
Source: SA5 (Telecom Management)
Title: 4 Rel-5 CRs 32111-2,-3 & -4 (Fault Management; Alarm IRP:
Information Service, CORBA & CMIP solution sets)
Document for: Approval
Agenda Item: 7.5.3

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Ver-Current	Doc-2nd-Level	Workitem
SP-030138	32.111-2	025	-	Rel-5	Add Missing security event types and probable causes	F	5.2.0	S5-036076	OAM-NIM
SP-030138	32.111-3	029	-	Rel-5	Correction of CORBA IDL Optional clearSystemId	F	5.2.0	S5-036333	OAM-NIM
SP-030138	32.111-4	017	-	Rel-5	Add missing x721AlarmNotificationsPackage	F	5.3.0	S5-036278	OAM-NIM
SP-030138	32.111-4	018	-	Rel-5	Corrections to GDMO and ASN.1 definitions in the Alarm IRP CMIP SS	F	5.3.0	S5-036332	OAM-NIM

CHANGE REQUEST

⌘ **32.111-2 CR 025** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.

Proposed change affects: UICC apps ME Radio Access Network Core Network

Title:	⌘ Add Missing security event types and probable causes		
Source:	⌘ S5		
Work item code:	⌘ OAM-NIM	Date:	⌘ 28/02/2003
Category:	⌘ F	Release:	⌘ Rel-5
	Use <u>one</u> of the following categories: F (correction) A (corresponds to a correction in an earlier release) B (addition of feature), C (functional modification of feature) D (editorial modification) Detailed explanations of the above categories can be found in 3GPP TR 21.900 .		Use <u>one</u> of the following releases: 2 (GSM Phase 2) R96 (Release 1996) R97 (Release 1997) R98 (Release 1998) R99 (Release 1999) Rel-4 (Release 4) Rel-5 (Release 5) Rel-6 (Release 6)

Reason for change:	⌘ Security event types and probable causes are missing.
Summary of change:	⌘ Add security event types and probable causes.
Consequences if not approved:	⌘ It would be impossible to implement Rel-5 security alarm feature.

Clauses affected:	⌘ Annex A, annex B										
Other specs affected:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;">X</td> </tr> </table>	Y	N		X		X		X	Other core specifications Test specifications O&M Specifications	⌘
Y	N										
	X										
	X										
	X										
Other comments:	⌘										

Change in Annex A

Annex A (normative): Event Types

This appendix lists and explains event types used by this document.

Event type is defined in 3GPP TS 32.302 [5]. The table below lists some of the event types referred to in this document.

Notification IRP: Information Service [5] defines a parameter called `notificationType` that shall be present in all notification. This document defines a parameter called `alarmType` that shall be present in all notifications carrying alarm information. Examples of the `notificationType` are "notification of new alarm", "notification of AlarmList rebuilt", "notification of alarm cleared", etc. Examples of the `alarmType` are the event types defined in table below.

This document also defines an attribute of `AlarmInformation` called `eventType`. The mapping of this `eventType` (internal attribute and not visible to `IRPManager`) to `notificationType` or `alarmType` (both visible to `IRPManager`) is defined in relevant sections of this document. The choice of using "eventType" is to keep the list of attributes of `AlarmList` unchanged (compared to Release 99). One can replace this `eventType` with two attributes, called `notificationType` and `alarmType` so that mapping of these two attributes to the externally visible parameters of the same name will be straight-forward.

It is noted that the `AlarmInformation.eventType` can capture more information than the ITU-T defined event types [2]. One example is "notification of alarm list rebuilt".

It is noted that the mapping of the IS `notificationType` and `alarmType` to CMIP's event type or CORBA `event_name` or other fields are specified in the respective SS documents.

TableA.1: Event Types

Event Types	Explanation
Communications Alarm	An alarm of this type is associated with the procedure and/or process required conveying information from one point to another (ITU-T Recommendation X.733 [2]).
Processing Error Alarm	An alarm of this type is associated with a software or processing fault (ITU-T Recommendation X.733 [2]).
Environmental Alarm	An alarm of this type is associated with a condition related to an enclosure in which the equipment resides (ITU-T Recommendation X.733 [2]).
Quality of Service Alarm	An alarm of this type is associated with degradation in the quality of a service (ITU-T Recommendation X.733 [2]).
Equipment Alarm	An alarm of this type is associated with an equipment fault (ITU-T Recommendation X.733 [2]).
Integrity Violation	An indication that information may have been illegally modified, inserted or deleted.
Operational Violation	An indication that the provision of the requested service was not possible due to the unavailability, malfunction or incorrect invocation of the service.
Physical Violation	An indication that a physical resource has been violated in a way that suggests a security attack.
Security Service or Mechanism Violation	An indication that a security attack has been detected by a security service or mechanism.
Time Domain Violation	An indication that an event has occurred at an unexpected or prohibited time.

End of Change in Annex A

Change in Annex B

Annex B (normative): Probable Causes

This appendix lists probable causes and their corresponding event types.

Sources of these probable causes are ITU-T Recommendation M.3100 [11], ITU-T Recommendation X.721 [3], ITU-T Recommendation X.733 [2], ITU-T Recommendation X.736 [15] and GSM 12.11 [4].

The list may be extended in the future, e.g. with UMTS-specific probable causes.

Table B.1: Probable Causes from ITU-T Recommendation M.3100 [11]

M.3100 Probable cause	Event type
Indeterminate	Unknown
Alarm Indication Signal (AIS)	Communications
Call Setup Failure	Communications
Degraded Signal	Communications
Far End Receiver Failure (FERF)	Communications
Framing Error	Communications
Loss Of Frame (LOF)	Communications
Loss Of Pointer (LOP)	Communications
Loss Of Signal (LOS)	Communications
Payload Type Mismatch	Communications
Transmission Error	Communications
Remote Alarm Interface	Communications
Excessive Bit Error Rate (EBER)	Communications
Path Trace Mismatch	Communications
Unavailable	Communications
Signal Label Mismatch	Communications
Loss Of Multi Frame	Communications
Back Plane Failure	Equipment
Data Set Problem	Equipment
Equipment Identifier Duplication	Equipment
External IF Device Problem	Equipment
Line Card Problem	Equipment
Multiplexer Problem	Equipment
NE Identifier Duplication	Equipment
Power Problem	Equipment
Processor Problem	Equipment
Protection Path Failure	Equipment
Receiver Failure	Equipment
Replaceable Unit Missing	Equipment
Replaceable Unit Type Mismatch	Equipment
Synchronisation Source Mismatch	Equipment
Terminal Problem	Equipment
Timing Problem	Equipment
Transmitter Failure	Equipment
Trunk Card Problem	Equipment
Replaceable Unit Problem	Equipment
Air Compressor Failure	Environmental
Air Conditioning Failure	Environmental
Air Dryer Failure	Environmental
Battery Discharging	Environmental
Battery Failure	Environmental
Commercial Power Failure	Environmental
Cooling Fan Failure	Environmental
Engine Failure	Environmental
Fire Detector Failure	Environmental
Fuse Failure	Environmental
Generator Failure	Environmental
Low Battery Threshold	Environmental
Pump Failure	Environmental
Rectifier Failure	Environmental
Rectifier High Voltage	Environmental
Rectifier Low F Voltage	Environmental
Ventilation System Failure	Environmental
Enclosure Door Open	Environmental
Explosive Gas	Environmental
Fire	Environmental
Flood	Environmental
High Humidity	Environmental
High Temperature	Environmental
High Wind	Environmental
Ice Build Up	Environmental
Intrusion Detection	Environmental

M.3100 Probable cause	Event type
Low Fuel	Environmental
Low Humidity	Environmental
Low Cable Pressure	Environmental
Low Temperature	Environmental
Low Water	Environmental
Smoke	Environmental
Toxic Gas	Environmental
Storage Capacity Problem	Processing error
Memory Mismatch	Processing error
Corrupt Data	Processing error
Out Of CPU Cycles	Processing error
Software Environment Problem	Processing error
Software Download Failure	Processing error

Table B.2: Probable Causes from ITU-T Recommendation X.721 [3] / ITU-T Recommendation X.733 [2] / ITU-T Recommendation X.736 [15]

X.721/X.733/X.736 Probable Cause	Event type
Adapter Error	Equipment
Application Subsystem Failure	Processing error
Authentication Failure	Security Service or Mechanism Violation
Bandwidth Reduction	Quality of service
Breach of Confidentiality	Security Service or Mechanism Violation
Cable Tamper	Physical Violation
Call Establishment Error	Communications
Communication Protocol Error	Communications
Communication Subsystem Failure	Communications
Configuration or Customizing Error	Processing error
Congestion	Quality of service
Corrupt Data	Processing error
CPU Cycles Limit Exceeded	Processing error
Data Set or Modem Error	Equipment
Degraded Signal	Communications
Delayed Information	Time Domain Violation
Denial of Service	Operational Violation
DTE-DCE Interface Error	Communications
Duplicate Information	Integrity Violation
Enclosure Door Open	Environmental
Equipment Malfunction	Equipment
Excessive Vibration	Environmental
File Error	Processing error
Fire Detected	Environmental
Flood Detected	Environmental
Framing Error	Communications
Heating or Ventilation or Cooling System Problem	Environmental
Humidity Unacceptable	Environmental
Information Missing	Integrity Violation
Information Modification detected	Integrity Violation
Information out of Sequence	Integrity Violation
Input/Output Device Error	Equipment
Input Device Error	Equipment
Intrusion Detection	Physical Violation
Key Expired	Time Domain Violation
LAN Error	Communications
Leak Detection	Environmental
Local Node Transmission Error	Communications
Loss of Frame	Communications
Loss of Signal	Communications
Material Supply Exhausted	Environmental
Multiplexer Problem	Equipment
Non-Repudiation Failure	Security Service or Mechanism Violation
Out of Hours Activity	Time Domain Violation
Out of Memory	Processing error
Out of Service	Operational Violation
Output Device Error	Equipment
Performance Degraded	Quality of service
Power Problem	Equipment
Pressure Unacceptable	Environmental
Procedural Error	Operational Violation
Processor Problem	Equipment
Pump Failure	Environmental
Queue Size Exceeded	Quality of service
Receive Failure	Equipment
Receiver Failure	Equipment
Remote Node Transmission Error	Communications
Resource at or Nearing Capacity	Quality of service

X.721/X.733/X.736 Probable Cause	Event type
Response Time Excessive	Quality of service
Re-transmission Rate Excessive	Quality of service
Software Error	Processing error
Software Program Abnormally Terminated	Processing error
Software Program Error	Processing error
Storage Capacity Problem	Processing error
Temperature Unacceptable	Environmental
Threshold Crossed	Quality of service
Timing Problem	Equipment
Toxic Leak Detected	Environmental
Transmit Failure	Equipment
Transmitter Failure	Equipment
Unauthorised Access Attempt	Security Service or Mechanism Violation
Underlying Resource Unavailable	Processing error
Unexpected Information	Integrity Violation
Unspecified Reason	Operational Violation
Unspecified Reason	Physical Violation
Unspecified Reason	Security Service or Mechanism Violation
Version Mismatch	Processing error

Table B.3: Probable Causes from GSM 12.11 [4]

GSM 12.11 Probable Cause	Event Type
A-bis to BTS interface failure	Equipment
A-bis to TRX interface failure	Equipment
Antenna problem	Equipment
Battery breakdown	Equipment
Battery charging fault	Equipment
Clock synchronisation problem	Equipment
Combiner problem	Equipment
Disk problem	Equipment
Equipment failure	Equipment
Excessive receiver temperature	Equipment
Excessive transmitter output power	Equipment
Excessive transmitter temperature	Equipment
Frequency hopping degraded	Equipment
Frequency hopping failure	Equipment
Frequency redefinition failed	Equipment
Line interface failure	Equipment
Link failure	Equipment
Loss of synchronisation	Equipment
Lost redundancy	Equipment
Mains breakdown with battery back-up	Equipment
Mains breakdown without battery back-up	Equipment
Power supply failure	Equipment
Receiver antenna fault	Equipment
Receiver Failure	Equipment
Receiver multicoupler failure	Equipment
Reduced transmitter output power	Equipment
Signal quality evaluation fault	Equipment
Timeslot hardware failure	Equipment
Transceiver problem	Equipment
Transcoder problem	Equipment
Transcoder or rate adapter problem	Equipment
Transmitter antenna failure	Equipment
Transmitter antenna not adjusted	Equipment
Transmitter failure	Equipment
Transmitter low voltage or current	Equipment
Transmitter off frequency	Equipment
Database inconsistency	Processing error
File system call unsuccessful	Processing error
Input parameter out of range	Processing error
Invalid parameter	Processing error
Invalid pointer	Processing error
Message not expected	Processing error
Message not initialised	Processing error
Message out of sequence	Processing error
System call unsuccessful	Processing error
Timeout expired	Processing error
Variable out of range	Processing error
Watch dog timer expired	Processing error
Cooling system failure	Environmental
External equipment failure	Environmental
External power supply failure	Environmental
External transmission device failure	Environmental
Fan failure	Environmental
High humidity	Environmental
High temperature	Environmental
Intrusion detected	Environmental
Low humidity	Environmental
Low temperature	Environmental
Smoke detected	Environmental
Excessive Error Rate	Quality of service
Reduced alarm reporting	Quality of service
Reduced event reporting	Quality of service

GSM 12.11 Probable Cause	Event Type
Reduced logging capability	Quality of service
System resources overload	Quality of service
Broadcast channel failure	Communications
Connection establishment error	Communications
Invalid message received	Communications
Invalid MSU received	Communications
LAPD link protocol failure	Communications
Local alarm indication	Communications
Remote alarm indication	Communications
Routing failure	Communications
SS7 protocol failure	Communications
Transmission error	Communications

~~Table 20~~ [This table](#) identifies probable causes that are defined by more than one standard. This is for information only.

