Technical Specification Group Services and System Aspects **TSGS#13(01)0470**

Meeting #13, Beijing, China, 24-27 September 2001

Source: **SA5**

Title: **Rel-4 CR32.111-4 for upgrading R99 to Rel-4 (Telecommunication Management; Fault Management; Part 4: Alarm Integration Reference Point: CMIP Solution Set Version 1:1)**

Document for: **Approval**

Agenda Item: **7.5.3**

Doc-1st-Level	Doc-2nd-Level	Spec	CR	Rev	Phase	Subject	Cat	Version Current	Version-New	Workitem
SP-010470	S5-010442	32.111-4	001	1	Rel-4	Addition of features	B	3.1.1	4.0.0	OAM-FM

3GPP TSG-SA5 (Telecom Management) **S5-010442**
Meeting #21, Naperville, IL, USA 16 – 20 July 2001 **S5F010130**

CR-Form-v4

CHANGE REQUEST

⌘ **32.111-4** **CR** **001** ⌘ ev **1** ⌘ Current version: **3.1.1** ⌘

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

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network **X** Core Network ☐

3. Title:	⌘	Addition of features
Source:	⌘	SA5
Work item code:	⌘	OAM-FM
Date:	⌘	20/07/2001
Category:	⌘	**B**
Release:	⌘	REL-4

Use one 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 one 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)

4. Reason for change:	⌘	Reflect changes to Alarm IRP: IS Rel4 (32.111-2).
5. Summary of change:	⌘	Modify the textual descriptions according to the new structure of 32..111-2. Add optional setComment() and notifyComment(). Remove the use of Extended Event Type.. Support Partial Alarm List Rebuilt Add optional operations getAlarmIRPOperationProfile() and getAlarmIRPNotificationProfile()
6. Consequences if not approved:	⌘	There will be no CMIP SS that corresponds to the Alarm IRP: IS Rel4.

Clauses affected:	⌘	All clauses
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

Please note that the changes have been introduced in two steps, therefore we have attached the following files:

- **TS 32111-4 R4 Draft_1.doc**, containing the first set of changes with “old” and “new” text marked,
- **TS 32111-4 R4 Draft_2.doc**, containing the second set of changes with “old” and “new” text marked,
- **TS 32111-4 R4 Whole Clean Draft.doc**, containing all the changes without revision marks.

3G TS 32.111-4 ~~V3.1.1 (2000-09)~~

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Telecommunication Management;
Fault Management;
Part 4: Alarm Integration Reference Point:
CMIP Solution Set ~~Version 1:1~~
(Release ~~1999~~4)

Keywords

Fault Management, Alarms

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

Contents

Foreword

This Technical Specification (TS) has been produced by the 3rd Generation Partnership Project (3GPP).

The present document is part 4 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects, as identifies below:

Part 1: "3G Fault Management Requirements";

Part 2: "Alarm Integration Reference Point: Information Service";

Part 3: "Alarm Integration Reference Point: CORBA Solution Set ~~Version 1:1~~";

Part 4: "Alarm Integration Reference Point: CMIP Solution Set".

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document (3G TS 32.111 Part-4) defines the alarm integration reference point for the CMIP solution set. In detail:

- Clause 4 contains an introduction to some basic concepts of the CMIP interfaces.
- Clause 5 contains the GDMO definitions for the Alarm Management over the CMIP interfaces
- Clause 6 contains the ASN.1 definitions supporting the GDMO definitions provided in clause 5.

2 References

The following documents contain provisions, which through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

~~[1] 3G TS 32.101: "3G Telecom Management principles and high level requirements".~~

~~[2] 3G TS 32.102: "3G Telecom Management architecture".~~

~~[3]~~[1] 3G TS 32.~~106~~301-2: "Notification Integration Reference Point: Information Service".

~~[4]~~[2] ITU-T Recommendation X.710: "Common management information service definition for CCITT applications".

~~[5]~~[3] ITU-T Recommendation X.711: "Common management information protocol specification for CCITT applications".

~~[6]~~[4] ITU-T Recommendation X.721: "Information technology - Open Systems Interconnection - Structure of management information: Definition of management information".

~~[7] ITU-T Recommendation X.731: "Information technology - Open Systems Interconnection - Systems Management: State management function".~~

~~[8]~~[5] ITU-T Recommendation X.733: "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

~~[9]~~[6] ITU-T Recommendation X.734: "Information technology - Open Systems Interconnection - Systems Management: Event report management function".

~~[10]~~[7] ITU-T Recommendation Q.821: “Specification of System Signalling No. 7 Q3 Interface- Stage 2 and Stage 3 description for the Q3 interface - Alarm Surveillance”

~~[11]~~[8] 3G TS 32.111-1: “3G Fault Management”.

~~[12]~~[9] 3G TS 32.111-2: “Alarm Integration Reference Point: Information Service”.

~~[13] 3G TS 32.111-3: “Alarm Integration Reference Point: CORBA Solution Set Version 1:1”.~~

~~[14]~~[10] 3G TS 32.301~~106~~-4: "Notification Integration Reference Point: CMIP Solution Set".

[11] 3G TS 32.112-2: " Generic IRP Management: InformationService".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions defined in 3G TS 32.111-1 ~~[11]~~[8] apply.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation number 1
CCITT	The International Telegraph and Telephone Consultative Committee
CM	Configuration Management
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
EFD	Event Forwarding Discriminator
EM	Element Manager
FTAM	File Transfer Access and Management
GDMO	Guidelines for the Defifition of Managed Objects
IOC	Information Object Class
IRP	Integration Reference Point
Itf-N	Interface N (between NM and EM/NE)
ITU-T	International Telecommunication Union – Telecommunications
M	Mandatory
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NM	Network Manager
NMC	Network Management Centre
O	Optional
OS	Operations System
TMN	Telecommunications Management Network

4 Basic aspects

The present document provides all the GDMO and ASN.1 definitions necessary to implement the Alarm IRP Information Service [?] for the CMIP interface. 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 modeled as GDMO “Managed Object Classes” (MOC) defined specifically for alarm management, the Operations are modeled as GDMO “Actions” of a MOC ~~defined specifically for alarm management~~ while the Notifications are modeled 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 (3G TS 32.301~~106~~-4 ~~[14]~~[10]).

4.3 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 ~~[8]~~[5] and ITU-T Recommendation Q.821 ~~[10]~~[7].

4.4 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 unacknowledge alarms;

– ITU-T ~~Recommendation~~ X.721 ~~[6]~~[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.

4.5 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.

4.5~~5~~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 or NULL, all alarm notifications are forwarded to the Manager through this EFD, 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 ~~[9]~~[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:

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 3G TS 32.301~~106~~-4 ~~[14]~~[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).

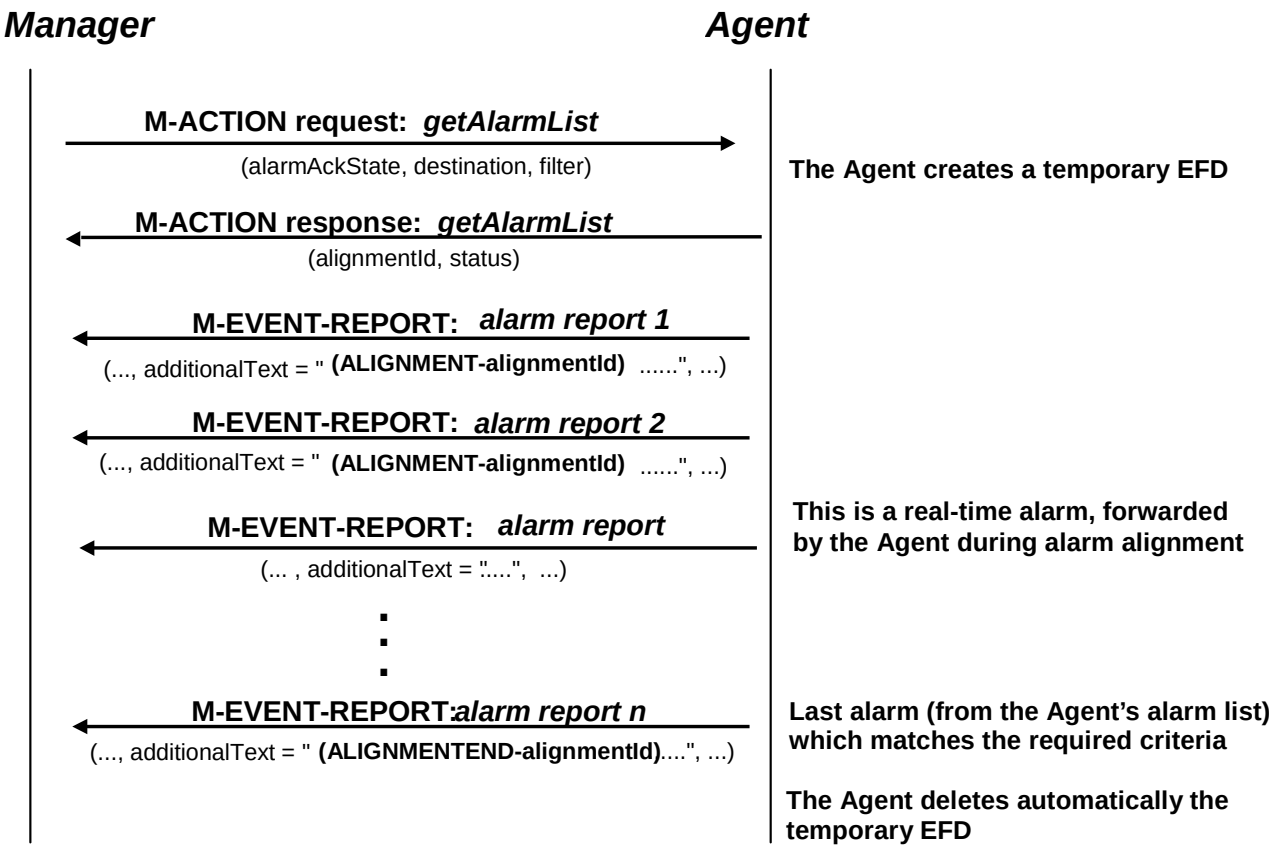

Figure 1: Alignment arrow diagram

Figure 2 shows the handling of a „real-time“ alarm notification (occured during the execution of the *getAlarmList* operation), which is forwarded by the Agent (according to ITU-T Recommendation X.734 ~~[9]~~[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.

