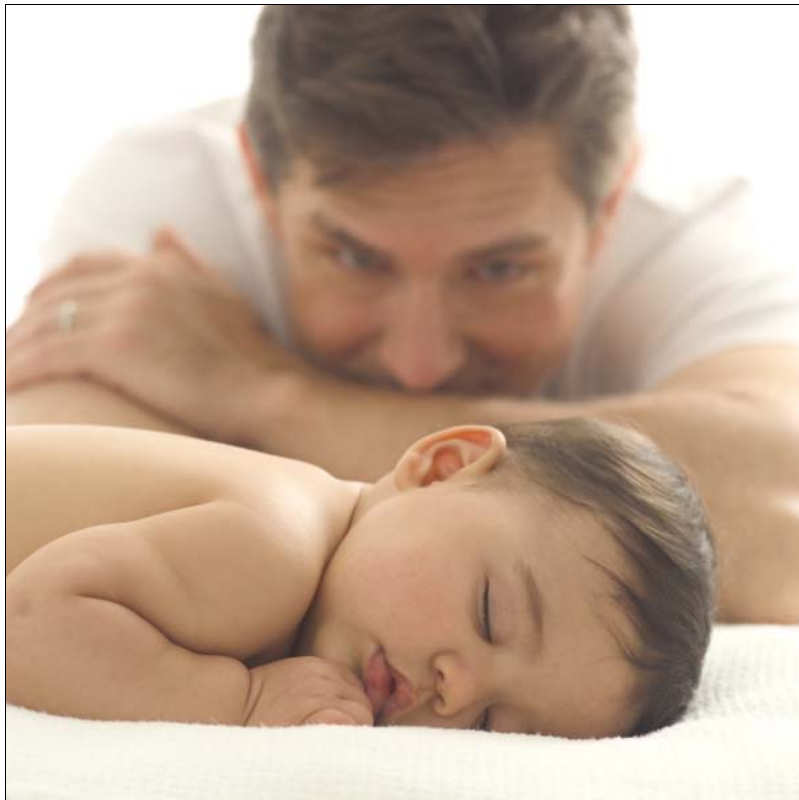


# ESA SNMP Interface for Fault Management

## Ericsson SNMP Agent 16.0

---

### INTERFACE DESCRIPTION



**Copyright**

© Ericsson AB 2004-2016. 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**

Ericsson is the trademark or registered trademark of Ericsson AB. All other products or service names mentioned in this document are trademarks of their respective companies.



# Contents

<b>1</b>	<b>About This Document</b>	<b>1</b>
1.1	Purpose	1
1.2	Target Group	1
1.3	Prerequisites	1
1.4	Typographic Conventions	1
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>Use Cases</b>	<b>5</b>
3.1	Send SNMP Messages	5
3.2	Synchronize Alarms	5
3.3	Alarm Statistics	6
3.4	SNMP Message Log	6
3.5	View Clear History Log	7
3.6	View Event History Log	7
<b>4</b>	<b>Alarm Description</b>	<b>9</b>
4.1	MIB Overview	9
4.2	Trap Definitions	9
4.3	Trap Content	9
<b>5</b>	<b>Alarm Details</b>	<b>13</b>
5.1	Overview	13
5.2	Data: Module Identity	13
5.3	Data: Error Code	14
5.4	Data: Resource Identity	15
5.5	Data: Severity	17
5.6	Data: Model Description	17
5.7	Data: Active Description	17
5.8	Data: Event Type	18
5.9	Data: Probable Cause	18
5.10	Data: Date and Time	18
5.11	Data: Originating Source	19
5.12	Data: Sequence Number	19



<b>6</b>	<b>Alarm Behavior</b>	<b>21</b>
6.1	Overview	21
6.2	Message Type: Alarm	21
6.3	Message Type: Event	21
6.4	Correlating SNMP Messages	22
<b>7</b>	<b>SNMP Message Tables</b>	<b>25</b>
7.1	Overview	25
7.2	Alarm Model, old	26
7.3	Alarm Model, new	26
7.4	Active Alarm List, old	27
7.5	Active Alarm List, new	28
7.6	Alarm Cleared List	28
7.7	Event History List	28
<b>8</b>	<b>Appendix A - Severity</b>	<b>31</b>
<b>9</b>	<b>Appendix B - Event Type</b>	<b>33</b>
<b>10</b>	<b>Appendix C - Probable Cause</b>	<b>35</b>
	<b>Glossary</b>	<b>39</b>
	<b>Reference List</b>	<b>41</b>



# 1 About This Document

## 1.1 Purpose

The purpose of this document is to describe the SNMP interface for Fault Management (FM) that is used in the Ericsson SNMP Agent (ESA). The interface is of interest for the Network Management System (NMS), which is the network entity receiving the alarms from the ESA, but also is using the ESA for alarm synchronization.

## 1.2 Target Group

The target group for this document is personnel working on NMS administration and need to understand the ESA Fault Management interface.

## 1.3 Prerequisites

It is assumed that the user of this document fulfils the following prerequisites.

- Has knowledge about SNMP and how to read and use MIBs.
- Has knowledge about technical English.

## 1.4 Typographic Conventions

The typographic conventions used in this document are described in Reference [1].





## 2 Introduction

The ESA uses a set of MIBs for Fault Management. The MIBs as such are found in the ESA software package and the MIB content, which defines the alarms and the alarm data, can be read. Reading MIBs is however not always a straight forward task. The MIBs does not always clearly indicate the behavior of the alarm flow nor the relation between the data in the alarms.

The Section 3 on page 5 describes the different use cases and scenarios that can be executed on the SNMP interface for Fault Management and Section 4 on page 9 describes the alarm content and data in more detail.







## 3 Use Cases

### 3.1 Send SNMP Messages

The ESA provides the following SNMP message types.

- **Alarm**

Consists of messages “alarm raise” and “alarm clear”.

- **Event**

Consists of message “event”.