Table B.4: Duplicated Probable Causes

Duplicated Probable Cause	GSM 12.11	X.721 X.733	X.736	M.3100	Event Type
Call Establishment Failure (X.721/X.733) Call Setup Failure (M.3100)		X		X	Communications
Degraded Signal		X		X	Communications
Framing Error		X		X	Communications
Loss of Frame		X		X	Communications
Loss of Signal		X		X	Communications
Equipment Failure (GSM 12.11) Equipment Malfunction (X.721/X.733)	X	X			Equipment
Multiplexer Problem		X		X	Equipment
Power Problem		X		X	Equipment
Processor Problem		X		X	Equipment
Receiver Failure	X	X		X	Equipment
Timing Problem		X		X	Equipment
Transmitter Failure	X	X		X	Equipment
Enclosure Door Open		X		X	Environmental
Fan Failure (GSM 12.11) Cooling Fan Failure (M.3100)	X			X	Environmental
Fire Detected (X.721/X.733) Fire (M.3100)		X		X	Environmental
Flood Detected (X.721/X.733) Flood (M.3100)		X		X	Environmental
High Humidity	X			X	Environmental
High Temperature	X			X	Environmental
Intrusion Detected (GSM 12.11) Intrusion Detection (X.736/M.3100)	X		X	X	Environmental (GSM 12.11): Physical Violation (X.736/M.3100)
Low Humidity	X			X	Environmental
Low Temperature	X			X	Environmental
Pump Failure		X		X	Environmental
Smoke Detected (GSM 12.11) Smoke (M.3100)	X			X	Environmental
Storage Capacity Problem		X		X	Processing Error
Excessive Bit Error Rate (M.3100) Excessive Error Rate (GSM 12.11)	X			X	
Corrupt Data		X		X	Processing Error

**End of Change in Annex B
End of Document**

**3GPP TSG-SA5 (Telecom Management)
Meeting #33, Phoenix, USA, 24-28 February 2003**

S5-036332

CR-Form-v7

CHANGE REQUEST

⌘ **32.111-4 CR 018** ⌘ rev **-** ⌘ Current version: **5.3.0** ⌘

For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.

Proposed change affects: UICC apps ME Radio Access Network Core Network

Title:	⌘ Corrections to GDMO and ASN.1 definitions in the Alarm IRP CMIP SS	
Source:	⌘ S5	
Work item code:	⌘ OAM-NIM	Date: ⌘ 28/02/2003
Category:	⌘ F Use <u>one</u> of the following categories: F (correction) A (corresponds to a correction in an earlier release) B (addition of feature), C (functional modification of feature) D (editorial modification) Detailed explanations of the above categories can be found in 3GPP TR 21.900 .	Release: ⌘ Rel-5 Use <u>one</u> of the following releases: 2 (GSM Phase 2) R96 (Release 1996) R97 (Release 1997) R98 (Release 1998) R99 (Release 1999) Rel-4 (Release 4) Rel-5 (Release 5) Rel-6 (Release 6)

Reason for change:	⌘ The GDMO and ASN.1 definitions contain some minor errors, e.g. missing notifications in package definitions, inconsistencies between the parameter names in the mapping tables and the corresponding ASN.1 definition, syntax errors.
Summary of change:	⌘ This CR corrects the errors in the GDMO and ASN.1 definitions and the syntax errors.
Consequences if not approved:	⌘ It is not possible to implement the Alarm IRP CMIP SS.

Clauses affected:	⌘ 4, 5									
Other specs affected:	<table border="1"> <tr> <td>Y</td> <td>N</td> </tr> <tr> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td><input type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> </tr> </table> Other core specifications Test specifications O&M Specifications	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	⌘
Y	N									
<input type="checkbox"/>	<input checked="" type="checkbox"/>									
<input type="checkbox"/>	<input checked="" type="checkbox"/>									
<input type="checkbox"/>	<input checked="" type="checkbox"/>									
Other comments:	⌘									

Change in Clause 4 and 5

4 Basic aspects

The present document provides all the GDMO and ASN.1 definitions necessary to implement the Alarm IRP Information Service (3GPP TS 32.111-2 [9]) for the CMIP interface.

4.1 Architectural aspects

The Alarm IRP Information Service description is based on Information Object Classes (IOC), Relationships among IOC and Interfaces (used or implemented by IOC) which include Operations and/or Notifications.

In the present document, for the CMIP interfaces the IOC are modelled as GDMO "Managed Object Classes" (MOC) defined specifically for alarm management, the Operations are modelled as GDMO "Actions" of a MOC while the Notifications are modelled as GDMO "Notifications" included in MOCs that need to report events to the Manager. In more detail, the Notifications related to alarm management are included in a MOC defined in the present document while the Notifications defined for alarm reporting are not included in any MOC defined in the present document. They will be included in other MOCs defined in other CMIP Solution Set or in other CMIP Information Models.

Regarding the Notifications, the present document is based on the Notification IRP CMIP Solution Set (3GPP TS 32.304 [10]).

4.1.1 Reporting new alarms

In case of an alarm occurrence the Agent notifies all subscribed Managers that a new alarm has occurred and has been added into the alarm list of the Agent.

For this purpose the standardised alarm notifications defined in ITU-T Recommendations X.721 [4] and X.733 [5] are used.

4.1.2 Reporting changed alarms

Although in the Alarm IRP Information Service (3GPP TS 32.111-2 [9]) there is a notification specifically defined to report the event of alarm attribute changes, on the CMIP interfaces such events are reported according to ITU-T Recommendations X.721 [4] and X.733 [5], i.e. the original alarm is first cleared (by means of a clear alarm notification) and then a new alarm notification with the changed parameter values is generated by the Agent.

4.1.3 Reporting cleared alarms

On the CMIP interfaces the clearing of alarms is reported by the Agent to the Managers in accordance with the mechanisms defined in ITU-T Recommendation X.733 [5] and ITU-T Recommendation Q.821 [7].

4.1.4 Acknowledgment of alarms

This clause relates to the co-operative alarm acknowledgment managed on Itf-N, which implies that the acknowledgment of alarms can be done on both NM and EM.

The acknowledgment of alarms is managed by means of the MOC `alarmControl`, which includes:

- One Action to acknowledge alarms;
- One Action to un-acknowledge alarms;
- ITU-T Recommendation X.721 [4] compliant Alarm Notification to inform Managers about changes of acknowledgment state.

In case an alarm is acknowledged by an operator or automatically by a management system, the `ackUserId`, `ackSystemId`, `ackState` and `ackTime` information is stored in the *additionalInformation* field of the alarm present in the alarm list.

4.1.5 Management of comments associated to alarms

This feature provides the Operators with the capability to add comments to an alarm and to share such information among all the OS (EM and NM) that are involved in the network management. An OS shall have the capability to record more than one comment for each alarm.

The management of the comments associated to alarms is similar to the management of the acknowledgment of alarms and is achieved by means of the same MOC `alarmControl`. For the management of the comments, the MOC `alarmControl` includes one Action to set the comment and Notification to distribute the comments to other OS.

4.1.6 Alignment of alarm conditions over the Itf-N

The IRP Manager is able to trigger the alarm conditions alignment using the Action `getAlarmList`

The following specifies the logical steps of the alignment procedure, by describing a possible implementation. Any other implementation showing the same behaviour on the Itf-N interface is compliant with the present document.

- The Manager sends to the Agent a `getAlarmList` request containing the following information:
 - `alarmAckState`, used to select the alarms from the Agent's alarm list for the current alignment (e.g. all active alarms).
 - `destination`, identifying the destination to which event reports that have passed the filter conditions are sent.
 - `filter`, this optional parameter defines the conditions an alarm notification shall fulfil in order to be forwarded to the Manager. It applies only for the current alignment request.
- After evaluation of the request, the Agent first generates an `alignmentId` value, which unambiguously identifies this alignment process. This value is used by the Manager to correlate alarm reports to the corresponding alignment requests, in case this Manager issues several alarm alignments in parallel.
- The Agent creates a temporary Event Forwarding Discriminator (EFD) instance for the purpose of this alarm alignment, using the parameters `destination` and `filter` received in the request. If the `filter` parameter is absent ~~in the alarm synchronisation request or NULL~~, all alarm notifications are forwarded to the Manager through this EFD, taking into account both the filter constraint currently active for the event reporting to the manager having invoked the synchronisation request and according to the value of the parameter `alarmAckState`. The filter is set by the Agent automatically in order to forward ~~to~~ only those alarm notifications containing, at the beginning of the field `additionalText`, either the string "(ALIGNMENT-<alignmentId>)" or the string "(ALIGNMENTEND-<alignmentId>)".
- The Agent sends back a `getAlarmList` response, which contains the `alignmentId` described above and the `status` information, indicating the result of the request. (see the message flow in Figure 1).
- The Agent scans now its alarm list. For every alarm, which matches the criteria defined by the `alarmAckState` parameter, the Agent inserts, at the beginning of the field `additionalText`, the string "(ALIGNMENT-<alignmentId>)". According to ITU-T Recommendation X.734 [6], the Agent forwards these alarm notifications towards all EFDs.
In the last alarm of the list the Agent inserts the string "(ALIGNMENTEND-<alignmentId>)" to indicate the end of the alarm alignment.

NOTE: These alarm notifications can reach the current Manager only via the temporary EFD created for the current alignment. They are filtered out:

~~NOTE:—These alarm notifications can reach the current Manager only via the temporary EFD created for the current alignment. They are filtered out:~~

- a) By all the EFD instances used for "real-time" alarm reporting, due to the presence of the sub-string "ALIGNMENT" in the field `additionalText` (see 3GPP TS 32.304 [10]).
 - b) By all temporary EFD instances possibly created for parallel alignments, due to the presence of the unambiguous sub-string "<alignmentId>" in the `additionalText` field.
- After sending the last alarm report (identified by the sub-string "ALIGNMENTEND" in the `additionalText`), the Agent automatically deletes the temporary EFD instance (see figure 1).

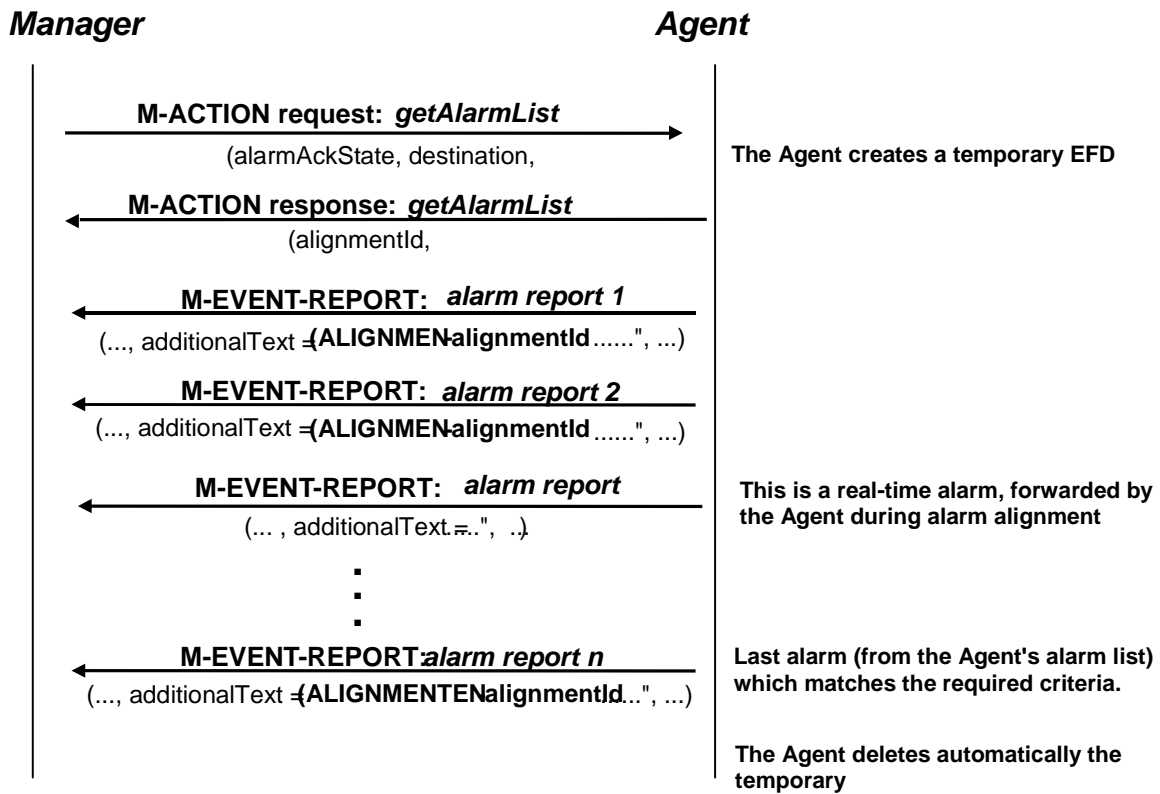


Figure 1: Alignment arrow diagram

Figure 2 shows the handling of a "real-time" alarm notification (occurred during the execution of the *getAlarmList* operation), which is forwarded by the Agent (according to ITU-T Recommendation X.734 [6]) to all currently available EFD instances. Dependent on the *discriminatorConstruct* setting of every EFD, such an alarm may or may not reach the related Manager. In any case, this alarm is filtered out by the temporary EFD assigned to the Manager, which triggered the *getAlarmList* request.