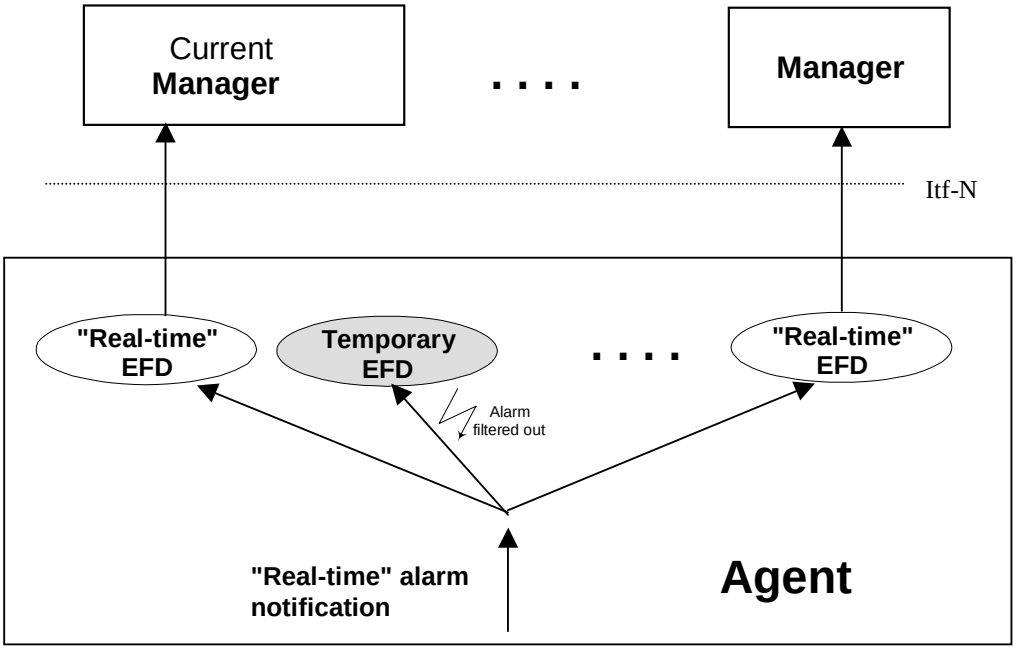

Figure 2: Treatment of “real time” alarms

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.

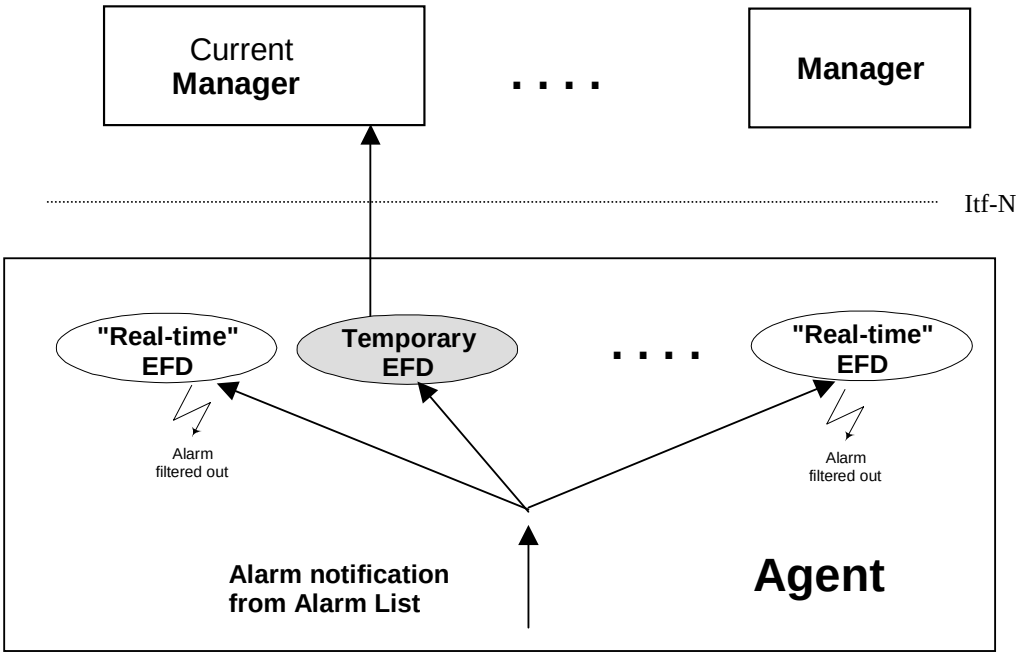

Figure 3: Treatment of “alignent” alarms

4.7~~6~~ Mapping

The semantics of the Alarm IRP is defined in 3G TS 32.111-2 ~~[12]~~[9]. The definitions of the management information defined there are independent of any implementation technology and protocol. This section maps these protocol-independent definitions onto the equivalences of the CMIP solution set of Alarm IRP.

4.7.1 Mapping of IOC and Interfaces

For this Alarm IRP CMIP Solution Sets, the Information Object Classes (IOC) and the Interfaces defined in 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 [9].

4.~~6~~7.2~~1~~ Mapping of Interface/~~-~~Operations

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

Table 1: Mapping of Operations

~~Operations of Information Services of the Alarm IRP~~	~~CMIP SS Equivalents solution set for the Alarm IRP~~	~~Qualifier~~
~~acknowledgeAlarms~~	~~acknowledgeAlarms~~	~~M~~

~~getAlarmCount~~	~~getAlarmCount~~	~~O~~
~~getAlarmList~~	~~getAlarmList~~	~~M~~
~~getAlarmIRPVersion~~	~~getAlarmIRPVersion~~	~~M~~
~~unacknowledgeAlarms~~	~~unacknowledgeAlarms~~	~~O~~

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
GenericIRPVersionOperation/getIRPVersion	getAlarmIRPVersion	M
GenericIRPProfileOperation/getNotificationProfile	getNotificationProfile	O
GenericIRPProfileOperation/getOperationProfile	getOperationProfile	O

NOTE: the Interfaces GenericIRPVersionOperation and GenericIRPProfileOperation are defined in [11]

4.76.32 Mapping of Parameters of each operation

The tables in the following subclauses show the parameters of each operations defined in the IS ~~described~~ 3G TS 32.111-2 ~~[12]~~[9] and their equivalents in this CMIP SS.

~~The parameters of the IS operations are mapped, in the CMIP SS equivalents.~~

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: Mapping of parameters of 'acknowledgementAlarms'

Operation parameters of Information Services	IN/OUT	CMIP equivalences	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

Table 3: Mapping of Parameters of 'getAlarmCount'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	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: Mapping of Parameters of 'getAlarmList'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
--		destination (input) - see NOTE 1	M
alarmInformationList	OUT	(sequence of alarm notifications) (see Clause 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: Mapping of Parameters of 'getAlarmIRPVersion'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
~~versionNumberList~~versionNumberSet	OUT	versionNumberList	M
status	OUT	status	M

Table 4: Mapping of Parameters of 'getOperationProfile'

Operation parameters of the Information Services.	IN/OUT	CMIP Solution Set equivalences	Qualifier
irpVersion	IN	irpVersionNumber	M
operationNameProfile	OUT	operationNameProfile	M
operationParameterProfile	OUT	operationParameterProfile	M
status	OUT	status	M

Table 4: Mapping of Parameters of 'getNotificatioProfile'

Operation parameters of the Information Services.	IN/OUT	CMIP Solution Set equivalences	Qualifier
irpVersion	IN	irpVersionNumber	M
notificationNameProfile	OUT	notificationNameProfile	M
notificationParameterProfile	OUT	notificationParameterProfile	M
status	OUT	status	M

Table 4: Mapping of Parameters of 'setComment'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M

commentUserId	IN	commentUserId	M
commentSystemId	IN	commentSystemId	O
commentText	IN	commentText	M
badAlarmInformationReferenceList	OUT	badAlarmReferenceList	M
Status	OUT	status	M

Table 6: Mapping of Parameters of 'unacknowledgeAlarms'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

4.~~6~~7.~~4~~3 Mapping of Notifications

Table 7 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 7: 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 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	M
notifyChangedAlarm	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	O
notifyClearedAlarm	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	M
notifyAckStateChanged	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	M
notifyAlarmListRebuilt	alarmListRebuilt	M
notifyComments	notifyComments	O

4.~~6~~7.~~5~~4 Mapping of Parameters of each notification

Table 8 and table 9 show the parameters of each notification defined in the Information Service described in 3G TS 32.111-2 ~~[12]~~[9] and their equivalence in this CMIP SS.

The input parameters of the Information Service notifications are mapped, in the CMIP SS, onto the “event information”.