The *alarm* message type indicates a fault. A fault is normally persistent until it is solved. This means that the alarm is *stateful*. When a fault occurs an *alarm raise* is triggered. The ESA will maintain its state *active* indicating the fault is still there and not solved yet. This also means that the alarm is maintained in the Active Alarm List (AAL). See Section 3.2 on page 5, which describes the use of AAL and stateful alarms. When the fault is solved an *alarm clear* is triggered. The alarm clear will trigger the ESA to clear the active alarm, which means the active alarm is removed from the AAL. The messages alarm raise and alarm clear are sent to the OSS as two different trap types.

The *event* message type also indicates a fault. The difference compared to the alarm message type is that the event is *stateless*. When an event message is sent it indicates a fault occurred, the event contains a severity and it is sent to the OSS. But, there is no state indicating it is active and thus the ESA does not store it in the AAL. If possible, please try to avoid using this message type. The only reason for using event is when there is no clear trigger found in the system. The FM designer should always try to find the alarm raise trigger as well as the alarm clear trigger for each alarm designed.

### 3.2 Synchronize Alarms

The ESA provides an active alarm list holding all active alarms. This list is visible on the SNMP interface and described in ERICSSON-SNF-ALARM-MIB.

The alarm synchronization is an OSS operation. It means that the OSS reads the alarms from the ESA AAL. This is useful in a number of scenarios, such as

- The OSS or the OSS alarm application restarted.
- Network problems might cause alarms to not reach the OSS.
- The system where the ESA resides or the ESA itself has restarted.
- Problems are found and/or solved in the system where the ESA resides.

In all the scenarios in the list above the OSS reads the ESA AAL in order to get a fresh update of the system status.

### 3.3 Alarm Statistics

The ESA provides FM statistics showing figures of alarms and severities being triggered. The statistics are accessible using the MIB `ITU-ALARM-MIB`.

- Critical Current  
Current number of alarms with severity *critical*.
- Major Current  
Current number of alarms with severity *major*.
- Minor Current  
Current number of alarms with severity *minor*.
- Warning Current  
Current number of alarms with severity *warning*.
- Indeterminate Current  
Current number of alarms with severity *indeterminate*.
- Criticals  
Number of alarms with severity *critical* since ESA start.
- Majors  
Number of alarms with severity *major* since ESA start.
- Minors  
Number of alarms with severity *minor* since ESA start.
- Warnings  
Number of alarms with severity *warning* since ESA start.
- Indeterminates  
Number of alarms with severity *indeterminate* since ESA start.

### 3.4 SNMP Message Log

The ESA supports the `NOTIFICATION-LOG-MIB`, which is based on RFC 3014.



The MIB holds a history log of sent messages, which can be used to see all alarms and events that have been sent from the ESA.

### 3.5 View Clear History Log

The ESA provides a history log of all cleared alarms in the ESA. This history log is visible on the SNMP interface and described in ERICSSON-SNF-ALARM-MIB.

The history log is limited in number of log entries, but the limit is configurable.

### 3.6 View Event History Log

The ESA provides a history log of all events that have been sent by the ESA. This history log is visible on the SNMP interface and described in ERICSSON-SNF-EVENT-MIB.

The history log is limited in number of log entries, but the limit is configurable.





## 4 Alarm Description

### 4.1 MIB Overview

The following MIBs contains the trap definitions.

- ERICSSON-SNF-ALARM-MIB
- ERICSSON-SNF-EVENT-MIB

### 4.2 Trap Definitions

The following trap definition is the *alarm raise* trap.

```
snfAlarmActiveState NOTIFICATION-TYPE  
OBJECTS ...
```

The following trap definition is the *alarm clear* trap.

```
snfAlarmClearState NOTIFICATION-TYPE  
OBJECTS ...
```

The following trap definition is the *event* trap.

```
snfEvent NOTIFICATION-TYPE  
OBJECTS ...
```

### 4.3 Trap Content

The following image describes an example of *alarm raise*.