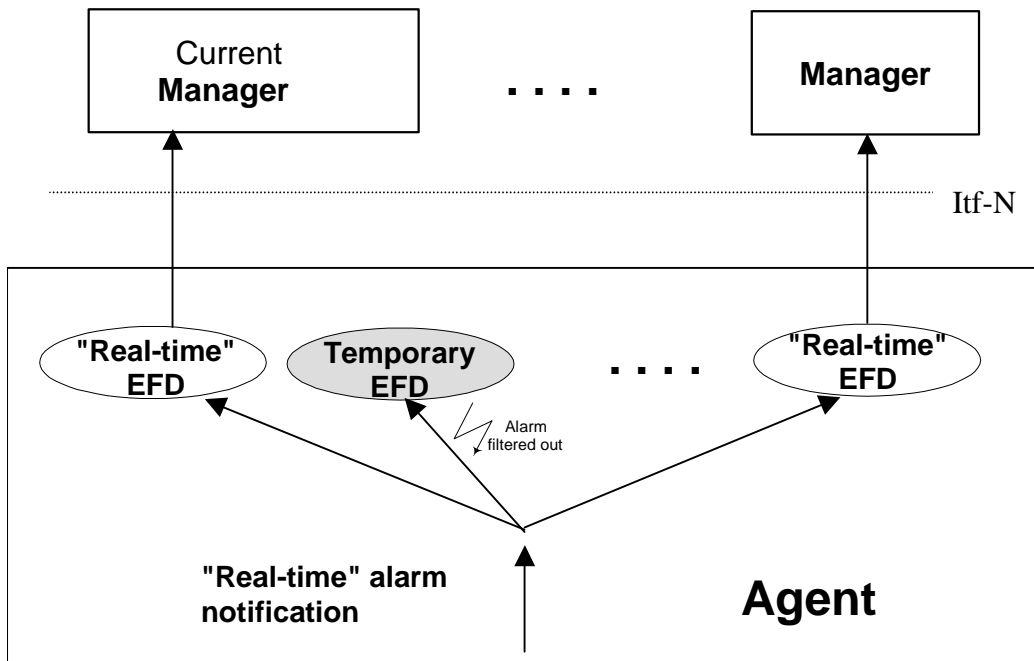


Figure 2: Treatment of "real time" alarms

Figure 3 shows the handling of an alarm notification from the alarm list, matching the criteria defined in the parameters *alarmAckState* of the *getAlarmList* request and forwarded by the Agent to all EFD instances as well. This alarm is filtered out by all EFD instances in charge of discrimination of "real-time" alarms and can reach only the Manager, which triggered the *getAlarmList* request, because it passes the temporary EFD instance assigned to this Manager.

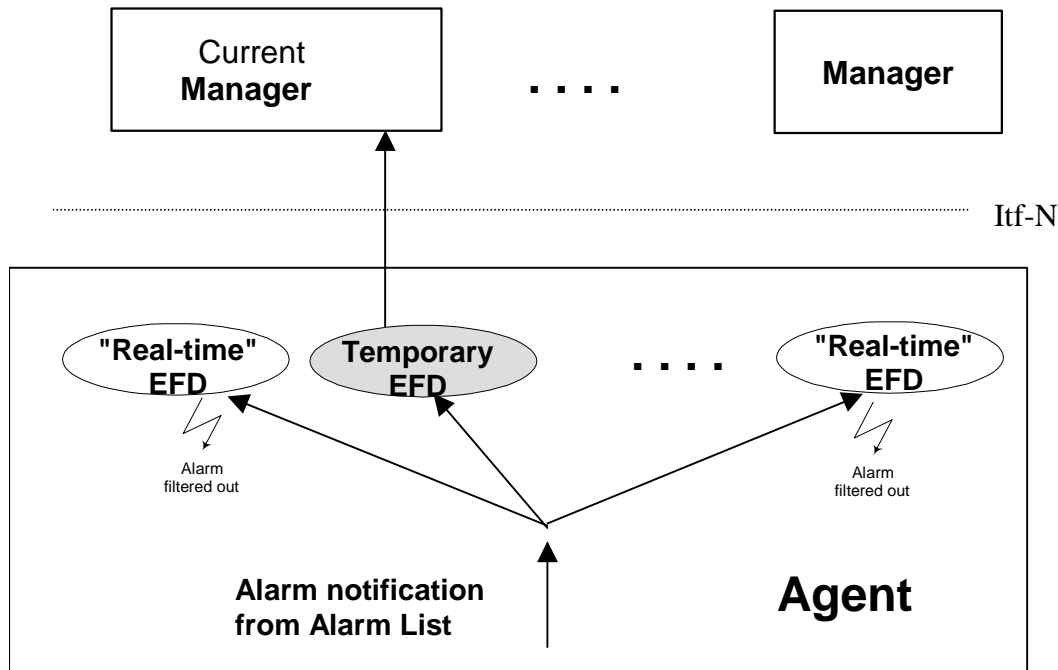


Figure 3: Treatment of "alignment" alarms

4.2 Mapping

The semantics of the Alarm IRP is defined in 3GPP TS 32.111-2 [9]. The definitions of the management information defined there are independent of any implementation technology and protocol. This clause maps these protocol-independent definitions onto the equivalences of the CMIP solution set of Alarm IRP.

4.2.1 Mapping of Information Object Classes and ~~Interfaces~~

For this Alarm IRP CMIP Solution Sets, the Information Object Classes (IOC) and the Interfaces defined in 3GPP TS 32.111-2 [9] are mapped to a Managed Object Classes (MOC) named `alarmControl` which includes all the Attributes, Actions and Notifications necessary to model the management described in (3GPP TS 32.111-2 [9]).

4.2.2 Mapping of ~~Interface~~/Operations

Table 1 maps the Interface/Operations defined in the IS of the Alarm IRP to their equivalents in the CMIP SS. The equivalents are qualified as Mandatory (M) or Optional (O).

Table 1: Mapping of Operations

Interface/Operations of the Alarm IRP Information Services	GDMO Actions of CMIP Solution Set	Qualifier
AlarmIRPOperations_1/acknowledgeAlarms	acknowledgeAlarms	M
AlarmIRPOperations_1/getAlarmList	getAlarmList	M
AlarmIRPOperations_2/getAlarmCount	getAlarmCount	O
AlarmIRPOperations_3/unacknowledgeAlarms	unacknowledgeAlarms	O
AlarmIRPOperations_4/setComment	setComment	O
AlarmIRPOperations_5/clearAlarms	clearAlarms	O
GenericIRPVersionOperation/getIRPVersion	getAlarmIRPVersion	M
GenericIRPProfileOperation/getNotificationProfile	getAlarmIRPNotificationProfile	O
GenericIRPProfileOperation/getOperationProfile	getAlarmIRPOperationProfile	O

NOTE: the Interfaces GenericIRPVersionOperation and GenericIRPPProfileOperation are defined in 3GPP TS 32.312 [11].

4.2.3 Mapping of Operation Parameters ~~of each operation~~

The tables in the following clauses show the parameters of each operations defined in the IS 3GPP TS 32.111-2 [9] and their equivalents in this CMIP SS.

The input parameters of the operations are mapped into "Action information" (see GDMO and ASN.1 definitions for more details).

The output parameters of the operations are mapped into "Action response" (see GDMO and ASN.1 definitions for more details).

Table 2: Parameter mapping of the operation ~~Mapping of parameters of 'acknowledgeAlarms'~~

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationAndSeverityReferenceList	IN	alarmReferenceList (note)	M
alarmInformationAndSeverityReferenceList	IN	AlarmReferenceList (note)	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M
NOTE: severity verification not required in CMIP solution set.			

Table 3: Parameter mapping of the operation ~~Mapping of Parameters of 'getAlarmCount'~~

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
criticalCount	OUT	criticalCount	M
majorCount	OUT	majorCount	M
minorCount	OUT	minorCount	M
warningCount	OUT	warningCount	M
indeterminateCount	OUT	indeterminateCount	M
clearedCount	OUT	clearedCount	M
status	OUT	status	M

Table 4: Parameter mapping of the operation ~~Mapping of Parameters of 'getAlarmList'~~

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
--		destination (input) - see note 1	M
alarmInformationList	OUT	(sequence of alarm notifications) (see subclause 4.5)	M
status	OUT	status	M
--		alignmentId (output) - see note 2	M
NOTE 1: Destination is a CMIP specific parameter and is determined by the Manager.			
NOTE 2: AlignmentId is a CMIP specific parameter and is determined by the Agent.			

Table 5: Parameter mapping of the operation ~~Mapping of Parameters of 'getAlarmIRPVersion'~~

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
versionNumberSet	OUT	versionNumberList	M
status	OUT	status	M

Table 6: Parameter mapping of the operation Mapping of Parameters of 'getOperationProfile'

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
irpVersion	IN	irpVersionNumber	M
operationNameProfile	OUT	operationNameProfile	M
operationParameterProfile	OUT	operationParameterProfile	M
status	OUT	status	M

Table 7: Parameter mapping of the operation Mapping of Parameters of 'getNotificatioProfile'

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
irpVersion	IN	irpVersionNumber	M
notificationNameProfile	OUT	notificationNameProfile	M
notificationParameterProfile	OUT	notificationParameterProfile	M
status	OUT	status	M

Table 8: Parameter mapping of the operation Mapping of Parameters of 'setComment'

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
commentUserId	IN	commentUserId	M
commentSystemId	IN	commentSystemId	O
commentText	IN	commentText	M
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

Table 9: Parameter mapping of the operation Mapping of Parameters of 'unacknowledgeAlarms'

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

Table 10: Parameter mapping of the operation Mapping of Parameters of 'clearAlarms'

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationReferenceList	IN	M-ACTION parameter 'Action information': alarmReferenceList	M
clearUserId	IN	M-ACTION parameter 'Action information': clearUserId	M
clearSystemId	IN	M-ACTION parameter 'Action information': clearSystemId	O
badAlarmInformationReferenceList	OUT	M-ACTION parameter 'Action reply': errorAlarmReferenceList	M
status	OUT	M-ACTION parameter 'Action reply': status	M

4.2.4 Mapping of Notifications

Table 10 maps the Notifications defined in the Information Service of the Alarm IRP to the equivalent Notifications of the CMIP solution set for the Alarm IRP. The CMIP Notifications are qualified as Mandatory (M) or Optional (O).

Table 11: Mapping of Notifications

Notifications of Information Services of the Alarm IRP	Equivalent Notifications of the CMIP solution set for the Alarm IRP	Qualifier
notifyNewAlarm	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationsAlarm ITU-T X.721 [4]	M
notifyChangedAlarm	notifyClearedAlarm notifyNewAlarm which are in turn mapped into environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationsAlarm ITU-T X.721 [4]	O
notifyClearedAlarm	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationsAlarm ITU-T X.721 [4]	M
notifyAckStateChanged	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationsAlarm ITU-T X.721 [4]	M
notifyAlarmListRebuilt	notifyAlarmListRebuilt	M
notifyComments	notifyComments	O
notifyPotentialFaultyAlarmList	notifyPotentialFaultyAlarmList	O

4.2.5 Mapping of [Notification](#) Parameters ~~of each notification~~

In the CMIP Solution Set, all the notifications originated within the Agent are reported to the Managers by means of the CMISE "M-EVENT-REPORT" primitive, which is implemented by means of the "m-EventReport OPERATION" (see ITU-T Recommendations X.710 [2] and X.711 [3]). The argument of m-EventReport OPERATION is defined in ITU-T Recommendation X.711 [3] as follows:

```
EventReportArgument ::= SEQUENCE {
    managedObjectClass      ObjectClass,
    managedObjectInstance  ObjectInstance,
    eventTime               [5] IMPLICIT GeneralizedTime OPTIONAL,
    eventType               EventTypeId,
    eventInfo               [8] ANY DEFINED BY eventType OPTIONAL
}
```

where *eventInfo* is further specified, for each specific notification, by means of specific GDMO/ASN.1 definitions.