Table 8: Mapping of Parameters of 'notifyNewAlarm', 'notifyClearedAlarm' and 'notifyAckStateChanged'

Notification parameters of Information Services	CMIP equivalences	Qualifier
--	notificationIdentifier (Note 1)	M
probableCause	probableCause	M
specificProblems	specificProblems	O
perceivedSeverity	perceivedSeverity	M
backedUpStatus	backedUpStatus	O
backUpObject	backUpObject	O
trendIndication	trendIndication	O
thresholdInfo	thresholdInfo	O
correlatedNotifications	correlatedNotifications	O
stateChangeDefinition	stateChangeDefinition	O
monitoredAttributes	monitoredAttributes	O
proposedRepairActions	proposedRepairActions	O
additionalText	additionalText	O
additionalInformation	additionalInformation	(Note 2)
NOTE 1: notificationIdentifier is a parameter of the Notification Header defined in 3G TS 32.~~106-2~~301-2 ~~[3]~~[1]. NOTE 2: See qualification information in 3G TS 32.111-2 ~~[12]~~[9], Table 13: Parameter-Attributes of `alarmInformationBody`.		

Table 9: Mapping of Parameters of 'notifyAalarmListRebuilt'

Notification parameters of Information Services	CMIP equivalents	Qualifier
	notificationIdentifier (see Note)	
reason	reason	M
NOTE: notificationIdentifier is a parameter of the Notification Header defined in 3G TS 32.~~106-2~~301-2 ~~[3]~~[1].		

Table 9: Mapping of Parameters of 'notifyComments'

Notification parameters of Information Services	CMIP equivalents	Qualifier
objectClass	alarmedObjectClass	M
objectInstance	alarmedObjectInstance	M
notificationId	notificationIdentifier	M
eventTime	alarmEventTime	M
systemDN	--	
notificationType	eventType	M
alarmType	alarmType	M
probableCause	alarmProbableCause	M
perceivedSeverity	alarmPerceivedSeverity	M
`comments`	`comments`	M
alarmId	--	

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,
        alarmCommentPackage,
        alarmIRPVersionPackage,
        alarmProfilePackage;
REGISTERED AS { ts32-111AlarmObjectClass 1};
```

5.2 Packages

5.2.1 alarmControlBasicPackage

```
alarmControlBasicPackage PACKAGE
    BEHAVIOUR
        alarmControlBasicPackageBehaviour;
    ATTRIBUTES
        alarmControlId          GET,
        alarmsCountSummary      GET;
    ACTIONS
        getAlarmCount,
        getAlarmList;
    NOTIFICATIONS
        alarmListRebuilt;
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 'getAlarmCount' is the means, for the Manager, to ask the number of currently available alarms in the Agent according to the specification in the action request.

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 alarmAcknowledgementPackage

alarmAcknowledgementPackage **PACKAGE**
BEHAVIOUR
alarmAcknowledgementPackageBehaviour;
ACTIONS
acknowledgeAlarms,
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 2};

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 action 'unacknowledgeAlarms' allows the NM operator to unacknowledge one or several alarms previously acknowledged by himself.

The ITU-T Recommendation X.721 ~~[6]~~[4] compliant alarm notifications are sent by the Agent to the Manager to inform that one alarm has been acknowledged or unacknowledged. The acknowledgement related information is carried in the *additionalInformation* attribute.";

5.2.3 alarmCommentPackage

alarmCommentPackage **PACKAGE**
BEHAVIOUR
alarmCommentPackageBehaviour;
ACTIONS
setComment;
NOTIFICATIONS
notifyComments;
REGISTERED AS { ts32-111AlarmPackage 3};

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.";

5.2.43 alarmIRPVersionPackage

alarmIRPVersionPackage **PACKAGE**
BEHAVIOUR
alarmIRPVersionPackageBehaviour;
ATTRIBUTES
supportedAlarmIRPVersions GET;
ACTIONS
getAlarmIRPVersion;
REGISTERED AS { ts32-111AlarmPackage 34};

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";

5.2.5 alarmProfilePackage

alarmProfilePackage PACKAGE
BEHAVIOUR
alarmProfilePackageBehaviour;
ACTIONS
getOperationProfile,
getNotificationProfile;

REGISTERED AS { ts32-111AlarmPackage 5};

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.";

5.3 Actions

5.3.1 acknowledgeAlarms (M)

acknowledgeAlarms **ACTION**
BEHAVIOUR
acknowledgeAlarmsBehaviour;
MODE
CONFIRMED;
WITH INFORMATION SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AckOrUnackAlarms;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AckOrUnackAlarmsReply;
REGISTERED AS { ts32-111AlarmAction 1};

acknowledgeAlarmsBehaviour **BEHAVIOUR**
DEFINED AS

”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-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmCount;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmCountReply;
REGISTERED AS { ts32-111AlarmAction 2};

getAlarmCountBehaviour **BEHAVIOUR**
DEFINED AS

”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-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmList;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~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 ~~[6]~~[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 (M)

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 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.”;

5.3.54 getAlarmIRPVersion (M)

getAlarmIRPVersion **ACTION**
BEHAVIOUR
getAlarmIRPVersionBehaviour;
MODE
CONFIRMED;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmIRPVersionReply;
REGISTERED AS { ts32-111AlarmAction 54};

getAlarmIRPVersionBehaviour **BEHAVIOUR**
DEFINED AS

”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 getNotificationProfile (O)

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

getNotificationProfileBehaviour **BEHAVIOUR**

DEFINED AS

"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 getOperationProfile (O)

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

getOperationProfileBehaviour **BEHAVIOUR**

DEFINED AS

"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.~~5~~8 unacknowledgeAlarms (O)

unacknowledgeAlarms **ACTION**
BEHAVIOUR
unacknowledgeAlarmsBehaviour;
MODE
CONFIRMED;
WITH INFORMATION SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AckOrUnackAlarms;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AckOrUnackAlarmsReply;
REGISTERED AS { ts32-111AlarmAction 8~~5~~};
unacknowledgeAlarmsBehaviour **BEHAVIOUR**
DEFINED AS

”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 unacknowledge it later.

- *ackSystemId*
  It indicates the management system where the acknowledgment is triggered. It may have also the value NULL. Note that the unacknowledgement 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 unknowledgement 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. “;

5.4 Notifications

5.4.1 alarmListRebuilt (M)

alarmListRebuilt **NOTIFICATION**
BEHAVIOUR
alarmListRebuiltBehaviour;
WITH INFORMATION SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmListRebuiltInfo;
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.

- *reason*

  The parameter indicates the reason for alarm list rebuilding (if applicable).”;

5.4.2 notifyComments (M)

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 a comment has been associated to one or more alarms.

The 'Event Information' field contains the following data:

- alarmedObjectClass: defined in ITU-T X.710 [2] and X.711[3]
- alarmedObjectInstance: defined in ITU-T X.710 [2] and X.711[3]
- alarmEventTime: defined in ITU-T X.721

- alarmType: the eventType of the alarm to which this comment is associated.
- alarmProbableCause: defined in ITU-T X.721
- alarmPerceivedSeverity: defined in ITU-T X.721
- comments: the text of the comment.

";

5.5 Attributes

5.5.1 alarmControlId

alarmControlId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~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-AlarmAsn1TypeModule~~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-AlarmAsn1TypeModule~~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-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~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-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .~~Ack~~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-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~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-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
TS32-111-4~~AlarmAsn1~~TypeModule.~~Ack~~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.”;

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)}

~~TS32-111-AlarmAsn1TypeModule--{ObjectIdentifierValue} to be defined~~
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 action(10)}

~~baseNode3gpp OBJECT IDENTIFIER ::= {baseNode(1)} -- to be defined~~
~~ts32-111Alarm OBJECT IDENTIFIER ::= { baseNode3gpp ts32-111Alarm(1)}~~
~~ts32-111AlarmObjectClass OBJECT IDENTIFIER ::= {ts32-111Alarm managedObjectClass(3)}~~
~~ts32-111AlarmPackage OBJECT IDENTIFIER ::= {ts32-111Alarm package(4)}~~
~~ts32-111AlarmParameter OBJECT IDENTIFIER ::= {ts32-111Alarm parameter(5)}~~
~~ts32-111AlarmAttribute OBJECT IDENTIFIER ::= {ts32-111Alarm attribute(7)}~~
~~ts32-111AlarmAction OBJECT IDENTIFIER ::= {ts32-111Alarm action(9)}~~
~~ts32-111AlarmNotification OBJECT IDENTIFIER ::= {ts32-111Alarm notification(10)}~~