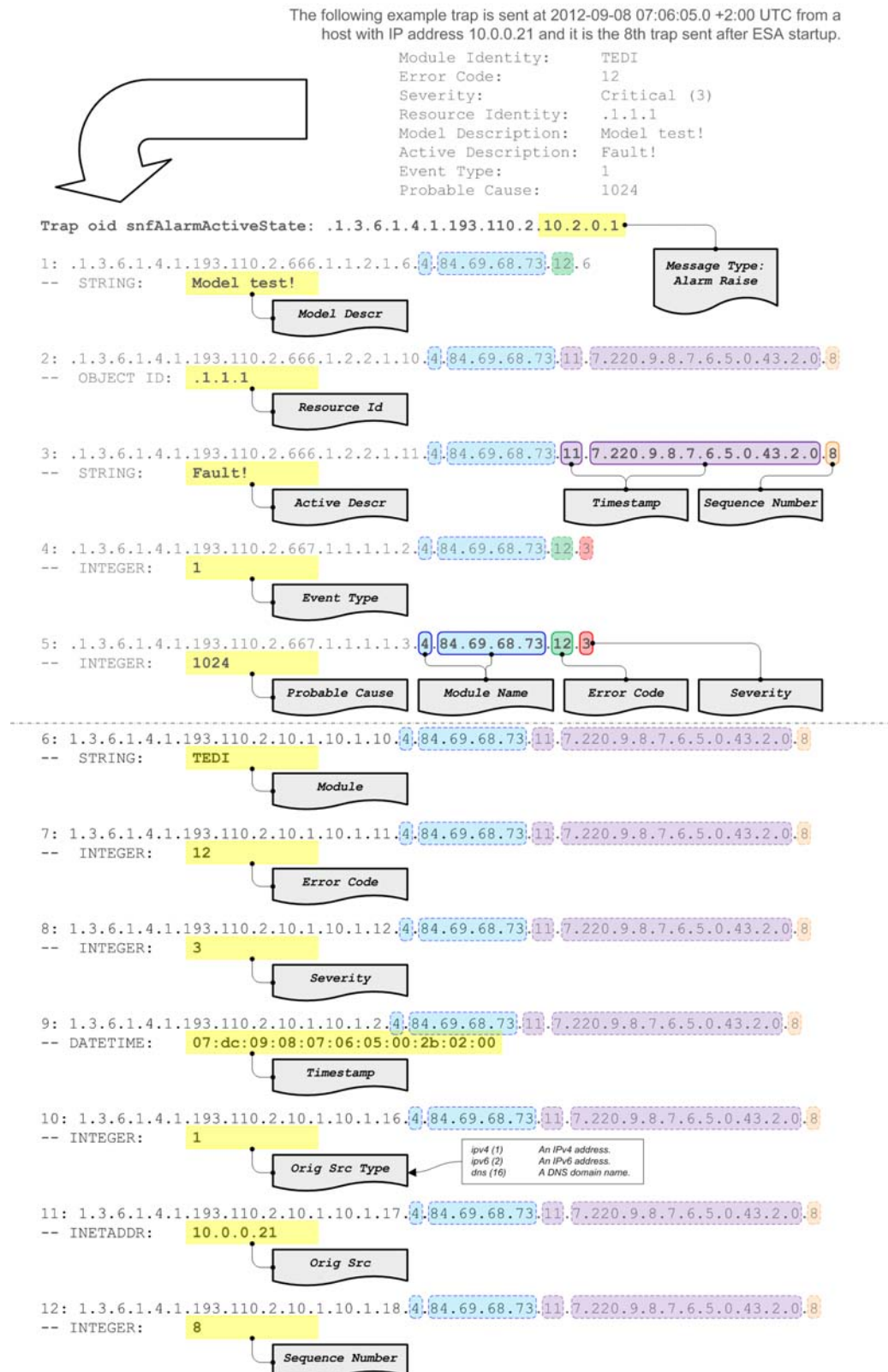


Figure 1 Trap content of an example 'alarm raise' trap.



The image above shows all the FM data that are sent in the SNMP trap *alarm raise* to the OSS. The dashed line between varbinds 5 and 6 in the image indicates what is extended in the updated version of the Ericsson SNF MIBs. As you can see the data in the OID strings within varbinds 1 to 5 are replicated as separate varbinds in 6 to 9 and 12 for easier reading and viewing of trap data. Varbinds 10 and 11 are new.

The structure and content of *alarm clear* is the same as *alarm raise*, but comes as a different trap type. The alarm clear is visualized below.

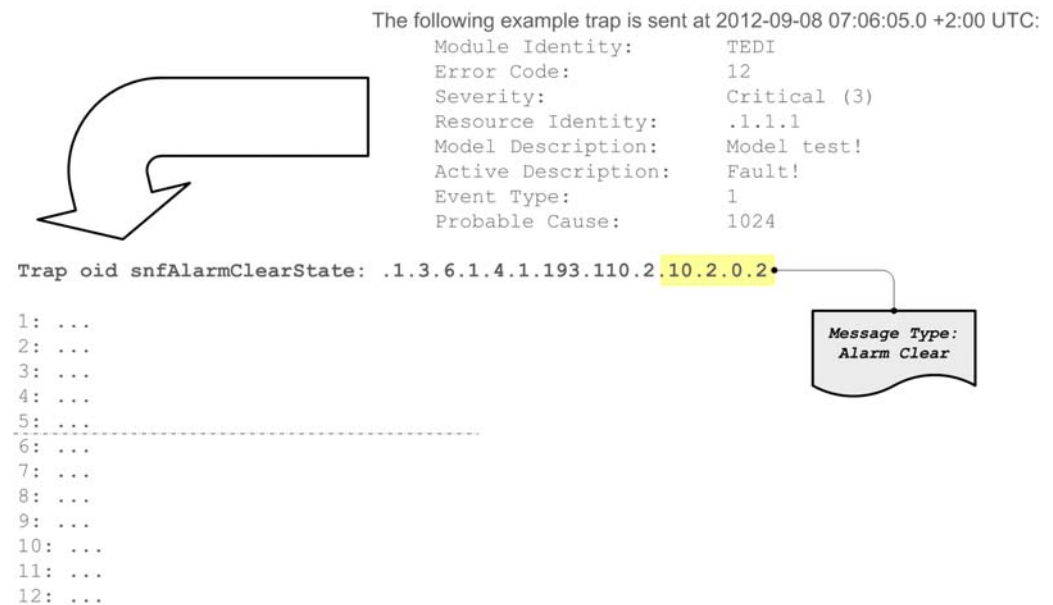


Figure 2 Trap content of an example 'alarm clear' trap.

The image of trap *alarm clear* above simply shows that the trap parameters are the same as in *alarm raise*. The only difference is the trap identity. The trap identity indicates to the OSS that it is a matching trap to an active *alarm raise*. The trap content is used to match the *alarm clear* with an *alarm raise*.