In the following tables, for the notifications defined in [9], all parameters are mapped to their CMIP SS equivalents. Note that the parameter mapping for the notification *notifyChangedAlarm* is not given. This is because in the CMIP SS the notifications *notifyClearedAlarm* and *notifyNewAlarm* are emitted instead of the notification *notifyChangedAlarm*.

The IS parameter *systemDN* defined in [9] (Alarm IRP: Information Services) is conditional and not used in the CMIP SS.

Except for the notification *notifyComments* the IS parameter *alarmType* has no direct CMIP SS equivalent. Instead the value of this parameter is reflected by the type of the emitted notification. More specifically:

- If the ~~event~~-[alarm](#) type is equal to 'Communications Alarm' the notification *communicationsAlarm* is emitted;

- If the **event-alarm** type is equal to 'Processing Error Alarm' the notification *processingErrorAlarm* is emitted;
- If the **event-alarm** type is equal to 'Environmental Alarm' the notification *environmentalAlarm* is emitted;
- If the **event-alarm** type is equal to 'Quality of Service Alarm' the notification *qualityofServiceAlarm* is emitted;
- If the **event-alarm** type is equal to 'Equipment Alarm' the notification *equipmentAlarm* is emitted.

Also the IS parameter *alarmId* is not mapped directly to a parameter in the CMIP SS. This is not required because an alarm is identified unambiguously by the notification identifier of the notification reporting the alarm the first time and, if the notification identifier is not unique across the IRP Agent, by the instance of the managed object emitting this notification. Notifications referring to an alarm already reported (e.g. *notifyClearedAlarm*, *notifyAckStateChanged*, *notifyComments*) do so by specifying in the M-EVENT REPORT parameter 'Event information': *correlatedNotifications* (ITU-T Recommendations X.721 [4] and X.733 [5]) the notification identifier of the notification having reported the new alarm and, if required, the instance of the object having emitted this notification.

Most parameters are mapped to the M-EVENT report parameter 'Event information'. For the notifications *notifyNewAlarm* (when reporting alarms not related to security), *notifyClearedAlarm* and *notifyAckStateChanged* the syntax and semantics of this structured parameter are defined in ITU-T X.721 [4] by the ASN.1 definition *AlarmInfo*. In case *notifyNewAlarm* reports a security alarm, the 'Event information' parameter is described by *SecurityAlarmInfo*, defined in ITU-T X.721 [4] as well. For the other notifications (*notifyAlarmListRebuilt*, *notifyComments*, *notifyPotentialFaultyAlarmList*) the 'Event information' parameter is described by ASN.1 definitions defined in this document.

Table 12: Parameter mapping of the notification *notifyNewAlarm* for alarms not related to security

IS Parameter	CMIP SS Equivalent	Qualifier
objectclass	M-EVENT-REPORT parameter 'Managed object class'	M
objectInstance	M-EVENT-REPORT parameter 'Managed object instance'	M
notificationId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -notificationIdentifier	M
eventTime	M-EVENT-REPORT parameter 'Event time'	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT-REPORT parameter 'Event type'	M
probableCause	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -probableCause	M
specificProblems	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -specificProblems	O
perceivedSeverity	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -perceivedSeverity	M
alarmType	The semantics of this parameter is conveyed by the notification type.	--
backedUpStatus	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -backedUpStatus	O
backUpObject	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -backUpObject	O
trendIndication	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -trendIndication	O
thresholdInfo	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -thresholdInfo	O
correlatedNotifications	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -correlatedNotifications	O
stateChangeDefinition	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -stateChangeDefinition	O
monitoredAttributes	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -monitoredAttributes	O
proposedRepairActions	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): proposedRepairActions	O
additionalText	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalText	O
alarmId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): notificationIdentifier M-EVENT-REPORT parameter 'Managed object instance'	M

Table 12a: Parameter mapping of the notification *notifyNewAlarm* for alarms related to security

IS Parameter	CMIP SS Equivalent	Qualifier
objectclass	M-EVENT-REPORT parameter 'Managed object class'	M
objectInstance	M-EVENT-REPORT parameter 'Managed object instance'	M
notificationId	M-EVENT-REPORT parameter 'Event information' (SecurityAlarmInfo): notificationIdentifier	M
eventTime	M-EVENT-REPORT parameter 'Event time'	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT-REPORT parameter 'Event type'	M
probableCause	M-EVENT-REPORT parameter 'Event information' (SecurityAlarmInfo): securityAlarmCause	M
perceivedSeverity	M-EVENT-REPORT parameter 'Event information' (SecurityAlarmInfo): securityAlarmSeverity	M
alarmType	The semantics of this parameter is conveyed by the notification type.	--
correlatedNotifications	M-EVENT-REPORT parameter 'Event information' (SecurityAlarmInfo): correlatedNotifications	O
additionalText	M-EVENT-REPORT parameter 'Event information' (SecurityAlarmInfo): -additionalText	O
serviceUser	serviceUser	M
serviceProvider	serviceProvider	M
securityAlarmDetector	securityAlarmDetector	M
alarmId	M-EVENT-REPORT parameter 'Event information' (SecurityAlarmInfo): notificationIdentifier M-EVENT-REPORT parameter 'Managed object instance'	M

Table 13: Parameter mapping of the notification ~~Mapping of Parameters of "~~*notifyClearedAlarm*~~"~~

IS Parameter	CMIP SS Equivalent	Qualifier
objectclass	M-EVENT-REPORT parameter 'Managed object class'	M
objectInstance	M-EVENT-REPORT parameter 'Managed object instance'	M
notificationId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -notificationIdentifier	M
eventTime	M-EVENT-REPORT parameter 'Event time'	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT-REPORT parameter 'Event type'	M
probableCause	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -probableCause	M
perceivedSeverity	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -perceivedSeverity	M
alarmType	The semantics of this parameter is conveyed by the notification type.	--
clearUserId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalInformation: clearUserIdParameter	O
clearSystemId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalInformation: clearSystemIdParameter	O
correlatedNotifications	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -correlatedNotifications	O
alarmId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -correlatedNotifications	M

Table 14: Parameter mapping of the notification Mapping of Parameters of 'notifyAckStateChanged'

IS Parameter	CMIP SS Equivalent	Qualifier
objectclass	M-EVENT-REPORT parameter 'Managed object class'	M
objectInstance	M-EVENT-REPORT parameter 'Managed object instance'	M
notificationId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -notificationIdentifier	M
eventTime	M-EVENT-REPORT parameter 'Event time'	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT-REPORT parameter 'Event type'	M
probableCause	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -probableCause	M
perceivedSeverity	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -perceivedSeverity	M
alarmType	The semantics of this parameter is conveyed by the notification type.	--
alarmId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -correlatedNotifications	M--
ackTime	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalInformation: ackTimeParameter	M
ackState	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalInformation: ackStateParameter	M
ackUserId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalInformation: ackUserIdParameter	M
ackSystemId	M-EVENT-REPORT parameter 'Event information' (AlarmInfo): -additionalInformation: ackSystemIdParameter	O

Table 15: Parameter mapping of the notification Mapping of Parameters of 'notifyAlarmListRebuilt'

IS Parameter	CMIP SS Equivalent	Qualifier
objectclass	M-EVENT-REPORT parameter 'Event information' (<u>NotifyAlarmListRebuiltInfo</u>): rebuiltObjectClass	M
objectInstance	M-EVENT-REPORT parameter 'Event information' (<u>NotifyAlarmListRebuiltInfo</u>): rebuiltObjectInstance	M
notificationId	M-EVENT-REPORT parameter 'Event information' (<u>NotifyAlarmListRebuiltInfo</u>): notificationIdentifier	M
eventTime	M-EVENT-REPORT parameter 'Event time'	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT-REPORT parameter 'Event type'	M
reason	M-EVENT-REPORT parameter 'Event information' (<u>NotifyAlarmListRebuiltInfo</u>): -reason	M
AlarmListAlignment Requirement	M-EVENT-REPORT parameter 'Event information' (<u>NotifyAlarmListRebuiltInfo</u>): alarmListAlignmentRequirement (see note)	O
NOTE: This parameter shall be supported only, if the IRPAgent supports the notification <u>notifyPotentialFaultyAlarmList</u> .		

Table 16: Parameter mapping of the notification Mapping of Parameters of 'notifyComments'

IS Parameter	CMIP SS Equivalent	Qualifier
objectClass	M-EVENT REPORT parameter 'Event information' (AlarmInfo): alarmedObjectClass	M
objectInstance	M-EVENT REPORT parameter 'Event information' (AlarmInfo): alarmedObjectInstance	M
notificationId	M-EVENT REPORT parameter 'Event information' (AlarmInfo): notificationIdentifier	M
eventTime	M-EVENT REPORT parameter 'Event information' (AlarmInfo): alarmEventTime	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT REPORT parameter 'Event type'	M
alarmType	M-EVENT REPORT parameter 'Event information' (AlarmInfo): alarmType	M
probableCause	M-EVENT REPORT parameter 'Event information' (AlarmInfo): alarmProbableCause	M
perceivedSeverity	M-EVENT REPORT parameter 'Event information' (AlarmInfo): alarmPerceivedSeverity	M
comments	M-EVENT REPORT parameter 'Event information' (AlarmInfo): comments	M
alarmId	M-EVENT REPORT parameter 'Event information' (AlarmInfo): correlatedNotifications	--

**Table 17: Parameter mapping of the notification ~~Mapping of Parameters of~~
~~'notifyPotentialFaultyAlarmList'~~**

IS Parameter	CMIP SS Equivalent	Qualifier
objectClass	M-EVENT-REPORT parameter 'Event information' (NotifyPotentialFaultyAlarmList Info): -potentialFaultyObjectClass	M
objectInstance	M-EVENT-REPORT parameter 'Event information' (NotifyPotentialFaultyAlarmList Info): -potentialFaultyObjectInstance	M
notificationId	M-EVENT-REPORT parameter 'Event information' (NotifyPotentialFaultyAlarmList Info): -notificationIdentifier	M
eventTime	M-EVENT-REPORT parameter 'Event time'	M
systemDN	This IS parameter is conditional and not used in the CMIP SS.	--
notificationType	M-EVENT-REPORT parameter: 'Event type'	M
reason	M-EVENT-REPORT parameter 'Event information' (NotifyPotentialFaultyAlarmList Info): -reason	M

5 GDMO definitions

5.1 Managed Object Classes

5.1.1 alarmControl

This Managed Object Class (MOC) models the alarm information available within the Agent and significant for the NM-EM interface. It deals with both active and cleared but not yet acknowledged alarms. The NMC may initiate the transfer of current alarms according to the required parameters in the M-ACTION request 'getAlarmList'.

```
alarmControl MANAGED OBJECT CLASS
DERIVED FROM
  "Rec. X.721 | ISO/IEC 10165-2 : 1992":top;
CHARACTERIZED BY
  alarmControlBasicPackage,
  alarmAcknowledgementPackage,
  alarmIRPVersionPackage;

CONDITIONAL PACKAGES
  alarmCountPackage PRESENT IF "an instance supports it",
  alarmCommentPackage PRESENT IF "an instance supports it",
  alarmProfilePackage PRESENT IF "an instance supports it",
  alarmUnacknowledgementPackage PRESENT IF "an instance supports it-",
  alarmPotentialFaultyAlarmListPackage PRESENT IF "an instance supports it-",
  alarmClearPackage PRESENT IF "an instance supports it ";
REGISTERED AS {-ts32-111AlarmObjectClass 1};
```

5.2 Packages

5.2.1 alarmControlBasicPackage

```
alarmControlBasicPackage PACKAGE
BEHAVIOUR
  alarmControlBasicPackageBehaviour;
ATTRIBUTES
  alarmControlId          GET,
  alarmsCountSummary     GET;
ACTIONS
  getAlarmList;
NOTIFICATIONS
  notifyAlarmListRebuilt;
REGISTERED AS {-ts32-111AlarmPackage 1};
```

```
alarmControlBasicPackageBehaviour BEHAVIOUR
DEFINED AS
```

"The MOC alarmControl has been defined to provide information to the Manager about the currently alarms controlled by the Agent.

An instance of the 'alarmControl' MOC is identified by the value of the attribute 'alarmControlId'.

The attribute 'alarmsCountSummary' provides a summary of the number of alarms managed in the Agent's alarm list (including the number of cleared but not yet acknowledged alarms).

The action 'getAlarmList' is the means, for the Manager, to trigger an alarm alignment procedure in accordance with the parameter specified in the action request (this may be needed e.g. for first time alignment or after a link interruption between the Agent and the Manager). The alarm list is sent as a sequence of single alarm reports.

The notification 'alarmListRebuilt' is sent by the Agent to the Manager to inform that the alarm list has changed. It is recommended that the Manager subsequently triggers an alarm alignment."