-- Start of 3GPP SA5 own definitions
AckErrorList ::= SET OF ErrorInfo
AlarmReference ::= SEQUENCE
{
moi ObjectInstance OPTIONAL, -- absent if scope of uniquness of notificationId is across IRPAgent
notificationIdentifier NotificationIdentifier
}

```
AckOrUnackAlarms ::= SEQUENCE
    {
    alarmReferenceList  SET OF AlarmReference,   -- ITU-T X.721
    ackUserId           AckUserId,
    ackSystemId         AckSystemId OPTIONAL
    }
AckOrUnackAlarmsReply ::= SEQUENCE
    {
    status              ErrorCauses,
    errorAlarmReferenceList     AckErrorList
    }
AckState ::= ENUMERATED
    {
    acknowledged        (0),
    unacknowledged (1)
    }
AckSystemId ::= GraphicString
AckTime ::= GeneralizedTime
AckUserId ::= GraphicString
AlarmChoice ::= ENUMERATED
    {
    allAlarms                   (0),
    allActiveAlarms             (1),
    allActiveAndAckAlarms       (2),
    allActiveAndUnackAlarms     (3),
    allClearedAndUnackAlarms    (4)
    }
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
    reason              ErrorCauses
    }
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)
    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

NotifyComments ::= SEQUENCE
{
alarmedObjectClass ObjectClass,
alarmedObjectInstance ObjectInstance,
alarmEventTime EventTime,
alarmType EventTypeId,
alarmProbableCause ProbableCause,
alarmPerceivedSeverity PerceivedSeverity,
comments SET OF Comment
}

OperationList ::= SET OF OperationName

OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

SetComment ::= SEQUENCE
{
alarmReferenceList SET OF AlarmReference,
commentUserId UserId,
commentSystemId SystemId,
commentText Comment
}
SetCommentReply ::= SEQUENCE
{
badAlarmReferenceList SET OF ErrorInfo,
status ErrorCauses
}
SystemId ::= GraphicString

SupportedAlarmIRPVersions ::= SET OF IRPVersionNumber
UserId ::= GraphicString

END -- of module ~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule

Annex A (informative): Change history

Change history					
TSG SA#	Version	CR	Tdoc SA	New Version	Subject/Comment
S_07	2.0.0	-	SP-000012	3.0.0	Approved at TSG SA #7 and placed under Change Control
Mar 2000	3.0.0			3.0.1	cosmetic
S_08	3.0.1	005	SP-000254	3.1.0	Split of TS - Part 4: Alarm Integration Reference Point (IRP): CMIP Solution Set (SS)
Sep 2000	3.1.0			3.1.1	Cosmetic

3GPP TS 32.111-4 ~~V3.1.1 (2000-09)~~

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Telecommunication Management;
Fault Management;
Part 4: Alarm Integration Reference Point: CMIP Solution Set
(Release 4)

Keywords
Fault Management, Alarms

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Specification (TS) has been produced by the 3rd Generation Partnership Project (3GPP).

The present document is part 4 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects, as identifies below:

Part 1: "3G Fault Management Requirements";

Part 2: "Alarm Integration Reference Point: Information Service";

Part 3: "Alarm Integration Reference Point: CORBA Solution Set";

Part 4: "Alarm Integration Reference Point: CMIP Solution Set".

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document (3G TS 32.111 Part-4) defines the alarm integration reference point for the CMIP solution set. In detail:

- Clause 4 contains an introduction to some basic concepts of the CMIP interfaces.
- Clause 5 contains the GDMO definitions for the Alarm Management over the CMIP interfaces
- Clause 6 contains the ASN.1 definitions supporting the GDMO definitions provided in clause 5.

2 References

The following documents contain provisions, which through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] 3G TS 32.302~~1-2~~: "Notification Integration Reference Point: Information Service".

[2] ITU-T Recommendation X.710: "Common management information service definition for CCITT applications".

[3] ITU-T Recommendation X.711: "Common management information protocol specification for CCITT applications".

[4] ITU-T Recommendation X.721: "Information technology - Open Systems Interconnection - Structure of management information: Definition of management information".

[5] ITU-T Recommendation X.733: "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

[6] ITU-T Recommendation X.734: "Information technology - Open Systems Interconnection - Systems Management: Event report management function".

[7] ITU-T Recommendation Q.821: “Specification of System Signalling No. 7 Q3 Interface- Stage 2 and Stage 3 description for the Q3 interface - Alarm Surveillance”

[8] 3G TS 32.111-1: “3G Fault Management”.

[9] 3G TS 32.111-2: “Alarm Integration Reference Point: Information Service”.

[10] 3G TS 32.304~~1-4~~: "Notification Integration Reference Point: CMIP Solution Set".

[11] 3G TS 32.312~~112-2~~: " Generic IRP Management: InformationService".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions defined in 3G TS 32.111-1 [8] apply.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation number 1
CCITT	The International Telegraph and Telephone Consultative Committee
CM	Configuration Management
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
EFD	Event Forwarding Discriminator
EM	Element Manager
FTAM	File Transfer Access and Management
GDMO	Guidelines for the Defifition of Managed Objects
IOC	Information Object Class
IRP	Integration Reference Point
Itf-N	Interface N (between NM and EM/NE)
ITU-T	International Telecommunication Union – Telecommunications
M	Mandatory
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NM	Network Manager
NMC	Network Management Centre
O	Optional
OS	Operations System
TMN	Telecommunications Management Network

4 Basic aspects

The present document provides all the GDMO and ASN.1 definitions necessary to implement the Alarm IRP Information Service [~~?~~9] for the CMIP interface. 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 (3G TS 32.30~~41-4~~ [10]).

4.1 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 [4] and [5] are used.

4.2 Reporting changed alarms

Although in the Alarm IRP Information Service [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 [4] and [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.

4.3 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].

4.4 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 unacknowledge alarms;

– ITU-T 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.

4.5 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.

4.6 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 or NULL, all alarm notifications are forwarded to the Manager through this EFD, 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:

  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 3G TS 32.3041~~-4~~ [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).

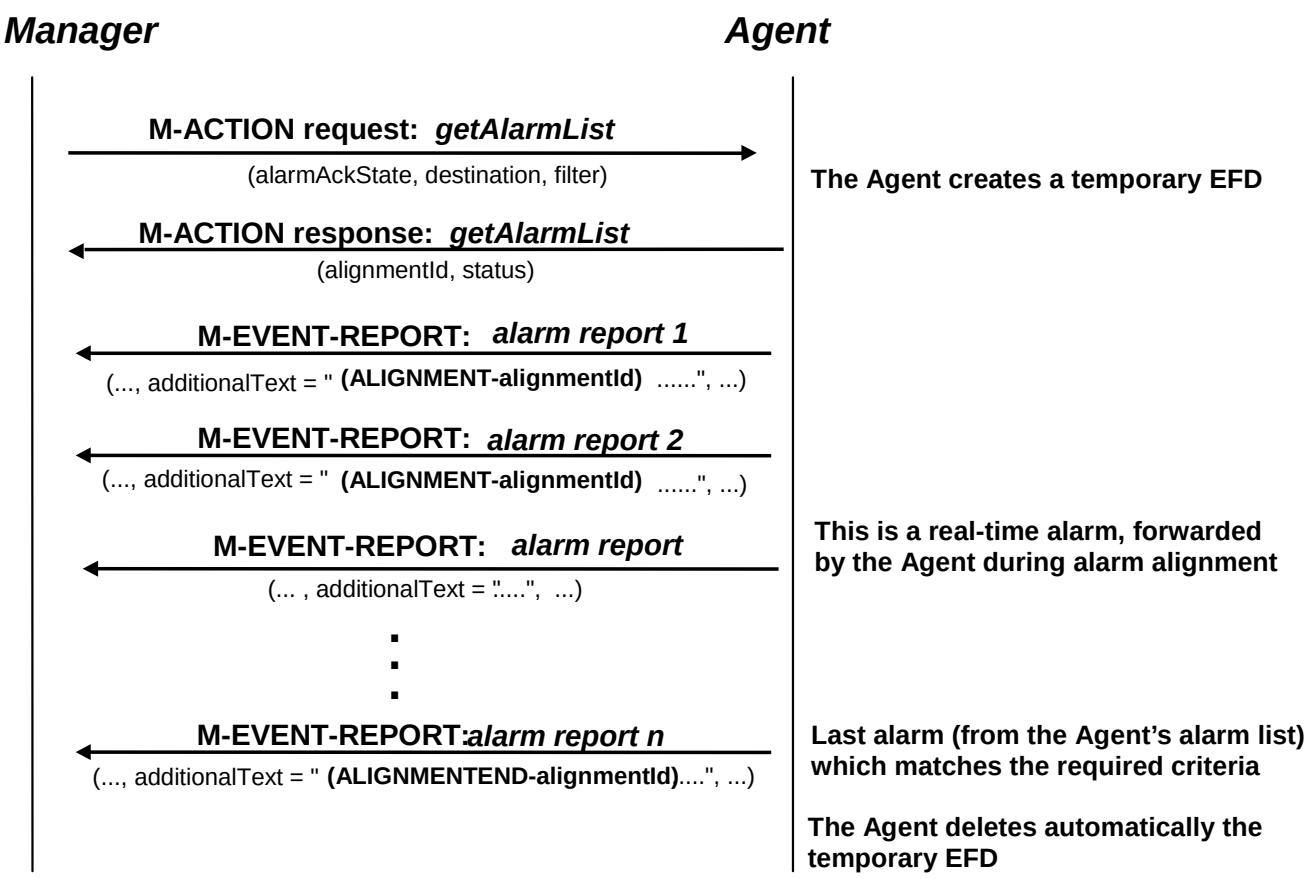

Figure 1: Alignment arrow diagram

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.

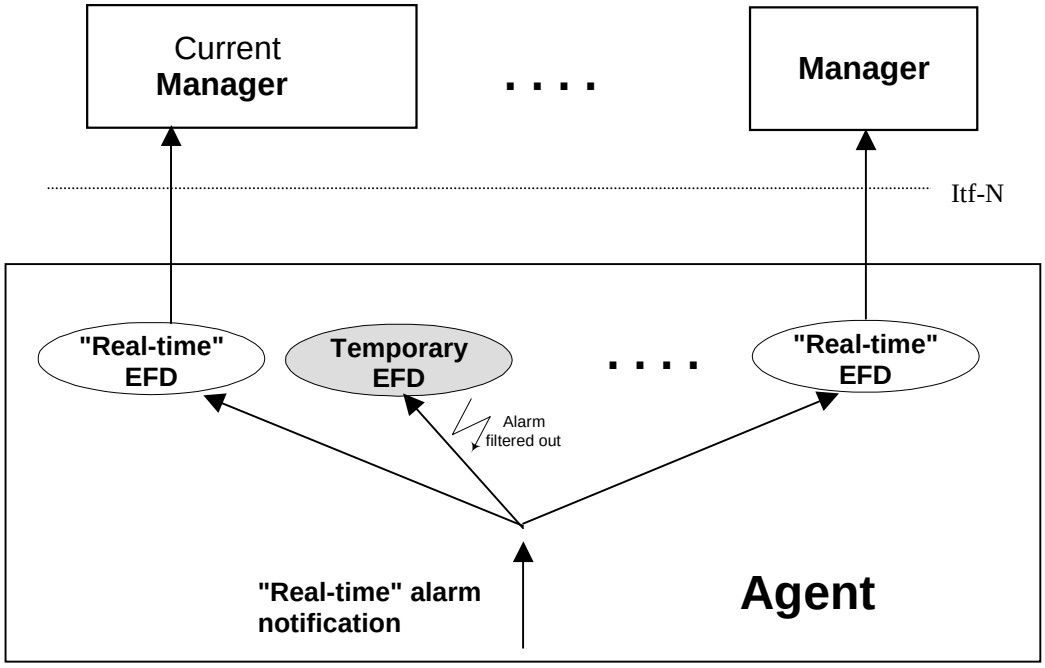

Figure 2: Treatment of “real time” alarms

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.

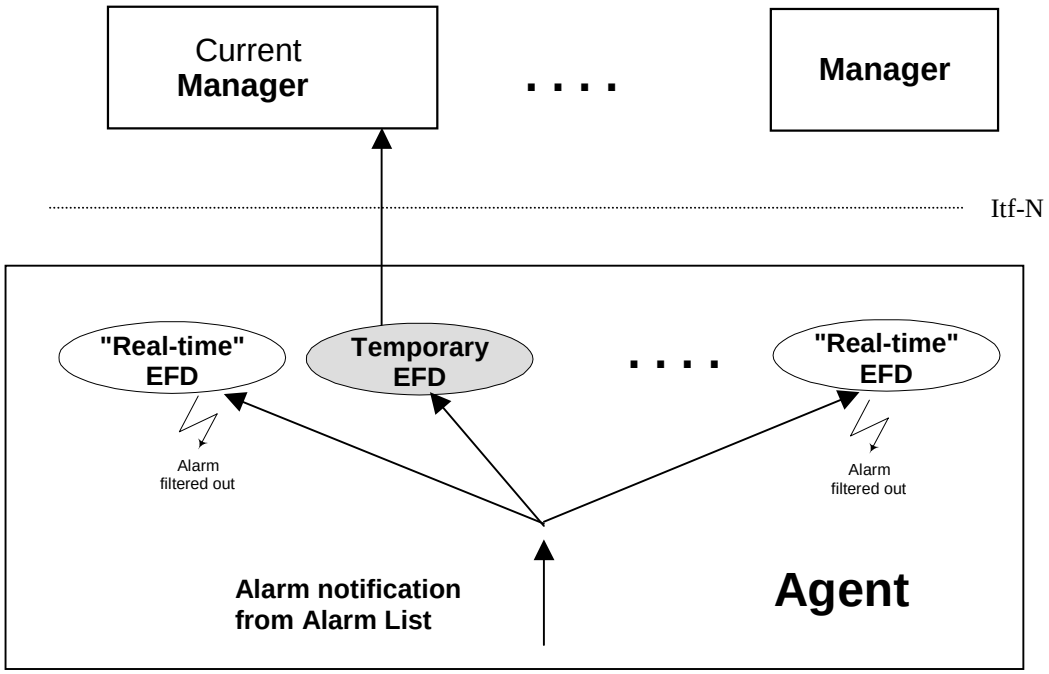

Figure 3: Treatment of “alignment” alarms

4.7 Mapping

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

4.7.1 Mapping of IOC and Interfaces

For this Alarm IRP CMIP Solution Sets, the Information Object Classes (IOC) and the Interfaces defined in 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 [9].

4.7.2 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
GenericIRPVersionOperation/getIRPVersion	getAlarmIRPVersion	M