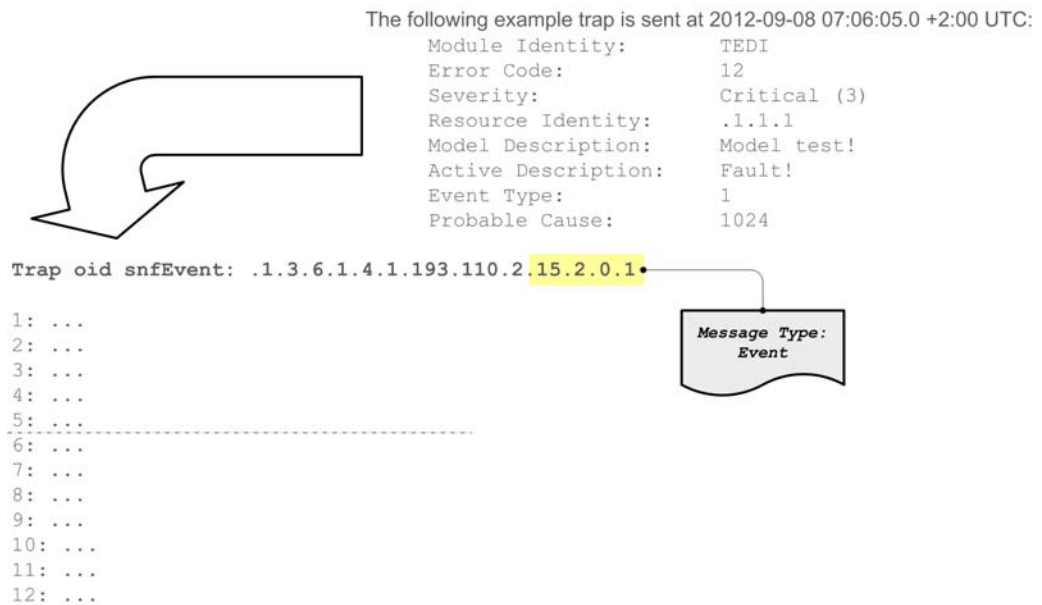


Figure 3 Trap content of an example 'event' trap.

The image of trap *event* above simply shows that the trap parameters are the same as in *alarm raise*. The only difference is the trap identity. The trap identity indicates to the OSS that the trap is stateless.





## 5 Alarm Details

### 5.1 Overview

The trap content is in detail specified in the MIBs. The following information is an overview about how to find the data within the traps. See Figure 1.

The following data is found in the traps *alarm raise*, *alarm clear* and *event*.

- **Module Identity**
- **Error Code**
- **Severity**
- **Model Description**
- **Active Description**
- **Event Type**
- **Probable Cause**
- **Date and Time**
- **Originating Source**
- **Sequence Number**

The following chapters describe the use of the parameter content in more detail.

### 5.2 Data: Module Identity

The alarm parameter *Module Identity* can be seen as an “alarm group”. The module identity identifies a group of alarms for a specific purpose and thus it must be unique within an ESA instance. It is not possible to define two alarm groups with the same module identity.

This parameter is crucial to uniquely identify an alarm within the ESA and in the OSS receiving the alarm. See Section 6.2 on page 21, which describes how it is being used.

A few examples of modules that can be created in the ESA:

- **HARDWARE**  
for hardware alarms.
- **SYSRES**



for system resources alarms.

- DATABASE

for database alarms.

The format is a string holding 1 to 255 characters where character space is not allowed. From a usability aspect the recommendation is to use module identities with length 6-25 characters.

Any number of modules can be defined. Please keep in mind that the naming of the modules shall be easily understood and used by the ESA users. For example, the module identities may also be used in CLI operations. Shorter and more distinct names are thus easier to use.

Often the abbreviation `MODULEID` is used to describe the Module Identity.

The Module Identity is defined in the Alarm Definition configuration XML file and used when triggering an alarm in the ESA.

## 5.3 Data: Error Code

The alarm parameter *Error Code* can be seen as an “alarm identity”. The error code uniquely identifies an alarm within a module. It is not possible to define two alarms with the same error code within a module.

This parameter is crucial to uniquely identify an alarm within the ESA and in the OSS receiving the alarm. See Section 6.2 on page 21, which describes how it is being used.

A few examples of error codes that can be created in the ESA.

- HARDWARE

```
1001 Fan speed low
1002 Fan stopped
2001 Power supply lost
3001 Temperature high
```

- SYSRES

```
51 CPU Load high, warning level
52 CPU Load high, minor level
53 CPU Load high, major level
54 CPU Load high, critical level
61 Memory usage high, warning level
62 Memory usage high, minor level
63 Memory usage high, major level
64 Memory usage high, critical level
```

- DATABASE



```
501001  Number of sessions high
601001  Table space usage high, warning
601002  Table space usage high, critical
701005  Process lost
```

The format is an integer32 value where value 0 is not allowed. From a usability aspect it is recommended to keep the numbers shorter than 6 characters.

Any number of alarms can be defined (in theory  $2^{32}-1$  alarms). Please keep in mind that the error code values of the alarms shall be easily understood and used by the ESA users. For example, the error codes may also be used in CLI operations. Shorter and more distinct error codes are thus easier to use.

Often the abbreviation `ERRORCODE` is used to describe the Error Code.

The Error Code is defined in the Alarm Definition configuration XML file and used when triggering an alarm in the ESA.

## 5.4 Data: Resource Identity

The alarm parameter *Resource Identity* is an object identifier (OID) indicating the alarming object in a system. The system using the ESA should create a small system topology that specifies the different components, features, functions and/or services that are running in the system *and* that can send alarms. The purpose of the model is to give the receiver of an alarm a view to more easily identify the alarming source and root cause. Please note that all OSSes are not capable of showing this topology view, but the topology is anyway useful as it can be presented as a picture in a document.

The model is defined in a XML file. Keep in mind that the model XML file is of interest for the OSSes only, not for the ESA itself and therefore you will not find detailed model descriptions in the ESA documentation. The ESA is however using the resource identities, which are sent as part of the SNMP alarm.

Why do the OSSes want to have the model? Well, they load the model in order to read the resource identities of it. When an alarm from an ESA is received, which holds the resource identity, the received alarm can be mapped to an alarming object. This means that the OSS in a GUI can highlight the alarming object with a flashing color to indicate a new alarm, the severity of the alarm and pinpoint where in the system the alarm actually occurred.