5.2.2 alarmCountPackage

```
alarmCountPackage PACKAGE
BEHAVIOUR
  alarmCountPackageBehaviour;
ACTIONS
  getAlarmCount;
REGISTERED AS {-ts32-111AlarmPackage 2};
```

```
alarmCountPackageBehaviour BEHAVIOUR
DEFINED AS
```

"This package has been defined to allow the Managers to get information from the Agent about the number of alarms currently present in the alarm list.";

5.2.3 alarmAcknowledgementPackage

```
alarmAcknowledgementPackage PACKAGE
```

BEHAVIOUR

```
alarmAcknowledgementPackageBehaviour;
```

ACTIONS

```
acknowledgeAlarms;
```

NOTIFICATIONS

```
"Rec. X.721 | ISO/IEC 10165-2 : 1992":communicationsAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":environmentalAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":equipmentAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":processingErrorAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":qualityofServiceAlarm;
```

```
REGISTERED AS {-ts32-111AlarmPackage 3};
```

```
alarmAcknowledgementPackageBehaviour BEHAVIOUR
```

DEFINED AS

"This package has been defined to provide information to the Manager about the acknowledgement status of the alarms controlled by the Agent.

The action 'acknowledgeAlarms' allows the NM operator to acknowledge one or several alarms previously sent by the Agent as alarm notifications.

The ITU-T Recommendation X.721 [4] compliant alarm notifications are sent by the Agent to the Manager to inform that one alarm has been acknowledged. The acknowledgement related information is carried in the *additionalInformation* attribute.";

5.2.4 alarmUnacknowledgementPackage

```
alarmUnacknowledgementPackage PACKAGE
```

BEHAVIOUR

```
alarmUnacknowledgementPackageBehaviour;
```

ACTIONS

```
unacknowledgeAlarms;
```

NOTIFICATIONS

```
"Rec. X.721 | ISO/IEC 10165-2 : 1992":communicationsAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":environmentalAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":equipmentAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":processingErrorAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":qualityofServiceAlarm;
```

```
REGISTERED AS {-ts32-111AlarmPackage 4};
```

```
alarmUnacknowledgementPackageBehaviour BEHAVIOUR
```

DEFINED AS

"This package has been defined to provide the Manager with the capability to un-acknowledge alarms.

The action 'unacknowledgeAlarms' allows the NM operator to un-acknowledge one or several alarms previously acknowledged by him.

The ITU-T Recommendation X.721 [4] compliant alarm notifications are sent by the Agent to the Manager to inform that one alarm has been unacknowledged. The acknowledgement related information is carried in the *additionalInformation* attribute.";

5.2.5 alarmCommentPackage

```
alarmCommentPackage PACKAGE
```

BEHAVIOUR

```
alarmCommentPackageBehaviour;
```

ACTIONS

```
setComment;
```

NOTIFICATIONS

```
notifyComments;
```

```
REGISTERED AS {-ts32-111AlarmPackage 5};
```

```
alarmCommentPackageBehaviour BEHAVIOUR
```

DEFINED AS

"This package has been defined to allow the Operators to write comments about alarms that are in the alarm list of the IRP Agent.";

5.2.6 alarmIRPVersionPackage

```
alarmIRPVersionPackage PACKAGE
  BEHAVIOUR
    alarmIRPVersionPackageBehaviour;
  ATTRIBUTES
    supportedAlarmIRPVersions          GET;
  ACTIONS
    getAlarmIRPVersion;
REGISTERED AS {-ts32-111AlarmPackage 6};
```

```
alarmIRPVersionPackageBehaviour BEHAVIOUR
DEFINED AS
```

"This package has been defined to allow the Manager to get information about the Alarm IRP versions supported by the Agent.
The attribute 'supportedAlarmIRPVersions' indicates all versions of the Alarm IRP currently supported by the Agent.
The action 'getAlarmIRPVersion' may be invoked by the Manager to get information about the Alarm IRP versions supported by the Agent. Such Alarm IRP versions must compatible to each other. This means that the Manager may use any one of such Alarm IRP versions";

5.2.7 alarmProfilePackage

```
alarmProfilePackage PACKAGE
  BEHAVIOUR
    alarmProfilePackageBehaviour;
  ACTIONS
    getAlarmIRPOperationProfile,
    getAlarmIRPNotificationProfile;
REGISTERED AS {-ts32-111AlarmPackage 7};
```

```
alarmProfilePackageBehaviour BEHAVIOUR
DEFINED AS
```

"This package has been defined to allow the Manager to get detailed information about the profile of Alarm IRP.
The action 'getOperationProfile' is invoked by the Manager to get detailed information about the operations supported by Alarm IRP.
The action 'getNotificationProfile' is invoked by the Manager to get detailed information about the notifications supported by Alarm IRP.";

5.2.8 alarmPotentialFaultyAlarmListPackage

```
alarmPotentialFaultyAlarmListPackage PACKAGE
  BEHAVIOUR
    alarmPotentialFaultyAlarmListPackageBehaviour;
  NOTIFICATIONS
    notifyPotentialFaultyAlarmList;
REGISTERED AS {ts32-111AlarmPackage 8};
```

```
alarmPotentialFaultyAlarmListPackageBehaviour BEHAVIOUR
DEFINED AS
```

"This package allows the IRPAgent to inform the IRPManager that the alarm list held by the IRPAgent might be faulty.";

5.2.9 alarmClearPackage

```
alarmClearPackage PACKAGE
  BEHAVIOUR
    alarmClearPackageBehaviour;
  ACTIONS
    clearAlarms;
REGISTERED AS {ts32-111AlarmPackage 9};
```

```
alarmClearPackageBehaviour BEHAVIOUR
DEFINED AS
```

"This package allows the IRPManager to clear one or multiple alarms in the IRPAgent.";

5.3 Actions

5.3.1 acknowledgeAlarms (M)

```

acknowledgeAlarms ACTION
  BEHAVIOUR
    acknowledgeAlarmsBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule-.AckOrUnackAlarms;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule-.AckOrUnackAlarmsReply;
  REGISTERED AS {-ts32-111AlarmAction 1};

```

```

acknowledgeAlarmsBehaviour BEHAVIOUR

```

DEFINED AS

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

"This action is invoked by the Manager to indicate to the Agent that one or several alarms (previously sent by the Agent as alarm notifications) have to be acknowledged. In the action request the NM supplies the parameter *ackUserId* and *ackSystemId*. The other acknowledgement history parameters, i.e. alarm acknowledgement state (in this case *acknowledged*) and the acknowledgement time are set by the Agent itself.

The 'Action information' field contains the following data:

- *alarmReferenceList*
This parameter contains a set of MOI (Managed Object Instance) and *notificationIdentifier*. Each pair identifies unambiguously in the scope of the Agent an alarm (previously received by the NM) that have to be now acknowledged. MOI can be absent if scope of uniqueness of *notificationIdentifier* is across the IRPAgent.
- *ackUserId*
It contains the name of the operator who acknowledged the alarm or a generic name (dependent on the operational concept). It may have also the value NULL.
- *ackSystemId*
It indicates the management system where the acknowledgment is triggered. It may have also the value NULL.

The 'Action response' contains the following data:

- *status*
This parameter contains the results of the NM acknowledgement action. Possible values: *noError* (0, all alarms found and ack state changed according to the manager request), *ackPartlySuccessful* (some alarms not found / not changeable, see next parameter), *error* (value indicates the reason why the complete operation failed).
- *errorAlarmReferenceList*
This parameter (significant only if *status* = *ackPartlySuccessful*) contains the list of *moi* (managed object instance) and *notificationIdentifier* pairs of the alarms which could not be acknowledged and, for each alarm, also the reason of the error.";

5.3.2 getAlarmCount (O)

```

getAlarmCount ACTION
  BEHAVIOUR
    getAlarmCountBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule-.GetAlarmCount;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule-.GetAlarmCountReply;
  REGISTERED AS {-ts32-111AlarmAction 2};

```

```

getAlarmCountBehaviour BEHAVIOUR

```

DEFINED AS

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

"The NM invokes this action to receive the number of available alarms in the Agent' alarm list according to the specification in the action request. The Manager may use this action to find out the number of alarms in the alarm list before invoking a synchronisation by means of the *getAlarmList* operation. The request is possible also before the Manager creates an own event forwarding discriminator instance within the Agent.

The 'Action information' field contains the following data:

- *alarmAckState*
Depending on this optional parameter value, the NM gets the number of alarms of each *perceivedSeverity* value according to the following possible choices:
- all alarms

- all active alarms (acknowledged or not yet acknowledged)
- all active and acknowledged alarms
- all active and unacknowledged alarms
- all cleared and unacknowledged alarms.

If the parameter is absent, all alarms from the Agent's alarm list are taken into consideration.

- *filter*
The handling of this optional parameter is as follows:
 - if present and not NULL, it indicates a filter constraint which shall apply in the calculation of the results
 - if its value is NULL, no filter shall be considered and the Agent shall return the number of all alarms according to the value of the parameter *alarmAckState* (see above)
 - if absent, the handling depends on the availability of an event forwarding discriminator instance within the Agent. If this instance is valid, the filter construct of the event forwarding discriminator shall apply. If no EFD instance is available, the Agent shall return the number of all alarms according to the value of the above-mentioned parameter *alarmAckState*.

The 'Action response' is composed of:

- The numbers of alarms for each *perceivedSeverity* value (if applicable).
- The parameter *status* containing the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).";

5.3.3 getAlarmList (M)

```
getAlarmList ACTION
  BEHAVIOUR
    getAlarmListBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule-.GetAlarmList;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule-.GetAlarmListReply;
REGISTERED AS {-ts32-111AlarmAction 3};
```

```
getAlarmListBehaviour BEHAVIOUR
DEFINED AS
```

"This action starts an alarm alignment procedure between a NM and Agent, which takes into account the acknowledgment state of the alarms and a dedicated filter (valid only for the current request).

The 'Action information' field contains the following data:

- *alarmAckState*
Depending on this optional parameter value, the NM gets the alarm reports according to the following possible choices:
 - all alarms
 - all active alarms (acknowledged or not yet acknowledged)
 - all active and acknowledged alarms
 - all active and unacknowledged alarms
 - all cleared and unacknowledged alarms.

If the parameter is absent, all alarms from the Agent's alarm list are taken into consideration.

- *destination*
This parameter identifies the destination to which the alarm reports that have passed the test conditions specified in the parameter 'filter' are sent. According to ITU-T Recommendation X.721 [4], if no destination is specified in the request, then the discriminator is created with the destination defaulted to the AE-Title of the invoker.
- *filter*
The handling of this optional parameter (valid only for the current alignment request) is as follows:
 - if present and not NULL, it indicates a filter constraint which shall apply in the forwarding of the alignment-related alarm reports
 - if its value is NULL, no real filter shall be considered and the Manager receives the alarms according to the value of the parameter *alarmAckState* (see above).

The 'Action response' contains the following data:

- *alignmentId*
The parameter is defined by the Agent and identifies unambiguously the current alarm alignment procedure. It allows the Manager to distinguish between alarm reports sent as consequence of several own alignment requests triggered in parallel.
- *status*
The parameter contains the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).
After the action response is forwarded to the NM, the Agent sends the alarm list as a sequence of single alarm notifications in accordance with the values of the request

parameters. Every alarm notification contains all fields of the alarm stored in the alarm list. In particular:

- The field *additionalText* contains at the beginning a string to allow a Manager to recognise that this alarm report is sent due to a previous *getAlarmList* request. The structure of this string is:
 - '(ALIGNMENT-alignmentId)' for every alarm report except the last one **or**
 - '(ALIGNMENTEND-alignmentId)' for the last alarm report sent by the Agent due to the current *getAlarmList* request.
- If available, the data related to the acknowledgment history (i.e. *ackState*, *ackTime*, *ackUserId*, *ackSystemId*) are provided in the field *additionalInformation*. Further details about the implementation of this operation are provided in the 'Introduction'.";

5.3.4 setComment (O)

```
setComment ACTION
  BEHAVIOUR
    setCommentBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule-.SetComment;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule-.SetCommentReply;
  REGISTERED AS {-ts32-111AlarmAction 4};
```

```
setCommentBehaviour BEHAVIOUR
```

DEFINED AS

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

The NM invokes this action to associate a comment to one or more alarms.

The 'Action information' field contains:

- *alarmReferenceList*
Contains a list of alarm identifiers to which the comment must be associated.
- *commentUserId*
Contains the identity of the User that invokes this operation.
- *commentSystemId*
Contains the identity of the NM that invokes this operation.
- *commentText*
Contains the text of the comment.

The 'Action response' is composed of the following data:

- *errorAlarmReferenceList*
List of pair of *alarmId* and failure reason.
- *status*
It contains the results of the NM action. Possible values: *actionSucceeded* (0), *actionPartiallyFailed* (12) or another value indicating the reason of the error.";

5.3.5 getAlarmIRPVersion (M)

```
getAlarmIRPVersion ACTION
  BEHAVIOUR
    getAlarmIRPVersionBehaviour;
  MODE
    CONFIRMED;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule-.GetAlarmIRPVersionReply;
  REGISTERED AS { ts32-111AlarmAction 5};
```

```
getAlarmIRPVersionBehaviour BEHAVIOUR
```

DEFINED AS

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

The NM invokes this action to get information about the Alarm IRP versions supported by the Agent.

The 'Action information' field contains no data.