GenericIRPProfileOperation/getNotificationProfile	getNotificationProfile	O
GenericIRPProfileOperation/getOperationProfile	getOperationProfile	O

NOTE: the Interfaces GenericIRPVersionOperation and GenericIRPProfileOperation are defined in [11]

4.7.3 Mapping of Parameters of each operation

The tables in the following subclauses show the parameters of each operations defined in the IS 3G 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: Mapping of parameters of ‘acknowledgementAlarms’

Operation parameters of Information Services	IN/OUT	CMIP equivalences	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

Table 3: Mapping of Parameters of ‘getAlarmCount’

Operation parameters of Information Services	IN/OUT	CMIP equivalents	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: Mapping of Parameters of ‘getAlarmList’

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
--		destination (input) - see NOTE 1	M
alarmInformationList	OUT	(sequence of alarm notifications) (see Clause 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: Mapping of Parameters of 'getAlarmIRPVersion'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
versionNumberSet	OUT	versionNumberList	M
status	OUT	status	M

Table 64: Mapping of Parameters of 'getOperationProfile'

Operation parameters of the Information Services.	IN/OUT	CMIP Solution Set equivalences	Qualifier
irpVersion	IN	irpVersionNumber	M
operationNameProfile	OUT	operationNameProfile	M
operationParameterProfile	OUT	operationParameterProfile	M
status	OUT	status	M

Table 74: Mapping of Parameters of 'getNotificatioProfile'

Operation parameters of the Information Services.	IN/OUT	CMIP Solution Set equivalences	Qualifier
irpVersion	IN	irpVersionNumber	M
notificationNameProfile	OUT	notificationNameProfile	M
notificationParameterProfile	OUT	notificationParameterProfile	M
status	OUT	status	M

Table 84: Mapping of Parameters of 'setComment'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
commentUserId	IN	commentUserId	M
commentSystemId	IN	commentSystemId	O
commentText	IN	commentText	M
badAlarmInformationReferenceList	OUT	badAlarmReferenceList	M
Status	OUT	status	M

Table 96: Mapping of Parameters of 'unacknowledgeAlarms'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

4.7.4 Mapping of Notifications

Table 710 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 107: 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 qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	M
notifyChangedAlarm	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	O
notifyClearedAlarm	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	M
notifyAckStateChanged	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	M
notifyAlarmListRebuilt	alarmListRebuilt		M
notifyComments	notifyComments		O

4.7.5 Mapping of Parameters of each notification

The notifications defined in [9] (Alarm IRP: Information Services) have a set of parameters that are common to all the notifications (see [10]: Notification IRP:CMIP Solution Set). Such common set of parameters are:

ManagedObjectClass, ManagedObjectInstance, EventTime, NotificationType, NotificationId.

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 [2,3]). The argument of m-EventReport OPERATION is defined in [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/ASN1 definitions.

Therefore the first four parameters of the notification header are mapped to the first four fields of the EventReportArgument. The fifth parameter of the notification header is mapped to the eventinfo field of the EventReportArgument, together with all the other (not common) parameters of the notification.

In the following tables, for the notifications defined in [9], all the parameters (but the common ones) are mapped to their corresponding elements of the CMIP SS notifications.

The parameter SystemDN defined in [9] (Alarm IRP: Information Services) is not used in this CMIP Solution Set.

~~Table 8 and table 9 show the parameters of each notification defined in the Information Service described in 3G TS 32.111-2 [9] and their equivalence in this CMIP SS.~~

~~The input parameters of the Information Service notifications are mapped, in the CMIP SS, onto the "event information".~~

Table 118: Mapping of Parameters of 'notifyNewAlarm' and, 'notifyClearedAlarm' and 'notifyAckStateChanged'

Notification parameters of Information Services	CMIP Notification equivalences	Qualifier
notificationId--	notificationIdentifier (Note 1)	MO
probableCause	probableCause	M
specificProblems	specificProblems	O
perceivedSeverity	perceivedSeverity	M
backedUpStatus	backedUpStatus	O
backUpObject	backUpObject	O
trendIndication	trendIndication	O
thresholdInfo	thresholdInfo	O
correlatedNotifications	correlatedNotifications	O
stateChangeDefinition	stateChangeDefinition	O
monitoredAttributes	monitoredAttributes	O
proposedRepairActions	proposedRepairActions	O
additionalText	additionalText	O
additionalInformation	additionalInformation (Note 2)	O (Note 2)
alarmId	-- (Note 2)	

NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3G TS 32.3021-2 [1].

NOTE 2: See qualification information in 3G TS 32.111-2 [9], Table 13: Parameter-Attributes of `alarmInformationBody`.

NOTE 2: In the CMIP Solution Set the alarmId is not used. In the CMIP Solution Set the alarm notifications are univocally identified by means of notificationIdentifier and managedObjectInstance.

Table 12: Mapping of Parameters of 'notifyAckStateChanged'

Notification parameters of Information Services	CMIP Notification equivalences	Qualifier
notificationId	notificationIdentifier (Note 1)	O
probableCause	probableCause	M
specificProblems	specificProblems	O
perceivedSeverity	perceivedSeverity	M
alarmId	-- (Note-2)	
ackTime	additionalInformation	M
ackState		M
ackUserId		M
ackSystemId		O

NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3G TS 32.302.

NOTE 2: In the CMIP Solution Set the alarmId is not used. In the CMIP Solution Set the alarm notifications are univocally identified by means of notificationIdentifier and managedObjectInstance.

Table 139: Mapping of Parameters of 'notifyAlarmListRebuilt'

Notification parameters of Information Services	CMIP Notification equivalents	Qualifier
notificationId	notificationIdentifier (see Note 1)	O
reason	reason	M
objectClass	rebuiltObjectClass	M
objectInstance	rebuiltObjectInstance	M

NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3G TS 32.3021-2 [1].

Table 149: Mapping of Parameters of 'notifyComments'

Notification parameters of Information Services	CMIP Notification equivalents	Qualifier
notificationId	notificationIdentifier (Note 1)	O
objectClass	alarmedObjectClass	M
objectInstance	alarmedObjectInstance	M
notificationId	notificationIdentifier	M
eventTime	alarmEventTime	M
~~systemDN~~	~~--~~	
notificationType	eventType	M
alarmType	alarmType	M
probableCause	alarmProbableCause	M
perceivedSeverity	alarmPerceivedSeverity	M
`comments`	`comments`	M
alarmId	-- (Note 2)	
NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3G TS 32.302. NOTE 2: In the CMIP Solution Set the alarmId is not used. In the CMIP Solution Set the alarm notifications are univocally identified by means of notificationIdentifier and managedObjectInstance.		

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,
        ~~alarmCommentPackage,~~
        alarmIRPVersionPackage;
        ~~alarmProfilePackage;~~
    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 ";
    REGISTERED AS { ts32-111AlarmObjectClass 1};
```