The following is a very simple topology model of an example system:

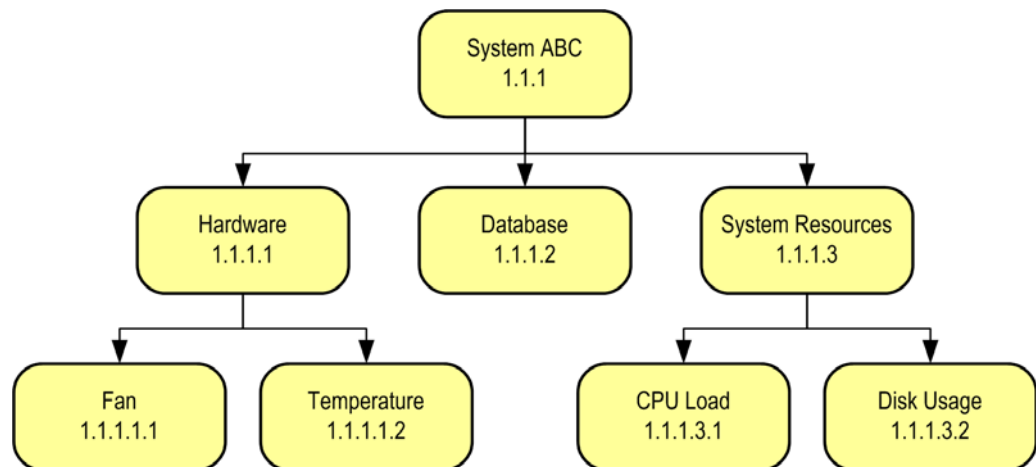


Figure 4 An example system topology model.

For further information about topology design, see Reference [2].

This parameter is crucial to uniquely identify an alarm within the ESA and in the OSS receiving the alarm. The resource id can be used to differentiate one alarm from another.

Often the abbreviation RESOURCEID is used to describe the Resource Identity.

The following is an example that shows how the resource identity can be used to differentiate alarms. The model in Figure 4 is used.

#### EXAMPLE:

The fans in a server are monitored. If any of the fans are stopped an alarm is triggered.

The following fan alarm is defined in the ESA:

```

MODULE      = HARDWARE
ERROR CODE  = 505
  
```

The model shows only one object "Fan" even though there are four fans in the server.

The following could be a fan status table:

Name	Status	Fan Id
FAN1	ACTIVE	1
FAN2	ACTIVE	2
FAN3	INACTIVE	3
FAN4	ACTIVE	4

The table shows that FAN3 has entered an inactive state and an alarm is triggered. The alarm could be sent with the following data including the resource identity.

Alarm Raise with "HARDWARE:505:1.1.1.1.1.3"



Assume that the fan status looks like this.

Name	Status	Fan Id
FAN1	ACTIVE	1
FAN2	INACTIVE	2
FAN3	ACTIVE	3
FAN4	ACTIVE	4

The table shows that FAN2 is now inactive, but FAN3 is back to active. The following alarms would be sent.

```
Alarm Raise with "HARDWARE:505:1.1.1.1.1.2"
Alarm Clear with "HARDWARE:505:1.1.1.1.1.3"
```

As you can see the same alarm is used for different fans. Adding the fan id as part of the resource id makes the alarm unique. It is unique in the ESA AAL, in the OSS Alarm Viewer and alarm raise and alarm clear can be matched correctly.

## 5.5 Data: Severity

The alarm parameter *Severity* is an integer indicating the severity level from Warning to Critical where Warning is the lowest severity and Critical is the highest.

The severity numbers and levels to use are found in Section 8 on page 31.

The Severity is defined in the Alarm Definition configuration XML file. Having it in the configuration file means that it is possible to modify the value according to customer requirements.

## 5.6 Data: Model Description

The alarm parameter *Model Description* is a string to be used as an alarm slogan. This is the “title” of the alarm. It is static and defined in the ESA alarm definition configuration file. It does *not* contain any dynamic or real time data.

The Model Description is defined in the Alarm Definition configuration XML file. Having it in the configuration file means that it is possible to modify the string according to customer requirements.

## 5.7 Data: Active Description

The alarm parameter *Active Description* is a string to be used as real time information. This string is created at trigger time of the alarm, which means it *can* contain dynamic and/or real time data.



The Active Description is defined in the Alarm Definition configuration XML file. The string in the configuration file is only used if there are no dynamic data provided for this parameter at the trigger time of an alarm.

## 5.8 Data: Event Type

The alarm parameter *Event Type* is an integer indicating the type of event of the alarm being triggered.

The event type numbers to be used are found in Section 9 on page 33.

The Event Type is defined in the Alarm Definition configuration XML file.

## 5.9 Data: Probable Cause

The alarm parameter *Probable Cause* is an integer indicating a probable cause to the alarm being triggered.

The probable cause numbers to be used are found in Section 10 on page 35.

The Probable Cause is defined in the Alarm Definition configuration XML file.

## 5.10 Data: Date and Time

The alarm parameter *Date and Time* is in a date and time format and indicates the trigger time for the alarm.

The format of the date and time is according to the MIB SNMPv2-TC.

```
DateAndTime ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "2d-1d-1d,1d:1d:1d.1d,1a1d:1d"
    STATUS current
    DESCRIPTION
        "A date-time specification.
```

field	octets	contents	range
----	-----	-----	-----
1	1-2	year	0..65536
2	3	month	1..12
3	4	day	1..31
4	5	hour	0..23
5	6	minutes	0..59
6	7	seconds	0..60
		(use 60 for leap-second)	
7	8	deci-seconds	0..9
8	9	direction from UTC	'+' / '-'
9	10	hours from UTC	0..11
10	11	minutes from UTC	0..59



For example, Tuesday May 26, 1992 at 1:30:15 PM EDT would be displayed as:

```
1992-5-26,13:30:15.0,-4:0
```

Note that if only local time is known, then timezone information (fields 8-10) is not present."

SYNTAX OCTET STRING

See Figure 1. Please note that the first digit in the OID string that is representing the date and time is a length indicator. It shows the length of the date and time data.

The Date and Time parameter is created at the trigger time of triggering an alarm in the ESA.

## 5.11 Data: Originating Source

The alarm parameter *Originating Source* indicates the IP address of the alarming node in a cluster. This is useful when a single alarm interface is setup for multiple nodes. The OSS will see the IP address of the ESA sending the SNMP trap, but the alarm could be originating from another node in the cluster.

The Originating Source is not defined in configuration files. It is read by the ESA from the trigger of the alarm.

## 5.12 Data: Sequence Number

The alarm parameter *Sequence Number* is an integer indicating the sequence number to each alarm being sent from the ESA.