The 'Action response' is composed of the following data:

- *versionNumbersList*
It defines a list of Alarm IRP versions supported by the Agent. A list containing no element, i.e. a NULL list means that the concerned Agent doesn't support any version of the Notification IRP.
- *status*
It contains the results of the NM action. Possible values: *noError* (0), *error* (the value indicates the reason of the error).";

5.3.6 [getAlarmIRPNotificationProfile](#) (O)

```
getAlarmIRPNotificationProfile ACTION
  BEHAVIOUR
    getAlarmIRPNotificationProfileBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule.IRPVersionNumber;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule.GetNotificationProfileReply;
  REGISTERED AS {-ts32-111AlarmAction 6};
```

```
getAlarmIRPNotificationProfileBehaviour BEHAVIOUR
DEFINED AS
```

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

A Manager invokes this action to enquiry about the notification profile (supported notifications and supported parameters) for this specific Alarm IRP version.

The 'Action information' contains the following data:

- *irpVersionNumber*
This mandatory parameter identifies the Alarm IRP version.

The 'Action response' is composed of the following data:

- *notificationNameProfile*
It contains a list of notification names, i.e. a NULL list means that the Alarm IRP doesn't support any notification.
- *notificationParameterProfile*.
It contains a set of elements, each element corresponds to a notification name and is composed by a set of parameter names.
- *status*
It contains the results of this action. Possible values: noError (0), error (the value indicates the reason of the error).";

5.3.7 [getAlarmIRPOperationProfile](#) (O)

```
getOperationProfile ACTION
  BEHAVIOUR
    getAlarmIRPOperationProfileBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule.IRPVersionNumber;
  WITH REPLY SYNTAX
    TS32-111-4TypeModule.GetOperationProfileReply;
  REGISTERED AS {-ts32-111AlarmAction 7};
```

```
getAlarmIRPOperationProfileBehaviour BEHAVIOUR
DEFINED AS
```

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

A Manager invokes this action to enquiry about the operation profile (supported operations and supported parameters) for this specific Alarm IRP version.

The 'Action information' contains the following data:

- *irpVersionNumber*
This mandatory parameter identifies the Alarm IRP version.

The 'Action response' is composed of the following data:

- *operationNameProfile*
It contains a list of operation names.
- *operationParameterProfile*.
It contains a set of elements, each element corresponds to an operation name and is composed by a set of parameter names.
- *status*
It contains the results of this action. Possible values: noError (0), error (the value indicates the reason of the error).";

5.3.8 [unacknowledgeAlarms](#) (O)

```
unacknowledgeAlarms ACTION
  BEHAVIOUR
    unacknowledgeAlarmsBehaviour;
  MODE
    CONFIRMED;
  WITH INFORMATION SYNTAX
```

```

    TS32-111-4TypeModule.AckOrUnackAlarms;
    WITH REPLY SYNTAX
    TS32-111-4TypeModule.AckOrUnackAlarmsReply;
    REGISTERED AS {-ts32-111AlarmAction 8};

```

unacknowledgeAlarmsBehaviour **BEHAVIOUR**
DEFINED AS

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

This action is used by the Manager to indicate to the Agent that one or several alarms (previously acknowledged) have to be unacknowledged. Subsequently the 'acknowledgement history' information of these alarms in the Agent's alarm list is completely removed (this operation may be used by operators in case of a previous acknowledgement by mistake).

The 'Action information' field contains the following data:

- *alarmReferenceList*
 This parameter contains a set of MOI (Managed Object Instance) and *notificationIdentifier* pair. Each of them identifies unambiguously in the scope of the Agent an alarm (previously acknowledged by the NM) that have to be now unacknowledged. MOI can be absent if scope of uniqueness of notificationIdentifier is across the IRPagent.
- *ackUserId*
 It contains the name of the operator who unacknowledged the alarm or a generic name (dependent on the operational concept). It may have also the value NULL. Note that only the user who previously acknowledged the alarm is allowed to un-acknowledge it later.
- *ackSystemId*
 It indicates the management system where the acknowledgment is triggered. It may have also the value NULL. Note that the un-acknowledgement is allowed only at the management system where previously the acknowledgement took place.

The 'Action response' contains the following data:

- *status*
 This parameter contains the results of the NM un-acknowledgement action. Possible values: noError (0, all alarms found and ack state changed according to the manager request), unackPartlySuccessful (some alarms not found / not changeable, see next response parameter), error (value indicates the reason why the complete operation failed).
- *errorAlarmReferenceList*
 This parameter (significant only if *status* = unackPartlySuccessful) contains the list of MOI (Managed Object Instance) and notificationIdentifier pairs of the alarms which could not be unacknowledged and, for each alarm, also the reason of the error. MOI can be absent if scope of uniqueness of notificationIdentifier is across the IRPagent. ";

5.3.9 clearAlarms (O)

```

clearAlarms ACTION
BEHAVIOUR
    clearAlarmsBehaviour;
MODE
    CONFIRMED;
WITH INFORMATION SYNTAX
    TS32-111-4TypeModule.ClearAlarmsInfo;
WITH REPLY SYNTAX
    TS32-111-4TypeModule.ClearAlarmsReply;
    REGISTERED AS {-ts32-111AlarmAction 9};

```

clearAlarmsBehaviour **BEHAVIOUR**
DEFINED AS

"The behaviour of this functionality is defined within 32.111-2 - below provides an overview and CMIP specific semantics.

This action is invoked by the IRPManager to clear manually one or multiple alarms. The M-ACTION request parameter 'Action information' *ClearAlarmsInfo* is composed of the following fields:

- *alarmReferenceList*
 This mandatory parameter identifies the alarms to be cleared. Each alarm is identified by the notification identifier of the notification that reported the alarm the first time and, if the notification identifier is not unique across the IRPagent, by the instance of the managed object that emitted this notification.
- *clearUserId*
 This mandatory parameter identifies the user that has invoked the *clearAlarms* operation.
- *clearSystemId*
 This optional parameter identifies the system on which the IRPManager, where the *clearAlarms* operation has been invoked, is running. This parameter may be absent.

The M-ACTION response parameter 'Action Reply' *ClearAlarmsReply* is composed of the following fields

- *errorAlarmReferenceList*
 This mandatory parameter identifies alarms that are specified in the *alarmReferenceList*, but which could not be cleared. The alarms are specified by the notification identifier of the notification that reported the alarm the first time and, if required, the instance of the managed object that emitted this notification. In addition to this, the parameter specifies for every alarm that could not be cleared the error reason. If all alarms

specified in the *alarmReferenceList* exist and could be cleared, this parameter contains no information. If the operation failed completely due to a general error, this parameter is not significant.

- *status*
This mandatory parameter provides informations about the result of the operation. If all alarms specified in the *alarmReferenceList* exist and are cleared, the value *noError* (0) is returned. If some alarms specified do not exist or could not be cleared, the value *clearPartlySuccessful* () is returned. In this case the parameter *errorAlarmReferenceList* provides additional information. If the operation failed completely due to a general error, this parameter returns the error reason.";

5.4 Notifications

5.4.1 notifyAlarmListRebuilt (M)

```
notifyAlarmListRebuilt NOTIFICATION
  BEHAVIOUR
    alarmListRebuiltBehaviour;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule-.NotifyAlarmListRebuiltInfo;
  REGISTERED AS _-{-ts32-111AlarmNotification 1};
```

```
alarmListRebuiltBehaviour BEHAVIOUR
```

DEFINED AS

"This notification is used by the Agent to inform the NM that the alarm list has been rebuilt. The 'Event Information' field contains the following data:

- *notificationIdentifier*
This ITU-T X.721 standardised parameter, together with MOI (Managed Object Instance), unambiguously identifies this notification.
- *rebuiltObjectClass*
This parameter carries the IRPAgent MOC when the entire AlarmList has been rebuilt. It carries a different MOC when the AlarmList has been partially rebuilt.
- *rebuiltObjectInstance*
This parameter carries DN of the IRPAgent when the entire AlarmList has been rebuilt. It carries the DN of another MOI when the AlarmList has been partially rebuilt and only the MOIs subordinate of this rebuilt MOI may be affected by this partial rebuilt.
- *reason*
The parameter indicates the reason for alarm list rebuilding (if applicable)."+
- *alarmListAlignmentRequirement*
This parameter indicates, if the IRPManager has to align its alarm list with the IRPAgent. Absence of this parameter means, that an alignment is required. _";

5.4.2 notifyComments (O)

```
notifyComments NOTIFICATION
  BEHAVIOUR
    notifyCommentsBehaviour;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule-.NotifyComments;
  REGISTERED AS _-{-ts32-111AlarmNotification 2};
```

```
notifyCommentsBehaviour BEHAVIOUR
```

DEFINED AS

"This notification is used by the Agent to inform the NM that one or more comments have been associated to one alarm.

The 'Event Information' field contains the following data:

- *alarmedObjectClass*
This parameter specifies the object class representing the resource that raised the alarm to which the comment was attached.
- *alarmedObjectInstance*
This parameter specifies the object instance representing the resource that raised the alarm to which the comment was attached.
- *notificationIdentifier*
This parameter specifies the notification identifier (ITU-T X.733 [5]), which, together with the instance of the object emitting this notification, unambiguously identifies this notification.
- *alarmEventTime*
This parameter specifies the time when the alarm, to which the comment was attached, was first raised by the alarmed resource.
- *alarmType*
This parameter specifies the event type of the notification that reported the alarm to which the comment was attached.

- *alarmProbableCause*
This parameter specifies the probable cause (ITU-T X.733 [5]) of the alarm to which the comment was attached.
- *alarmPerceivedSeverity*
This parameter specifies the perceived severity (ITU-T X.733 [5]) of the alarm to which the comment is attached.
- *comments*
This parameter carries the text of the comment.";

5.4.3 notifyPotentialFaultyAlarmList (O)

```
notifyPotentialFaultyAlarmList NOTIFICATION
  BEHAVIOUR
    notifyPotentialFaultyAlarmListBehaviour;
  WITH INFORMATION SYNTAX
    TS32-111-4TypeModule.-NotifyPotentialFaultyAlarmListInfo;
  REGISTERED AS _{-ts32-111AlarmNotification 3};
```

```
notifyPotentialFaultyAlarmListBehaviour BEHAVIOUR
```

DEFINED AS

"This notification is used by the IRPAgent to inform the IRPManager that the IRPAgent has lost confidence in the integrity of its alarm list.

The 'Event information' field contains the following data:

- *potentialFaultyObjectClass*
This parameter specifies together with the parameter *potentialFaultyObjectInstance* the unreliable alarm information instances in the alarm list.
If this parameter carries the MOC of the IRPAgent, then the entire alarm list is unreliable.
If this parameter carries the MOC of another MO, then only a part of the alarm list is unreliable. The mechanism for identifying the unreliable part is described below.
- *potentialFaultyObjectInstance*
This parameter specifies together with the parameter *potentialFaultyObjectClass* the unreliable alarm information instances in the alarm list.
If *potentialFaultyObjectClass* carries the MOC of the IRPAgent, the this parameter carries the DN of the IRPAgent and the entire alarm list is unreliable.
If *potentialFaultyObjectClass* carries the MOC of another MO, then this parameter carries the DN of an instance of this class. All alarm information instances representing alarms raised by this MOI and its subordinates may be unreliable in this case.
- *notificationIdentifier*
This parameter specifies the notification identifier (ITU-T X.733 [5]), which, together with the instance of the object emitting this notification, unambiguously identifies this notification.
- *reason*
This parameter specifies the reason why the IRPAgent has lost confidence in the integrity of its alarm list and needs to rebuild it.";

5.5 Attributes

5.5.1 alarmControlId

```
alarmControlId ATTRIBUTE
  WITH ATTRIBUTE SYNTAX
    TS32-111-4TypeModule-.GeneralObjectId;
  MATCHES FOR
    EQUALITY;
  BEHAVIOUR
    alarmControlIdBehaviour;
  REGISTERED AS {-ts32-111AlarmAttribute 1};
```

```
alarmControlIdBehaviour BEHAVIOUR
```

DEFINED AS

"This attribute names an instance of a 'alarmControl' object class.";

5.5.2 alarmsCountSummary

```
alarmsCountSummary ATTRIBUTE
  WITH ATTRIBUTE SYNTAX
    TS32-111-4TypeModule-.AlarmsCountSummary;
  MATCHES FOR
    EQUALITY;
  BEHAVIOUR
```



```
alarmsCountSummaryBehaviour;
REGISTERED AS {-ts32-111AlarmAttribute 2};
```

```
alarmsCountSummaryBehaviour BEHAVIOUR
DEFINED AS
```

"This attribute indicates a summary of number of alarms managed in the Agent's alarm list sorted according to the perceived severity (including the number of cleared but not yet acknowledged alarms). Additionally the number of all currently active alarms is provided.";

5.5.3 supportedAlarmIRPVersions

```
supportedAlarmIRPVersions ATTRIBUTE
WITH ATTRIBUTE SYNTAX
TS32-111-4TypeModule-.SupportedAlarmIRPVersions;
MATCHES FOR
EQUALITY;
BEHAVIOUR
supportedAlarmIRPVersionsBehaviour;
REGISTERED AS {-ts32-111AlarmAttribute 3};
```

```
supportedAlarmIRPVersionsBehaviour BEHAVIOUR
DEFINED AS
```