5.2 Packages

5.2.1 alarmControlBasicPackage

```
alarmControlBasicPackage PACKAGE
    BEHAVIOUR
        alarmControlBasicPackageBehaviour;
    ATTRIBUTES
        alarmControlId          GET,
        alarmsCountSummary      GET;
```

ACTIONS
~~getAlarmCount,~~
getAlarmList;
NOTIFICATIONS
alarmListRebuilt;
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 'getAlarmCount' is the means, for the Manager, to ask the number of currently available alarms in the Agent according to the specification in the action request.~~

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~~2~~ alarmAcknowledgementPackage

alarmAcknowledgementPackage **PACKAGE**
BEHAVIOUR
alarmAcknowledgementPackageBehaviour;
ACTIONS
acknowledgeAlarms~~;~~,
~~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 3~~2~~};

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 action 'unacknowledgeAlarms' allows the NM operator to unacknowledge one or several alarms previously acknowledged by himself.~~

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 ~~or unacknowledged~~. The acknowledgement related information is carried in the *additionalInformation* attribute.";

5.2.4 alarmUnacknowledgementPackage

alarmUnacknowledgementPackage **PACKAGE**
BEHAVIOUR
alarmUnacknowledgementPackageBehaviour;
ACTIONS
unacknowledgeAlarms;
REGISTERED AS { ts32-111AlarmPackage 4};

alarmUnacknowledgementPackageBehaviour **BEHAVIOUR**
DEFINED AS

"This package has been defined to provide the Manager with the capability to unacknowledge alarms.

The action 'unacknowledgeAlarms' allows the NM operator to unacknowledge 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~~3 alarmCommentPackage

alarmCommentPackage **PACKAGE**
BEHAVIOUR
alarmCommentPackageBehaviour;
ACTIONS
setComment;
NOTIFICATIONS
notifyComments;
REGISTERED AS { ts32-111AlarmPackage ~~3~~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~~4 alarmIRPVersionPackage

alarmIRPVersionPackage **PACKAGE**
BEHAVIOUR
alarmIRPVersionPackageBehaviour;
ATTRIBUTES
supportedAlarmIRPVersions GET;
ACTIONS
getAlarmIRPVersion;

REGISTERED AS { ts32-111AlarmPackage 46};

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";

5.2.75 alarmProfilePackage

alarmProfilePackage PACKAGE
BEHAVIOUR
alarmProfilePackageBehaviour;
ACTIONS
getOperationProfile,
getNotificationProfile;

REGISTERED AS { ts32-111AlarmPackage 57};

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.";

5.3 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

”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 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 (M)

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 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 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 getNotificationProfile (O)

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

getNotificationProfileBehaviour **BEHAVIOUR**

DEFINED AS

"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 getOperationProfile (O)

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

getOperationProfileBehaviour **BEHAVIOUR**

DEFINED AS

"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

”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 unacknowledge it later.

- *ackSystemId*
  It indicates the management system where the acknowledgment is triggered. It may have also the value NULL. Note that the unacknowledgement 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 unknowledgement 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. “;

5.4 Notifications

5.4.1 alarmListRebuilt (M)

alarmListRebuilt **NOTIFICATION**
BEHAVIOUR
alarmListRebuiltBehaviour;
WITH INFORMATION SYNTAX
TS32-111-4TypeModule .AlarmListRebuiltInfo;
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).”;

5.4.2 notifyComments (M)

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~~a~~ comment~~s~~ have ~~has~~ been associated to one ~~or more~~ alarm~~s~~.

The 'Event Information' field contains the following data:

- alarmedObjectClass: defined in ITU-T X.710 [2] and X.711[3]

- alarmedObjectInstance: defined in ITU-T X.710 [2] and X.711[3]

- alarmEventTime: defined in ITU-T X.721
- alarmType: the eventType of the alarm to which this comment is associated.
- alarmProbableCause: defined in ITU-T X.721
- alarmPerceivedSeverity: defined in ITU-T X.721
- `comments`: the text of the comment.

";

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.”;

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 action(10)}



-- Start of 3GPP SA5 own definitions
AckErrorList ::= SET OF ErrorInfo
AlarmReference ::= SEQUENCE
{
  moi ObjectInstance OPTIONAL, -- absent if scope of uniquness of notificationId is across IRPAgent
  notificationIdentifier NotificationIdentifier
}
AckOrUnackAlarms ::= SEQUENCE
    {
    alarmReferenceList  SET OF AlarmReference,   -- ITU-T X.721
    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)
    }
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
    }
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)
    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

NotifyComments ::= SEQUENCE
    {
    alarmedObjectClass      ObjectClass,
    alarmedObjectInstance   ObjectInstance,
    alarmEventTime          EventTime,
    alarmType               EventTypeId,
    alarmProbableCause      ProbableCause,
    alarmPerceivedSeverity  PerceivedSeverity,
    comments                SET OF Comment
    }

OperationList ::= SET OF OperationName
```

```
OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

SetComment ::= SEQUENCE
    {
    alarmReferenceList      SET OF AlarmReference,
    commentUserId           UserId,
    commentSystemId         SystemId,
    commentText             Comment
    }
SetCommentReply ::= SEQUENCE
    {
    badAlarmReferenceList   SET OF ErrorInfo,
    status                  ErrorCauses
    }
SystemId ::= GraphicString