The OSS can find this useful for a number of purposes, such as:

- The OSS can use it to match an alarm raise with an alarm clear, since the alarm clear will hold the same sequence number as the active alarm raise.
- The alarms are sent from the ESA in a sequence order, but the OSS receives them in another order. The sequence number can support the OSS to sort the alarms properly in order to take proper actions for alarm raise and alarm clear. If an alarm clear is received before an alarm raise, the alarm clear might be thrown away since there is no matching alarm raise.
- The alarms are sent from the ESA in a sequence order, but the OSS does not receive some of the alarms. If the alarms 10, 11, 12, 13, 14 and 15 are sent from the ESA, but the OSS only receives 10, 11, 12, 14 and 15 the OSS can detect that number 13 is lost. When the lost alarm is detected, the OSS can trigger an Alarm Synchronization operation in order to read all active alarms from the ESA AAL.



Please note that alarms and events have separate sequences and thus separate sequence numbers. There is one *alarm sequence number*, which is found in trap type *alarm* and one *event sequence number*, which is found in trap type *event*.





## 6 Alarm Behavior

### 6.1 Overview

The ESA supports the message types *alarm* and *event*. The following sections describe what they are used for and how they relate to each other.

### 6.2 Message Type: Alarm

The message type *alarm* is *stateful*. This means that the alarm maintains an *active* state, which in turn means it represents a persistent fault, until it is cleared.

This is the message type to use for *all* ESA alarms in a system, if possible. Only in rare cases the message type *event* shall be used. For message type *event*, see Section 6.3 on page 21.

When using message type *alarm* there is a benefit using an *Active Alarm List (AAL)*. Each alarm that is sent to the OSS will also end up in the AAL maintaining an active state until the error is solved and the alarm is cleared. The AAL is useful for both technicians and the OSS. The technicians can see which alarms are still active (not solved). The OSS can use the AAL to synchronize the alarms in its own alarm list with the AAL in scenarios where there might be irregularities in the OSS alarm list.

For further information about how the alarm raise and alarm clear are correlated with each other, see Reference [2].

### 6.3 Message Type: Event

The message type *event* is *stateless*. This means that the alarm does not maintain a state. From ESA and system point of view it is a fire-and-forget message.

This message type should be used in rare cases only, which means in the cases where it is impossible or complex to find the clear detection for an alarm.

The event messages are not put into the AAL and thus the technicians nor the OSS can use the AAL to see event messages representing not solved errors.

## 6.4 Correlating SNMP Messages

This chapter is valid for both message type *alarm* and *event*. The SNMP messages contains data that makes it possible to correlate alarms, filter alarms and sort alarms on the OSS side.

Alarms and Events defined in the ESA are all related to Alarm Definitions even though a defined alarm can be sent as an event. The only difference is that the message type alarm is stateful and message type event is stateless. See previous chapters for details about alarm and event. The message format is however the same and thus the messages can be processed with regards to the content.

First of all the alarms defined in the ESA are uniquely defined by an *alarm group* and an *alarm identity*. The alarm group is known as Module Identity (MODULEID) and the alarm identity is known as the Error Code (ERRORCODE) in the ESA. The concatenated form MODULEID:ERRORCODE can not be defined twice in the ESA. Each MODULEID:ERRORCODE is created uniquely for a specific fault in the system.

The next data to be used to distinguish one alarm from another is the *alarming object*. The alarming object is known as the Resource Identity (RESID) in the ESA. This can be used to reuse an alarm definition for multiple faults in the system, but still allow the alarm user to distinguish one alarm from another.

The last data to use as a differentiator in an alarm message is the *originating source*. The originating source (ORIGSRC) is known as the IP address or the Hostname of the node triggering the alarm in the ESA. It can be either local or remote to the ESA handling the trigger. This data can be used to differentiate alarms of the same kind from the same cluster.

Let's have a look at a few examples which describes the alarm content according to `<moduleid>:<errorcode>:<resid>:<origsrc>`.

### Example 1) A single alarm

The hardware agent reports that a CPU is gone in a node. The following alarm is sent.

```
HARDWARE:1001:<resid>:<origsrc>
```

The viewer of the alarm will see the unique alarm group and error code.

### Example 2) Multiple alarming sources

The hardware agent reports that CPU #2 and #5 out of 8 CPUS are gone in a node. The following alarms are sent.

```
HARDWARE:1001:1.1.1.10.2:<origsrc>
HARDWARE:1001:1.1.1.10.5:<origsrc>
```



The viewer of the alarm will see the same alarm indicating a CPU error twice. They can be sorted as "hardware alarm", but are still unique identifying two different problems.

### **Example 3) Multiple alarm triggers**

The hardware agents on two nodes 10.0.0.4 and 10.0.0.8 reports that CPU #3 are gone in both nodes. The following alarms are sent.

```
HARDWARE:1001:1.1.1.10.3:10.0.0.4  
HARDWARE:1001:1.1.1.10.3:10.0.0.8
```

The viewer of the alarm will also in this case see the same alarm indicating a CPU error twice. They can be sorted as "hardware alarm", but are still unique identifying two different problems.





## 7 SNMP Message Tables

### 7.1 Overview

The ESA supports a few SNMP tables with data that can be read by anyone using SNMP.

- Alarm Model, *old*

This SNMP table holds all alarms that are defined in the ESA, which means the reader can see which alarms that *possibly* can be sent from the system.

**Note:** This table is in fact multiple tables that must be aggregated to get all data for a defined alarm. It is supported in the ESA for backward compatibility reasons. It is recommended to use the new Alarm Model table instead.

- Alarm Model, *new*

This SNMP table holds all alarms that are defined in the ESA, which means the reader can see which alarms that *possibly* can be sent from the system. This table is a single table that contains all data for a defined alarm.

- Active Alarm List, *old*

This SNMP table holds all active alarms in the ESA, which means all alarms that represent a not solved fault in the system.