"This attribute provides the information concerning the Alarm IRP versions currently supported by the Agent.";

5.6 Parameters

5.6.1 ackStateParameter

```
ackStateParameter PARAMETER
CONTEXT
TS32-111-4TypeModule-.AlarmInfo.additionalInformation;
WITH SYNTAX
TS32-111-4TypeModule-.AckState;
BEHAVIOUR
ackStateParameterBehaviour;
REGISTERED AS _{-ts32-111AlarmParameter 1};
```

```
ackStateParameterBehaviour BEHAVIOUR
DEFINED AS
```

"This parameter models the optional *additionalInformation* field of the alarm notification. If present, it informs the NM about the current acknowledgement state of the present alarm.";

5.6.2 ackSystemIdParameter

```
ackSystemIdParameter PARAMETER
CONTEXT
TS32-111-4TypeModule-.AlarmInfo.additionalInformation;
WITH SYNTAX
TS32-111-4TypeModule-.SystemId;
BEHAVIOUR
ackSystemIdParameterBehaviour;
REGISTERED AS _{-ts32-111AlarmParameter 2};
```

```
ackSystemIdParameterBehaviour BEHAVIOUR
DEFINED AS
```

"This parameter models the optional *additionalInformation* field of the alarm notification. If present, it informs the NM about the identifier of the management system where the present alarm has been acknowledged.";

5.6.3 ackTimeParameter

```
ackTimeParameter PARAMETER
CONTEXT
TS32-111-4TypeModule-.AlarmInfo.additionalInformation;
WITH SYNTAX
TS32-111-4TypeModule-.AckTime;
BEHAVIOUR
ackTimeParameterBehaviour;
REGISTERED AS _{-ts32-111AlarmParameter 3};
```

```
ackTimeParameterBehaviour BEHAVIOUR
```

DEFINED AS

"This parameter models the optional *additionalInformation* field of the alarm notification. If present, it informs the NM about the time the present alarm has been acknowledged by the Agent.";

5.6.4 ackUserIdParameter

ackUserIdParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule .AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.UserId;

BEHAVIOUR

ackUserIdParameterBehaviour;

REGISTERED AS `_{-ts32-111AlarmParameter 4};`

ackUserIdParameterBehaviour **BEHAVIOUR**

DEFINED AS

"This parameter models the optional *additionalInformation* field of the alarm notification. If present, it informs the NM about the identifier of the user who acknowledged the present alarm.";

5.6.5 clearUserIdParameter

clearUserIdParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule .AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.UserId;

BEHAVIOUR

clearUserIdParameterBehaviour;

REGISTERED AS `_{-ts32-111AlarmParameter 5};`

clearUserIdParameterBehaviour **BEHAVIOUR**

DEFINED AS

"This parameter is carried by *additionalInformation* in the notification reporting the clearance of an alarm. It identifies the user that has invoked the *clearAlarms* operation, that has led to the clearance of the ~~the~~ reported alarm clearance.";

5.6.6 clearSystemIdParameter

clearSystemIdParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule .AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.UserId;

BEHAVIOUR

clear~~SystemUser~~IdParameterBehaviour;

REGISTERED AS `_{-ts32-111AlarmParameter 6};`

clear~~SystemUser~~IdParameterBehaviour **BEHAVIOUR**

DEFINED AS

"This parameter is carried by *additionalInformation* in the notification reporting the clearance of an alarm. It identifies the system on which the IRPManager, where the *clearAlarms* operation that has led to the clearance of the reported alarm, is running";

6 ASN.1 definitions for Alarm IRP

```
TS32-111-4TypeModule {itu-t(0) identified-organization(4) etsi(0) mobileDomain(0) umts-Operation-
Maintenance(3) ts-32-111(111) part4(4) informationModel(0) asn1Module(2) version1(1)}
```

```
DEFINITIONS IMPLICIT TAGS ::=
```

```
BEGIN
```

```
--EXPORTS everything
```

```
IMPORTS
```

```
NotificationIdentifier, Destination, EventTime, ProbableCause, PerceivedSeverity
```

```
FROM Attribute-ASN1Module {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1}
```

```
AlarmInfo
```

```
FROM Notification-ASN1Module {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 2}
```

```
CMISFilter, ObjectInstance, ObjectClass, EventTypeId
```

```
FROM CMIP-1 {joint-iso-ccitt ms(9) cmip(1) modules(0) protocol(3)};
```

```
baseNodeUMTS OBJECT IDENTIFIER ::= { itu-t (0) identified-organization (4) etsi (0) mobileDomain
```

```
(0)
```

```
umts-Operation-Maintenance (3) }
```

```
ts32-111Prefix OBJECT IDENTIFIER ::= { baseNodeUMTS ts-32-111(111)}
```

```
ts32-111Part4 OBJECT IDENTIFIER ::= { ts32-111Prefix part4(4)}
```

```
ts32-111-4InfoModel OBJECT IDENTIFIER ::= { ts32-111Part4 informationModel(0)}
```

```
ts32-111AlarmObjectClass OBJECT IDENTIFIER ::= { ts32-111-4InfoModel managedObjectClass(3)}
```

```
ts32-111AlarmPackage OBJECT IDENTIFIER ::= { ts32-111-4InfoModel package(4)}
```

```
ts32-111AlarmParameter OBJECT IDENTIFIER ::= { ts32-111-4InfoModel parameter(5)}
```

```
ts32-111AlarmAttribute OBJECT IDENTIFIER ::= { ts32-111-4InfoModel attribute(7)}
```

```
ts32-111AlarmAction OBJECT IDENTIFIER ::= { ts32-111-4InfoModel action(9)}
```

```
ts32-111AlarmNotification OBJECT IDENTIFIER ::= { ts32-111-4InfoModel actionnotification(10)}
```

```
-- Start of 3GPP SA5 own definitions
```

```
AckErrorList ::= SET OF ErrorInfo
```

```
AlarmReference ::= SEQUENCE
```

```
{
```

```
moi ObjectInstance OPTIONAL, -- absent if scope of uniqueness of
notificationIdentifier NotificationIdentifier
}
```

```
AckOrUnackAlarms ::= SEQUENCE
```

```
{
```

```
alarmReferenceList SET OF AlarmReference,
ackUserId UserId,
ackSystemId SystemId OPTIONAL
}
```

```
AckOrUnackAlarmsReply ::= SEQUENCE
```

```
{
```

```
status ErrorCauses,
errorAlarmReferenceList AckErrorList
}
```

```
AckState ::= ENUMERATED
```

```
{
```

```
acknowledged (0),
unacknowledged (1)
}
```

```
AckTime ::= GeneralizedTime
```

```
AlarmChoice ::= ENUMERATED
```

```
{
```

```
allAlarms (0),
allActiveAlarms (1),
allActiveAndAckAlarms (2),
allActiveAndUnackAlarms (3),
allClearedAndUnackAlarms (4),
```

```

    allUnackAlarms          (5)
}

AlarmsCountSummary ::= SEQUENCE
{
    activeAlarmsCount      INTEGER,      -- this is the sum of criticalCount, majorCount,
                                        -- minorCount, warningCount
                                        -- and indeterminateCount

    criticalCount          INTEGER,
    majorCount             INTEGER,
    minorCount             INTEGER,
    warningCount           INTEGER,
    indeterminateCount     INTEGER,
    clearedCount          INTEGER
}

AlarmListRebuiltInfo ::= SEQUENCE
{
notificationIdentifier NotificationIdentifier, -- ITU-T X.721
rebuiltObjectClass      ObjectClass,
rebuiltObjectInstance  ObjectInstance,
reason                  ErrorCauses,
alarmListAlignmentRequirement AlarmListAlignmentRequirement
}

AlarmListAlignmentRequirement ::= ENUMERATED
{
    alignmentRequired      (0),          -- An alarm alignment is required.
    alignmentNotRequired   (1)          -- An alarm alignment is not required.
}

ClearAlarmsInfo ::= SEQUENCE
{
    alarmReferenceList     SET OF AlarmReference,
    clearUserId            UserId,
    clearSystemId          SystemId OPTIONAL
}

ClearAlarmsReply ::= SEQUENCE
{
    status                 ErrorCauses,
    errorAlarmReferenceList ClearErrorList
}

ClearErrorList ::= SET OF ErrorInfo

Comment ::= GraphicString

ErrorCauses ::= ENUMERATED
{
    noError (0),                -- operation / notification successfully performed
    wrongFilter (1),            -- the value of the filter parameter is not valid
    wrongAlarmAckState (2),     -- the value of the alarmAckState parameter (e.g.
                                -- getAlarmCount) is not valid
    ackPartlySuccessful (3),    -- acknowledgment request partly successful
    unackPartlySuccessful (4),  -- unacknowledgment request partly successful
    wrongAlarmReference (5),    -- alarm identifier used in the alarm reference list not found
                                -- (e.g. in case of acknowledgement request)
    wrongAlarmReferenceList (6), -- the alarm reference list (e.g. in case of acknowledgement
                                -- request) is empty or completely wrong
    alarmAlreadyAck (7),        -- alarm to be acknowledged is already in this state
    alarmAlreadyUnack (8),      -- alarm to be acknowledged is already in this state
    wrongUserId (9),            -- the user identifier in the unacknowledgement operation is
                                -- not the same as in the previous acknowledgementAlarms
                                -- request
    wrongSystemId (10),         -- the system identifier in the unacknowledgement operation is
                                -- not the same as in the previous acknowledgementAlarms
                                -- request
    alarmAckNotAllowed (11),    -- current management system not allowed to acknowledge the
                                -- alarm (e.g. due to acknowledgement competence rules)
    setCommentPartlySuccessful (12), -- the setComment action partly successful (e.g. some alarmId
                                -- are not in the alarmList)
    clearAlarmsPartlySuccessful (13), -- only some alarms to be cleared could be cleared
    clearAlarmsNotAllowed (14),    -- current management system not allowed to clear the alarm
    clearAlarmsAlarmAlreadyCleared (15), -- alarm to be cleared is already cleared
    unspecifiedErrorReason (255) -- operation failed, specific error unknown
}

```

```

ErrorInfo ::= SEQUENCE
{
    moi                ObjectInstance OPTIONAL,    -- absent if uniqueness of
                                                         -- notificationIdentifier is across
                                                         -- IRPAgent
    notificationIdentifier NotificationIdentifier,  -- ITU-T X.721
    reason              ErrorCauses
}

GeneralObjectId ::= INTEGER

GetAlarmCount ::= SEQUENCE
{
    alarmAckState      AlarmChoice OPTIONAL,
    filter             CMISFilter OPTIONAL -- ITU-T X.711
}

GetAlarmCountReply ::= SEQUENCE
{
    criticalCount      INTEGER,
    majorCount         INTEGER,
    minorCount         INTEGER,
    warningCount       INTEGER,
    indeterminateCount INTEGER,
    clearedCount       INTEGER,
    status             ErrorCauses
}

GetAlarmIRPVersionReply ::= SEQUENCE
{
    versionNumberList SupportedAlarmIRPVersions,
    status             ErrorCauses
}

GetAlarmList ::= SEQUENCE
{
    alarmAckState      AlarmChoice OPTIONAL,
    destination        Destination,           -- ITU-T X.721
    filter             CMISFilter OPTIONAL -- ITU-T X.711
}

GetAlarmListReply ::= SEQUENCE
{
    alignmentId        INTEGER,
    status             ErrorCauses
}

GetNotificationProfileReply ::= SEQUENCE
{
    notificationNameProfile NotificationList,
    notificationParameterProfile ParameterListOfList,
    status             ErrorCauses
}

GetOperationProfileReply ::= SEQUENCE
{
    operationNameProfile OperationList,
    operationParameterProfile ParameterListOfList,
    status             ErrorCauses
}

IRPVersionNumber ::= GraphicString

NotificationList ::= SET OF NotificationName

NotificationName ::= GraphicString

NotifyAlarmListRebuiltInfo ::= SEQUENCE
{
    notificationIdentifier NotificationIdentifier,  -- ITU-T X.721
    rebuiltObjectClass     ObjectClass,           -- ITU-T X.721
    rebuiltObjectInstance  ObjectInstance,       -- ITU-T X.721
    reason                 ErrorCauses,
    alarmListAlignmentRequirement AlarmListAlignmentRequirement OPTIONAL
}

NotifyComments ::= SEQUENCE
{
    alarmedObjectClass    ObjectClass,           -- ITU-T X.711
    alarmedObjectInstance ObjectInstance,       -- ITU-T X.711
    notificationIdentifier NotificationIdentifier, -- ITU-T X.721
}

```

```

alarmEventTime      EventTime,           -- ITU-T X.721
alarmType           EventTypeId,        -- ITU-T X.711
alarmProbableCause  ProbableCause,     -- ITU-T X.721
alarmPerceivedSeverity PerceivedSeverity, -- ITU-T X.721
comments           SET OF Comment
}

NotifyPotentialFaultyAlarmListInfo ::= SEQUENCE
{
    potentialFaultyObjectClass  ObjectClass,           -- ITU-T X.711
    potentialFaultyObjectInstance ObjectInstance,     -- ITU-T X.711
    notificationIdentifier      NotificationIdentifier, -- ITU-T X.721
    reason                      ReasonPotentialFaultyAlarmList
}

OperationList ::= SET OF OperationName

OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

ReasonPotentialFaultyAlarmList ::= ENUMERATED
{
    communicationErrorNEAgent (0) 1 -- A communication error between a NE and the agent has
    occurred.
    agentRestart list (1) 1 -- The agent has restarted and not yet updated its alarm
    list.
    indeterminate (2) 1 -- The reason could not be determined.
}

SetComment ::= SEQUENCE
{
    alarmReferenceList SET OF AlarmReference,
    commentUserId      UserId,
    commentSystemId    SystemId,
    commentText        Comment
}

SetCommentReply ::= SEQUENCE
{
    baderrorAlarmReferenceList SET OF ErrorInfo,
    status                      ErrorCauses
}

SystemId ::= GraphicString

SupportedAlarmIRPVersions ::= SET OF IRPVersionNumber

UserId ::= GraphicString

END -- of module TS32-111-4TypeModule

```

End of Change in Clause 4 and 5
--

CHANGE REQUEST

⌘ **32.111-4 CR 017** ⌘ rev **-** ⌘ Current version: **5.3.0** ⌘

For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.