SupportedAlarmIRPVersions ::= SET OF IRPVersionNumber
UserId ::= GraphicString

END -- of module TS32-111-4TypeModule
```

Annex A (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2000	S_07	SP-000012	-		Approved at TSG SA #7 and placed under Change Control	2.0.0	3.0.0
Mar 2000		-	-		cosmetic	3.0.0	3.0.1
Jun 2000	S_08	SP-000254	005		Split of TS - Part 4: Alarm Integration Reference Point (IRP): CMIP Solution Set (SS)	3.0.1	3.1.0
Sep 2000					cosmetic	3.1.0	3.1.1
Jun 2001	S_12		001		Alarm IRP: CMIP SS Rel4 - Addition of feature. As SA5 had not reviewed this part, it is submitted to SA#12 for Information only.	3.1.2	???

3GPP TS 32.111-4 ~~V3.1.1~~ (2001-07)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Telecommunication Management;
Fault Management;
Part 4: Alarm Integration Reference Point: CMIP Solution Set
(Release 4)

Keywords
Fault Management, Alarms

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Specification (TS) has been produced by the 3rd Generation Partnership Project (3GPP).

The present document is part 4 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects, as identifies below:

Part 1: "3G Fault Management Requirements";

Part 2: "Alarm Integration Reference Point: Information Service";

Part 3: "Alarm Integration Reference Point: CORBA Solution Set";

Part 4: "Alarm Integration Reference Point: CMIP Solution Set".

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document (3GPP TS 32.111 Part-4) defines the alarm integration reference point for the CMIP solution set. In detail:

- Clause 4 contains an introduction to some basic concepts of the CMIP interfaces.
- Clause 5 contains the GDMO definitions for the Alarm Management over the CMIP interfaces
- Clause 6 contains the ASN.1 definitions supporting the GDMO definitions provided in clause 5.

2 References

The following documents contain provisions, which through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] 3GPP TS 32.302: "Notification Integration Reference Point: Information Service".

[2] ITU-T Recommendation X.710: "Common management information service definition for CCITT applications".

[3] ITU-T Recommendation X.711: "Common management information protocol specification for CCITT applications".

[4] ITU-T Recommendation X.721: "Information technology - Open Systems Interconnection - Structure of management information: Definition of management information".

[5] ITU-T Recommendation X.733: "Information technology - Open Systems Interconnection - Systems Management: Alarm reporting function".

[6] ITU-T Recommendation X.734: "Information technology - Open Systems Interconnection - Systems Management: Event report management function".

[7] ITU-T Recommendation Q.821: “Specification of System Signalling No. 7 Q3 Interface- Stage 2 and Stage 3 description for the Q3 interface - Alarm Surveillance”

[8] 3GPP TS 32.111-1: “3G Fault Management”.

[9] 3GPP TS 32.111-2: “Alarm Integration Reference Point: Information Service”.

[10] 3GPP TS 32.304: "Notification Integration Reference Point: CMIP Solution Set".

[11] 3GPP TS 32.312: " Generic IRP Management: InformationService".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions defined in 3GPP TS 32.111-1 [8] apply.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation number 1
CCITT	The International Telegraph and Telephone Consultative Committee
CM	Configuration Management
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
EFD	Event Forwarding Discriminator
EM	Element Manager
FTAM	File Transfer Access and Management
GDMO	Guidelines for the Defifition of Managed Objects
IOC	Information Object Class
IRP	Integration Reference Point
Itf-N	Interface N (between NM and EM/NE)
ITU-T	International Telecommunication Union – Telecommunications
M	Mandatory
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NM	Network Manager
NMC	Network Management Centre
O	Optional
OS	Operations System
TMN	Telecommunications Management Network

4 Basic aspects

The present document provides all the GDMO and ASN.1 definitions necessary to implement the Alarm IRP Information Service [9] for the CMIP interface. 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 modeled as GDMO “Managed Object Classes” (MOC) defined specifically for alarm management, the Operations are modeled as GDMO “Actions” of a MOC while the Notifications are modeled 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 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 [4] and [5] are used.

4.2 Reporting changed alarms

Although in the Alarm IRP Information Service [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 [4] and [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.

4.3 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].

4.4 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 unacknowledge alarms;

– ITU-T 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.

4.5 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.

4.6 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 or NULL, all alarm notifications are forwarded to the Manager through this EFD, 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:

  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).

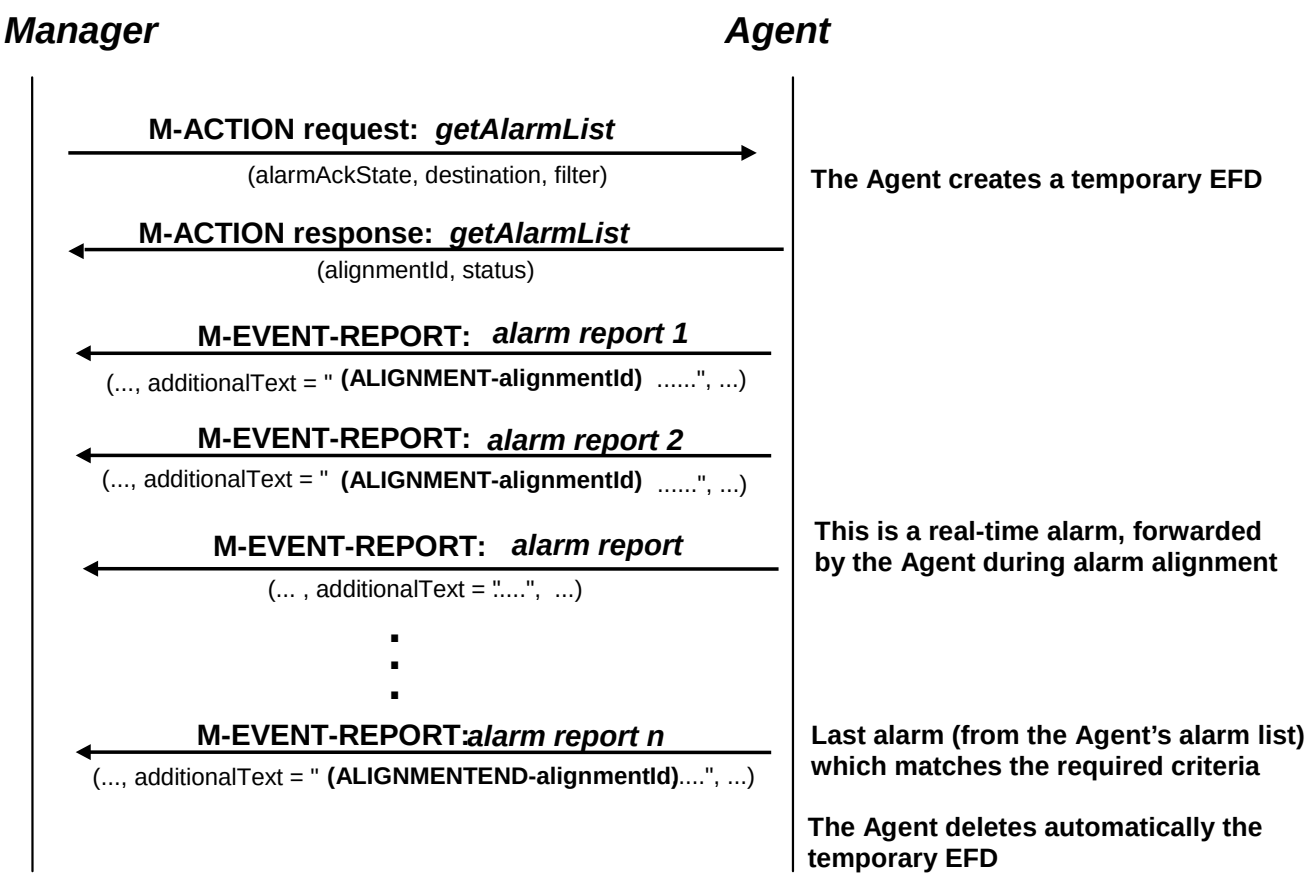

Figure 1: Alignment arrow diagram

Figure 2 shows the handling of a „real-time“ alarm notification (occured 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.

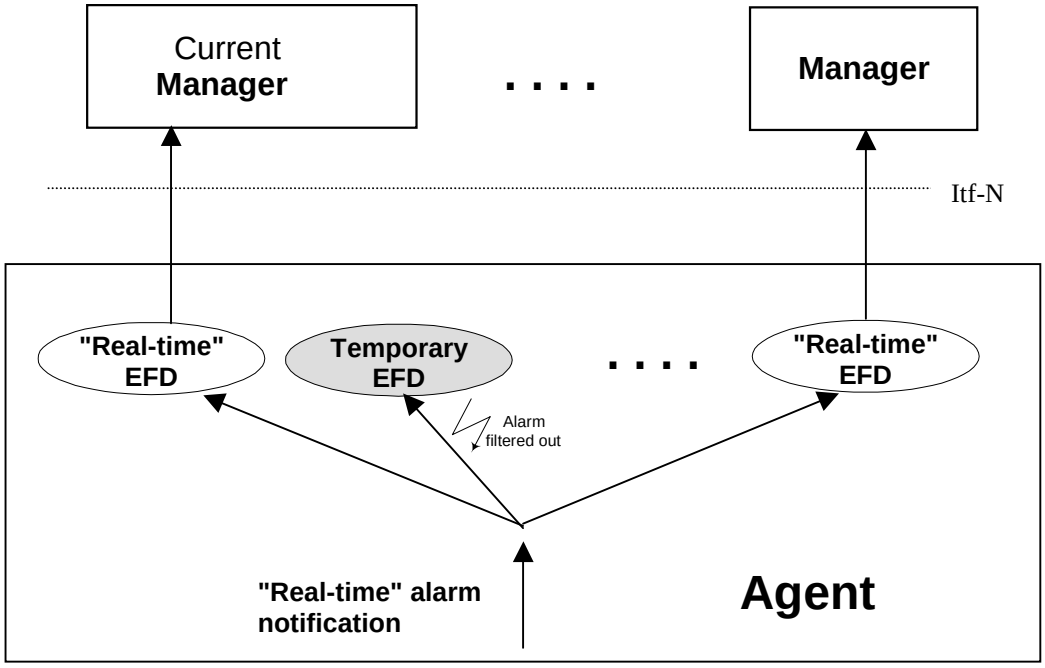

Figure 2: Treatment of “real time” alarms

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.

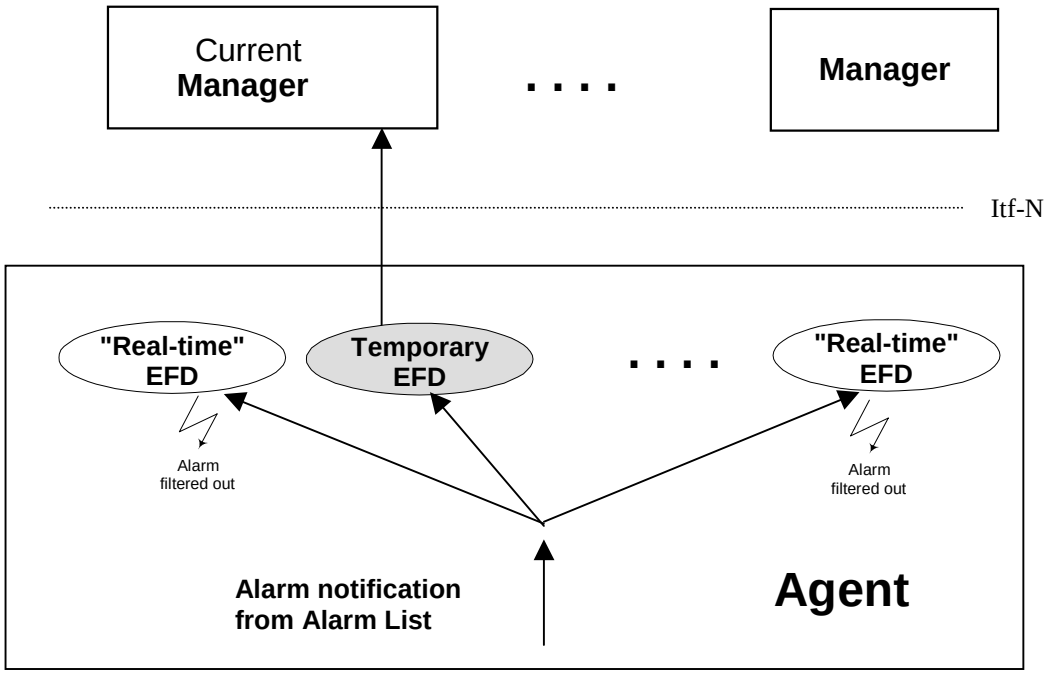

Figure 3: Treatment of “alignent” alarms

4.7 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 section maps these protocol-independent definitions onto the equivalences of the CMIP solution set of Alarm IRP.

4.7.1 Mapping of IOC and Interfaces

For this Alarm IRP CMIP Solution Sets, the Information Object Classes (IOC) and the Interfaces defined in 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 [9].

4.7.2 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
GenericIRPVersionOperation/getIRPVersion	getAlarmIRPVersion	M

GenericIRPProfileOperation/getNotificationProfile	getNotificationProfile	O
GenericIRPProfileOperation/getOperationProfile	getOperationProfile	O

NOTE: the Interfaces GenericIRPVersionOperation and GenericIRPProfileOperation are defined in [11]

4.7.3 Mapping of Parameters of each operation

The tables in the following subclauses 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: Mapping of parameters of ‘acknowledgementAlarms’

Operation parameters of Information Services	IN/OUT	CMIP equivalences	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

Table 3: Mapping of Parameters of ‘getAlarmCount’

Operation parameters of Information Services	IN/OUT	CMIP equivalents	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: Mapping of Parameters of ‘getAlarmList’

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
--		destination (input) - see NOTE 1	M
alarmInformationList	OUT	(sequence of alarm notifications) (see Clause 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: Mapping of Parameters of 'getAlarmIRPVersion'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
versionNumberSet	OUT	versionNumberList	M
status	OUT	status	M

Table 6: Mapping of Parameters of 'getOperationProfile'

Operation parameters of the Information Services.	IN/OUT	CMIP Solution Set equivalences	Qualifier
irpVersion	IN	irpVersionNumber	M
operationNameProfile	OUT	operationNameProfile	M
operationParameterProfile	OUT	operationParameterProfile	M
status	OUT	status	M

Table 7: Mapping of Parameters of 'getNotificatioProfile'

Operation parameters of the Information Services.	IN/OUT	CMIP Solution Set equivalences	Qualifier
irpVersion	IN	irpVersionNumber	M
notificationNameProfile	OUT	notificationNameProfile	M
notificationParameterProfile	OUT	notificationParameterProfile	M
status	OUT	status	M

Table 8: Mapping of Parameters of 'setComment'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
commentUserId	IN	commentUserId	M
commentSystemId	IN	commentSystemId	O
commentText	IN	commentText	M
badAlarmInformationReferenceList	OUT	badAlarmReferenceList	M
Status	OUT	status	M

Table 9: Mapping of Parameters of 'unacknowledgeAlarms'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
alarmInformationReferenceList	IN	alarmReferenceList	M
ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

4.7.4 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 10: 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 qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	M
notifyChangedAlarm	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	O
notifyClearedAlarm	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	M
notifyAckStateChanged	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationAlarm	ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4] ITU-T X.721 [4]	M
notifyAlarmListRebuilt	alarmListRebuilt		M
notifyComments	notifyComments		O

4.7.5 Mapping of Parameters of each notification

The notifications defined in [9] (Alarm IRP: Information Services) have a set of parameters that are common to all the notifications (see [10]: Notification IRP:CMIP Solution Set). Such common set of parameters are:

ManagedObjectClass, ManagedObjectInstance, EventTime, NotificationType, NotificationId.

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 [2,3]). The argument of m-EventReport OPERATION is defined in [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/ASN1 definitions.

Therefore the first four parameters of the notification header are mapped to the first four fields of the EventReportArgument. The fifth parameter of the notification header is mapped to the eventinfo field of the EventReportArgument, together with all the other (not common) parameters of the notification.

In the following tables, for the notifications defined in [9], all the parameters (but the common ones) are mapped to their corresponding elements of the CMIP SS notifications.

The parameter SystemDN defined in [9] (Alarm IRP: Information Services) is not used in this CMIP Solution Set.

Table 11: Mapping of Parameters of 'notifyNewAlarm' and 'notifyClearedAlarm' '

Notification parameters of Information Services	CMIP Notification equivalences	Qualifier
notificationId	notificationIdentifier (Note 1)	M
probableCause	probableCause	M
specificProblems	specificProblems	O
perceivedSeverity	perceivedSeverity	M
backedUpStatus	backedUpStatus	O
backUpObject	backUpObject	O
trendIndication	trendIndication	O
thresholdInfo	thresholdInfo	O
correlatedNotifications	correlatedNotifications	O
stateChangeDefinition	stateChangeDefinition	O
monitoredAttributes	monitoredAttributes	O
proposedRepairActions	proposedRepairActions	O
additionalText	additionalText	O
alarmId	-- (Note 2)	
NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3GPP TS 32.302. NOTE 2: In the CMIP Solution Set the alarmId is not used. In the CMIP Solution Set the alarm notifications are univocally identified by means of notificationIdentifier and managedObjectInstance.		

Table 12: Mapping of Parameters of 'notifyAckStateChanged'

Notification parameters of Information Services	CMIP Notification equivalences	Qualifier
notificationId	notificationIdentifier (Note 1)	M
probableCause	probableCause	M
specificProblems	specificProblems	O
perceivedSeverity	perceivedSeverity	M
alarmId	-- (Note-2)	
ackTime	additionalInformation	M
ackState		M
ackUserId		M
ackSystemId		O
NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3GPP TS 32.302. NOTE 2: In the CMIP Solution Set the alarmId is not used. In the CMIP Solution Set the alarm notifications are univocally identified by means of notificationIdentifier and managedObjectInstance.		

Table 13: Mapping of Parameters of 'notifyAlarmListRebuilt'

Notification parameters of Information Services	CMIP Notification equivalents	Qualifier
notificationId	notificationIdentifier (Note 1)	M
reason	reason	M
objectClass	rebuiltObjectClass	M
objectInstance	rebuiltObjectInstance	M
NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3GPP TS 32.302.		

Table 14: Mapping of Parameters of 'notifyComments'

Notification parameters of Information Services	CMIP Notification equivalents	Qualifier
notificationId	notificationIdentifier (Note 1)	M
objectClass	alarmedObjectClass	M

objectInstance	alarmedObjectInstance	M
notificationId	notificationIdentifier	M
eventTime	alarmEventTime	M
notificationType	eventType	M
alarmType	alarmType	M
probableCause	alarmProbableCause	M
perceivedSeverity	alarmPerceivedSeverity	M
comments	comments	M
alarmId	-- (Note 2)	
NOTE 1: notificationIdentifier is a parameter of the Notification Header also defined in 3GPP TS 32.302. NOTE 2: In the CMIP Solution Set the alarmId is not used. In the CMIP Solution Set the alarm notifications are univocally identified by means of notificationIdentifier and managedObjectInstance.		

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 ";
REGISTERED AS { ts32-111AlarmObjectClass 1};
```