**Note:** This table is in fact multiple tables that must be aggregated to get all data for an active alarm. It is supported in the ESA for backward compatibility reasons. It is recommended to use the new AAL table instead.

- Active Alarm List, *new*

This SNMP table holds all active alarms in the ESA, which means all alarms that represent a not solved fault in the system. This table is a single table that contains all data for an active alarm.

- Alarm Cleared List

This SNMP table holds the last *n* cleared alarms in the ESA, which means all alarms that once represented a fault in the system, but now is solved. This table is a single table that contains all data for a cleared alarm.

- Event History List

This SNMP table holds the last  $n$  events sent from the ESA, which means all stateless alarms that represents or represented a fault in the system. This table is a single table that contains all data for an event.

## 7.2 Alarm Model, old

The initial version of the Alarm Model in the ESA is found in the tables at the locations presented below. The tables marked in bold do combined contain alarm parameters for the defined alarms. The starting point for viewing alarm definitions is table `ituAlarmTable`.

```
private
|
+ enterprises
|
+ ericsson
|
+- snfTopMib
|
+- snfGeneric
|
+- alarmMIB
|
|+- alarmObjects
|
|+- alarmModel
|
|+- alarmModelTable
|
+- ituAlarmMIB
|
+- ituAlarmObjects
|
+- ituAlarmModel
|
+- ituAlarmTable
```

## 7.3 Alarm Model, new

The new version of the Alarm Model in the ESA is found in the table at the location presented below. The table marked in bold contains the alarm parameters for the defined alarms. The only table needed for viewing defined alarms is table `snfModelTable`.

```
private
|
+ enterprises
|
+ ericsson
```



```

|
+- snfTopMib
|
+- snfGeneric
|
+- snfModel
|
+- snfModelObjects
|
+- snfModelTable

```

## 7.4 Active Alarm List, old

The initial version of the Active Alarm List (AAL) in the ESA is found in the tables at the locations presented below. The tables marked in bold do combined contain alarm data for the active alarms. The starting point for viewing active alarms is table `ituAlarmActiveTable`.

```

private
|
+ enterprises
|
+ ericsson
|
+- snfTopMib
|
+- snfGeneric
|
+- alarmMIB
|
|+- alarmObjects
|
|+- alarmModel
|
|+- alarmModelTable
|
|+- alarmActive
|
|+- alarmActiveTable
|
|+- alarmActiveVariableTable
|
+- ituAlarmMIB
|
+- ituAlarmObjects
|
+- ituAlarmActive
|
+- ituAlarmActiveTable

```



## 7.5 Active Alarm List, new

The new version of the Active Alarm List (AAL) in the ESA is found in the table at the location presented below. The table marked in bold contains alarm data for the active alarms. The only table needed for viewing active alarms is table `snfAlarmActiveTable`.

```
private
|
+ enterprises
|
+ ericsson
|
+- snfTopMib
|
+- snfGeneric
|
+- snfAlarmModule
|
+- alarmObjects
|
+- snfAlarmActiveTable
```

## 7.6 Alarm Cleared List

The Alarm Cleared List in the ESA is found in the table at the location presented below. The table marked in bold contains alarm data for the cleared alarms. The table `snfAlarmClearTable` solely contains all data for the cleared alarms.

```
private
|
+ enterprises
|
+ ericsson
|
+- snfTopMib
|
+- snfGeneric
|
+- snfAlarmModule
|
+- alarmObjects
|
+- snfAlarmClearTable
```

## 7.7 Event History List

The Event History List in the ESA is found in the table at the location presented below. The table marked in bold contains event data for the event sent from





the ESA. The table `snfEventHistoryTable` solely contains all data for the sent events.

```
private
|
+ enterprises
|
+ ericsson
|
+- snfTopMib
|
+- snfGeneric
|
+- snfEventModule
|
+- eventObjects
|
+- snfEventHistoryTable
```





## 8 Appendix A - Severity

This section covers the definitions of ITU perceived severity values as per M.3100 and X.733.

The following severity levels are used in the ESA trap types *alarm* and *event*.

1. Cleared (1)
2. Indeterminate (2)
3. Critical (3)
4. Major (4)
5. Minor (5)
6. Warning (6)

Note that *Cleared (1)* is not used in the ESA. When clearing an alarm the trap type alarm clear is sent with the severity of the alarm raise that is to be cleared.

The severity *Indeterminate (2)* is not recommended to be used. It indicates that the severity can not be determined. However, at design time, which means when alarms are defined and created, the severities are specified as well. Thus alarms should not be designed with severity *Intermediate*.





## 9 Appendix B - Event Type

The following Event Types are the ITU event type values as per M.3100. They are also found in the MIB IANA-ITU-ALARM-TC.

1. other (1)
2. communicationsAlarm (2)
3. qualityOfServiceAlarm (3)
4. processingErrorAlarm (4)
5. equipmentAlarm (5)
6. environmentalAlarm (6)
7. integrityViolation (7)
8. operationalViolation (8)
9. physicalViolation (9)
10. securityServiceOrMechanismViolation (10)
11. timeDomainViolation (11)





## 10 Appendix C - Probable Cause

This section covers ITU probable cause values for the alarms as per M.3100, X.733 and X.736. Duplicate values defined in X.733 are appended with \_X733 to ensure uniqueness. Probable causes 1-207 are defined in M.3100. They are also found in the MIB IANA-ITU-ALARM-TC.