Proposed change affects: UICC apps ME Radio Access Network Core Network

Title:	⌘ Add missing x721AlarmNotificationsPackage		
Source:	⌘ S5		
Work item code:	⌘ OAM-NIM	Date:	⌘ 28/02/2003
Category:	⌘ F	Release:	⌘ Rel-5
	Use <u>one</u> of the following categories: F (correction) A (corresponds to a correction in an earlier release) B (addition of feature), C (functional modification of feature) D (editorial modification) Detailed explanations of the above categories can be found in 3GPP TR 21.900 .		Use <u>one</u> of the following releases: 2 (GSM Phase 2) R96 (Release 1996) R97 (Release 1997) R98 (Release 1998) R99 (Release 1999) Rel-4 (Release 4) Rel-5 (Release 5) Rel-6 (Release 6)

Reason for change:	⌘ Alarms in many object classes of the CMIP solutions sets are missing. 32.111-4 is the right place to define an alarm related package for general re-use.
Summary of change:	⌘ X721AlarmNotificationsPackage defined
Consequences if not approved:	⌘ No alarming possible for most object classes

Clauses affected:	⌘ 5.2										
Other specs affected:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;">⌘</td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;">⌘</td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;">X</td> <td style="text-align: center;">⌘</td> </tr> </table>	Y	N	⌘	X	⌘	X	X	⌘	Other core specifications	⌘
	Y	N									
	⌘	X									
⌘	X										
X	⌘										
		Test specifications									
		O&M Specifications	32.634/ 644/ 654 re-use this package								
Other comments:	⌘										

How to create CRs using this form:

Change in Clause 5.2

5.2.9 alarmClearPackage

```
alarmClearPackage PACKAGE
  BEHAVIOUR
    alarmClearPackageBehaviour;
  ACTIONS
    clearAlarms;
  REGISTERED AS {ts32-111AlarmPackage 9};
```

```
alarmClearPackageBehaviour BEHAVIOUR
DEFINED AS
  "This package allows the IRPManager to clear one or multiple alarms in the IRPAgent.";
```

5.2.10 x721AlarmNotificationsPackage

```
x721AlarmNotificationsPackage PACKAGE
  BEHAVIOUR
    x721AlarmNotificationsPackageBehaviour;
  NOTIFICATIONS

|             |  |                                                |
|-------------|--|------------------------------------------------|
| "Rec. X.721 |  | ISO/IEC 10165-2 : 1992":communicationsAlarm,   |
| "Rec. X.721 |  | ISO/IEC 10165-2 : 1992":environmentalAlarm,    |
| "Rec. X.721 |  | ISO/IEC 10165-2 : 1992":equipmentAlarm,        |
| "Rec. X.721 |  | ISO/IEC 10165-2 : 1992":processingErrorAlarm,  |
| "Rec. X.721 |  | ISO/IEC 10165-2 : 1992":qualityofServiceAlarm; |

REGISTERED AS { ts32-111AlarmPackage 3};
```

```
x721AlarmNotificationsPackage BEHAVIOUR
DEFINED AS
  "This package contains all alarm notifications defined in X.721.";
```

End of Change in Clause 5.2
End of Document

CHANGE REQUEST

⌘ **32.111-3 CR 029** ⌘ rev **-** ⌘ Current version: **5.2.0** ⌘

For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.

Proposed change affects: UICC apps ME Radio Access Network Core Network

Title:	⌘ Correction of CORBA IDL Optional clearSystemId		
Source:	⌘ S5		
Work item code:	⌘ OAM-NIM	Date:	⌘ 28/02/2003
Category:	⌘ F	Release:	⌘ Rel-5
	Use <u>one</u> of the following categories: F (correction) A (corresponds to a correction in an earlier release) B (addition of feature), C (functional modification of feature) D (editorial modification) Detailed explanations of the above categories can be found in 3GPP TR 21.900 .	Use <u>one</u> of the following releases: 2 (GSM Phase 2) R96 (Release 1996) R97 (Release 1997) R98 (Release 1998) R99 (Release 1999) Rel-4 (Release 4) Rel-5 (Release 5) Rel-6 (Release 6)	

Reason for change:	⌘ Support for Optional clearSystemId		
Summary of change:	⌘ Update clear_system_id parameter of clear_alarms to be of type ManagedGenericIRPCConstDefs::StringTypeOpt		
Consequences if not approved:	⌘ clear_system_id parameter is not optional		

Clauses affected:	⌘ Annex A.2						
Other specs affected:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> </table> Other core specifications	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	⌘	
Y	N						
<input type="checkbox"/>	<input checked="" type="checkbox"/>						
	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> </table> Test specifications	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	⌘	
Y	N						
<input type="checkbox"/>	<input checked="" type="checkbox"/>						
	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> </table> O&M Specifications	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	⌘	
Y	N						
<input type="checkbox"/>	<input checked="" type="checkbox"/>						
Other comments:	⌘						

How to create CRs using this form:

Change in Clause Annex A.2

A.2 IDL specification (file name "AlarmIRPSystem.idl")

```

#ifndef AlarmIRPSystem_idl
#define AlarmIRPSystem_idl

#include "AlarmIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"

// This statement must appear after all include statements
#pragma prefix "3gppsa5.org"

/* ## Module: AlarmIRPSystem
This module contains the specification of all operations of Alarm IRP Agent.
=====
*/
module AlarmIRPSystem
{
    /*
    System fails to complete the operation. System can provide reason
    to qualify the exception. The semantics carried in reason
    is outside the scope of this IRP.
    */
    exception GetAlarmIRPVersions { string reason; };
    exception GetAlarmIRPOperationsProfile { string reason; };
    exception GetAlarmIRPNotificationProfile { string reason; };
    exception AcknowledgeAlarms { string reason; };
    exception UnacknowledgeAlarms { string reason; };
    exception CommentAlarms { string reason; };
    exception ClearAlarms { string reason; };
    exception GetAlarmList { string reason; };
    exception GetAlarmCount { string reason; };
    exception NextAlarmInformations { string reason; };

    /*
    The AlarmInformationIterator is used to iterate through a snapshot of
    Alarm Informations taken from the Alarm List when IRPManager invokes
    get_alarm_list. IRPManager uses it to pace the return of Alarm
    Informations.

    IRPAgent controls the life-cycle of the iterator. However, a destroy
    operation is provided to handle the case where IRPManager wants to stop
    the iteration procedure before reaching the last iteration.
    */
    interface AlarmInformationIterator
    {
        /*
        This method returns between 1 and "how_many" Alarm Informations. The
        IRPAgent may return less than "how_many" items even if there are more
        items to return. "how_many" must be non-zero. Return TRUE if there may
        be more Alarm Information to return. Return FALSE if there are no more
        Alarm Information to be returned.

        If FALSE is returned, the IRPAgent will automatically destroy the
        iterator.
        */
        boolean next_alarmInformations (
            in unsigned short how_many,
            out AlarmIRPConstDefs::AlarmInformationSeq alarm_informations
        )
        raises (NextAlarmInformations, ManagedGenericIRPSystem::InvalidParameter);

        /*
        This method destroys the iterator.
        */
        void destroy();
    };

    interface AlarmIRP
    {
        /*
        Return the list of all supported Alarm IRP versions.
        */
        ManagedGenericIRPConstDefs::VersionNumberSet get_alarm_IRP_versions (
        )
    }
}

```

```
raises (GetAlarmIRPVersions);

/*
Return the list of all supported operations and their supported
parameters for a specific Alarm IRP version.
*/
ManagedGenericIRPConstDefs::MethodList get_alarm_IRP_operations_profile (
    in ManagedGenericIRPConstDefs::VersionNumber alarm_irp_version
)
raises (GetAlarmIRPOperationsProfile,
    ManagedGenericIRPSystem::OperationNotSupported,
    ManagedGenericIRPSystem::InvalidParameter);

/*
Return the list of all supported notifications and their supported
parameters for a specific Alarm IRP version.
*/
ManagedGenericIRPConstDefs::MethodList get_alarm_IRP_notification_profile
(
    in ManagedGenericIRPConstDefs::VersionNumber alarm_irp_version
)
raises (GetAlarmIRPNotificationProfile,
    ManagedGenericIRPSystem::OperationNotSupported,
    ManagedGenericIRPSystem::InvalidParameter);

/*
Request to acknowledge one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal acknowledge_alarms (
    in AlarmIRPConstDefs::AlarmInformationIdAndSevSeq
    alarm_information_id_and_sev_list,
    in string ack_user_id,
    in ManagedGenericIRPConstDefs::StringTypeOpt ack_system_id,
    out AlarmIRPConstDefs::BadAcknowledgeAlarmInfoSeq
    bad_ack_alarm_info_list
)
raises (AcknowledgeAlarms, ManagedGenericIRPSystem::ParameterNotSupported,
    ManagedGenericIRPSystem::InvalidParameter);

/*
Request to remove acknowledgement information of one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal unacknowledge_alarms (
    in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
    in string ack_user_id,
    in ManagedGenericIRPConstDefs::StringTypeOpt ack_system_id,
    out AlarmIRPConstDefs::BadAlarmInformationIdSeq
    bad_alarm_information_id_list
)
raises (UnacknowledgeAlarms,
    ManagedGenericIRPSystem::OperationNotSupported,
    ManagedGenericIRPSystem::ParameterNotSupported,
    ManagedGenericIRPSystem::InvalidParameter);

/*
Make comment to one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal comment_alarms (
    in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
    in string comment_user_id,
    in ManagedGenericIRPConstDefs::StringTypeOpt comment_system_id,
    in string comment_text,
    out AlarmIRPConstDefs::BadAlarmInformationIdSeq
    bad_alarm_information_id_list
)
raises (CommentAlarms, ManagedGenericIRPSystem::OperationNotSupported,
    ManagedGenericIRPSystem::ParameterNotSupported,
    ManagedGenericIRPSystem::InvalidParameter);

/*
Request to clear one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal clear_alarms (
    in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
```

```
        in string clear_user_id,
        in ManagedGenericIRPConstDefs::StringTypeOpt string clear_system_id,
        out AlarmIRPConstDefs::BadAlarmInformationIdSeq
            bad_alarm_information_id_list
    )
    raises (ClearAlarms, ManagedGenericIRPSystem::ParameterNotSupported,
           ManagedGenericIRPSystem::InvalidParameter);

/*
This method returns Alarm Informations.
If flag is TRUE, all returned Alarm Informations shall be
in AlarmInformationSeq that contains 0 or more Alarm Informations.
Output parameter iter shall be useless.
If flag is FALSE, no Alarm Informations shall be in AlarmInformationSeq.
IRPAgent needs to use iter to retrieve them.
*/
AlarmIRPConstDefs::AlarmInformationSeq get_alarm_list (
    in ManagedGenericIRPConstDefs::StringTypeOpt filter,
    out boolean flag,
    out AlarmInformationIterator iter
)
raises (GetAlarmList, ManagedGenericIRPSystem::ParameterNotSupported,
       ManagedGenericIRPSystem::InvalidParameter);

/*
This method returns the count of Alarm Informations.
*/
void get_alarm_count (
    in ManagedGenericIRPConstDefs::StringTypeOpt filter,
    out unsigned long critical_count,
    out unsigned long major_count,
    out unsigned long minor_count,
    out unsigned long warning_count,
    out unsigned long indeterminate_count,
    out unsigned long cleared_count
)
raises (GetAlarmCount, ManagedGenericIRPSystem::OperationNotSupported,
       ManagedGenericIRPSystem::ParameterNotSupported,
       ManagedGenericIRPSystem::InvalidParameter);
};
#endif
```

End of Change in Annex A.2 End of Document