5.2 Packages

5.2.1 alarmControlBasicPackage

```
alarmControlBasicPackage PACKAGE
    BEHAVIOUR
        alarmControlBasicPackageBehaviour;
    ATTRIBUTES
        alarmControlId          GET,
        alarmsCountSummary      GET;
    ACTIONS

        getAlarmList;
    NOTIFICATIONS
        alarmListRebuilt;
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;
REGISTERED AS { ts32-111AlarmPackage 4};

alarmUnacknowledgementPackageBehaviour **BEHAVIOUR**
DEFINED AS

"This package has been defined to provide the Manager with the capability to unacknowledge alarms.

The action 'unacknowledgeAlarms' allows the NM operator to unacknowledge 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
getOperationProfile,
getNotificationProfile;

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.”;

5.3 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

”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 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 (M)

```
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 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 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 getNotificationProfile (O)

getNotificationProfile **ACTION**
BEHAVIOUR
getNotificationProfileBehaviour;
MODE
CONFIRMED;
WITH INFORMATION SYNTAX
TS32-111-4TypeModule.IRPVersionNumber;
WITH REPLY SYNTAX

TS32-111-4TypeModule.GetNotificationProfileReply;
REGISTERED AS { ts32-111AlarmAction 6};

getNotificationProfileBehaviour **BEHAVIOUR**

DEFINED AS

"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 getOperationProfile (O)

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

getOperationProfileBehaviour **BEHAVIOUR**

DEFINED AS

"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

"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 unacknowledge it later.

- *ackSystemId*
  It indicates the management system where the acknowledgment is triggered. It may have also the value NULL. Note that the unacknowledgement 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 unknowledgement 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. ";

5.4 Notifications

5.4.1 alarmListRebuilt (M)

alarmListRebuilt **NOTIFICATION**
BEHAVIOUR
alarmListRebuiltBehaviour;
WITH INFORMATION SYNTAX
TS32-111-4TypeModule .AlarmListRebuiltInfo;
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 pertially 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).”;

5.4.2 notifyComments (M)

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: defined in ITU-T X.710 [2] and X.711[3]
- alarmedObjectInstance: defined in ITU-T X.710 [2] and X.711[3]
- alarmEventTime: defined in ITU-T X.721
- alarmType: the eventType of the alarm to which this comment is associated.
- alarmProbableCause: defined in ITU-T X.721

- alarmPerceivedSeverity: defined in ITU-T X.721
- `comments`: the text of the comment.

";

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.”;

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 action(10)}



-- Start of 3GPP SA5 own definitions
AckErrorList ::= SET OF ErrorInfo
AlarmReference ::= SEQUENCE
{
  moi ObjectInstance OPTIONAL, -- absent if scope of uniquness of notificationId is across IRPAgent
  notificationIdentifier NotificationIdentifier
}
AckOrUnackAlarms ::= SEQUENCE
    {
    alarmReferenceList  SET OF AlarmReference,   -- ITU-T X.721
    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)
    }
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
    }
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)
    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

NotifyComments ::= SEQUENCE
    {
    alarmedObjectClass      ObjectClass,
    alarmedObjectInstance   ObjectInstance,
    alarmEventTime          EventTime,
    alarmType               EventTypeId,
    alarmProbableCause      ProbableCause,
    alarmPerceivedSeverity  PerceivedSeverity,
    comments                SET OF Comment
    }

OperationList ::= SET OF OperationName
```

OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

SetComment ::= SEQUENCE
{
alarmReferenceList SET OF AlarmReference,
commentUserId UserId,
commentSystemId SystemId,
commentText Comment
}
SetCommentReply ::= SEQUENCE
{
badAlarmReferenceList SET OF ErrorInfo,
status ErrorCauses
}
SystemId ::= GraphicString

SupportedAlarmIRPVersions ::= SET OF IRPVersionNumber
UserId ::= GraphicString

END -- of module TS32-111-4TypeModule

Annex A (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2000	S_07	SP-000012	-		Approved at TSG SA #7 and placed under Change Control	2.0.0	3.0.0
Mar 2000		-	-		cosmetic	3.0.0	3.0.1
Jun 2000	S_08	SP-000254	005		Split of TS - Part 4: Alarm Integration Reference Point (IRP): CMIP Solution Set (SS)	3.0.1	3.1.0
Sep 2000					cosmetic	3.1.0	3.1.1
Jun 2001	S_12		001		Alarm IRP: CMIP SS Rel4 - Addition of feature. As SA5 had not reviewed this part, it is submitted to SA#12 for Information only.	3.1.2	???