The following probable causes are related to communications alarms:

alS (1)	unavailable (14)
callSetUpFailure (2)	signalLabelMismatch (15)
degradedSignal (3)	lossOfMultiFrame (16)
farEndReceiverFailure (4)	receiveFailure (17)
framingError (5)	transmitFailure (18)
lossOfFrame (6)	modulationFailure (19)
lossOfPointer (7)	demodulationFailure (20)
lossOfSignal (8)	broadcastChannelFailure (21)
payloadTypeMismatch (9)	connectionEstablishmentError (22)
transmissionError (10)	invalidMessageReceived (23)
remoteAlarmInterface (11)	localNodeTransmissionError (24)
excessiveBER (12)	remoteNodeTransmissionError (25)
pathTraceMismatch (13)	routingFailure (26)

Values 27-50 are reserved for communications alarm related probable causes. The following are used with equipment alarms:



backplaneFailure (51)	transmitterFailure (67)
dataSetProblem (52)	trunkCardProblem (68)
equipmentIdentifierDuplication (53)	replaceableUnitProblem (69)
externalIFDeviceProblem (54)	realTimeClockFailure (70)
lineCardProblem (55)	antennaFailure (71)
multiplexerProblem (56)	batteryChargingFailure (72)
nEIdentifierDuplication (57)	diskFailure (73)
powerProblem (58)	frequencyHoppingFailure (74)
processorProblem (59)	iODeviceError (75)
protectionPathFailure (60)	lossOfSynchronisation (76)
receiverFailure (61)	lossOfRedundancy (77)
replaceableUnitMissing (62)	powerSupplyFailure (78)
replaceableUnitTypeMismatch (63)	signalQualityEvaluationFailure (79)
synchronizationSourceMismatch (64)	tranceiverFailure (80)
terminalProblem (65)	protectionMechanismFailure (81)
timingProblem (66)	protectingResourceFailure (82)

Values 83-100 are reserved for equipment alarm related probable causes. The following are used with environmental alarms:

airCompressorFailure (101)	explosiveGas (119)
airConditioningFailure (102)	fire (120)
airDryerFailure (103)	flood (121)
batteryDischarging (104)	highHumidity (122)
batteryFailure (105)	highTemperature (123)
commercialPowerFailure (106)	highWind (124)
coolingFanFailure (107)	iceBuildUp (125)
engineFailure (108)	intrusionDetection (126)
fireDetectorFailure (109)	lowFuel (127)
fuseFailure (110)	lowHumidity (128)
generatorFailure (111)	lowCablePressure (129)
lowBatteryThreshold (112)	lowTemperature (130)
pumpFailure (113)	lowWater (131)
rectifierFailure (114)	smoke (132)
rectifierHighVoltage (115)	toxicGas (133)
rectifierLowVoltage (116)	coolingSystemFailure (134)
ventilationsSystemFailure (117)	externalEquipmentFailure (135)
enclosureDoorOpen (118)	externalPointFailure (136)

Values 137-150 are reserved for environmental alarm related probable causes. The following are used with processing error alarms:





storageCapacityProblem (151)	configurationOrCustomisationError (159)
memoryMismatch (152)	databaseInconsistency (160)
corruptData (153)	fileError (161)
outOfCPUCycles (154)	outOfMemory (162)
sfwrEnvironmentProblem (155)	softwareError (163)
sfwrDownloadFailure (156)	timeoutExpired (164)
lossOfRealTimeI (157)	underlyingResourceUnavailable (165)
applicationSubsystemFailure (158)	versionMismatch (166)

Values 168-200 are reserved for processing error alarm related probable causes.

bandwidthReduced (201)	excessiveRetransmissionRate (205)
congestion (202)	reducedLoggingCapability (206)
excessiveErrorRate (203)	systemResourcesOverload (207)
excessiveResponseTime (204)	

The following probable causes are defined in X.733:



adapterError (500)	materialSupplyExhausted (528)
applicationSubsystemFailure (501)	multiplexerProblemX733 (529)
bandwidthReducedX733 (502)	outOfMemoryX733 (530)
callEstablishmentError (503)	outputDeviceError (531)
communicationsProtocolError (504)	performanceDegraded (532)
communicationsSubsystemFailure (505)	powerProblems (533)
configurationOrCustomizationError (506)	pressureUnacceptable (534)
congestionX733 (507)	processorProblems (535)
corruptData (508)	pumpFailureX733 (536)
cpuCyclesLimitExceeded (509)	queueSizeExceeded (537)
dataSetOrModemError (510)	receiveFailureX733 (538)
degradedSignalX733 (511)	receiverFailureX733 (539)
dteDceInterfaceError (512)	remoteNodeTransmissionErrorX733 (540)
enclosureDoorOpenX733 (513)	resourceAtOrNearingCapacity (541)
equipmentMalfunction (514)	responseTimeExcessive (542)
excessiveVibration (515)	retransmissionRateExcessive (543)
fileErrorX733 (516)	softwareErrorX733 (544)
fireDetected (517)	softwareProgramAbnormallyTerminated (545)
framingErrorX733 (518)	softwareProgramError (546)
heatingVentCoolingSystemProblem (519)	storageCapacityProblemX733 (547)
humidityUnacceptable (520)	temperatureUnacceptable (548)
inputOutputDeviceError (521)	thresholdCrossed (549)
inputDeviceError (522)	timingProblemX733 (550)
lanError (523)	toxicLeakDetected (551)
leakDetected (524)	transmitFailureX733 (552)
localNodeTransmissionErrorX733 (525)	transmitterFailure (553)
lossOfFrameX733 (526)	underlyingResourceUnavailable (554)
lossOfSignalX733 (527)	versionMismatchX733 (555)

The following are probable causes defined in X.736:

authenticationFailure (600)	informationOutOfSequence (608)
breachOfConfidentiality (601)	keyExpired (609)
cableTamper (602)	nonRepudiationFailure (610)
delayedInformation (603)	outOfHoursActivity (611)
denialOfService (604)	outOfService (612)
duplicateInformation (605)	proceduralError (613)
informationMissing (606)	unauthorizedAccessAttempt (614)
informationModificationDetected (607)	unexpectedInformation (615)

Other probable causes are covered by other (1024).



# Glossary

## **Glossary**

*ESA Glossary of Terms and Acronyms,*  
0033-CSH 109 532





## Reference List

- [1] *ESA Library Overview*  
*DIRECTIONS FOR USE*, 1/1553-CSH 109 532
- [2] *ESA Fault Management*  
*SYSTEM ADMINISTRATION GUIDE*, 2/1543-CSH 109 532