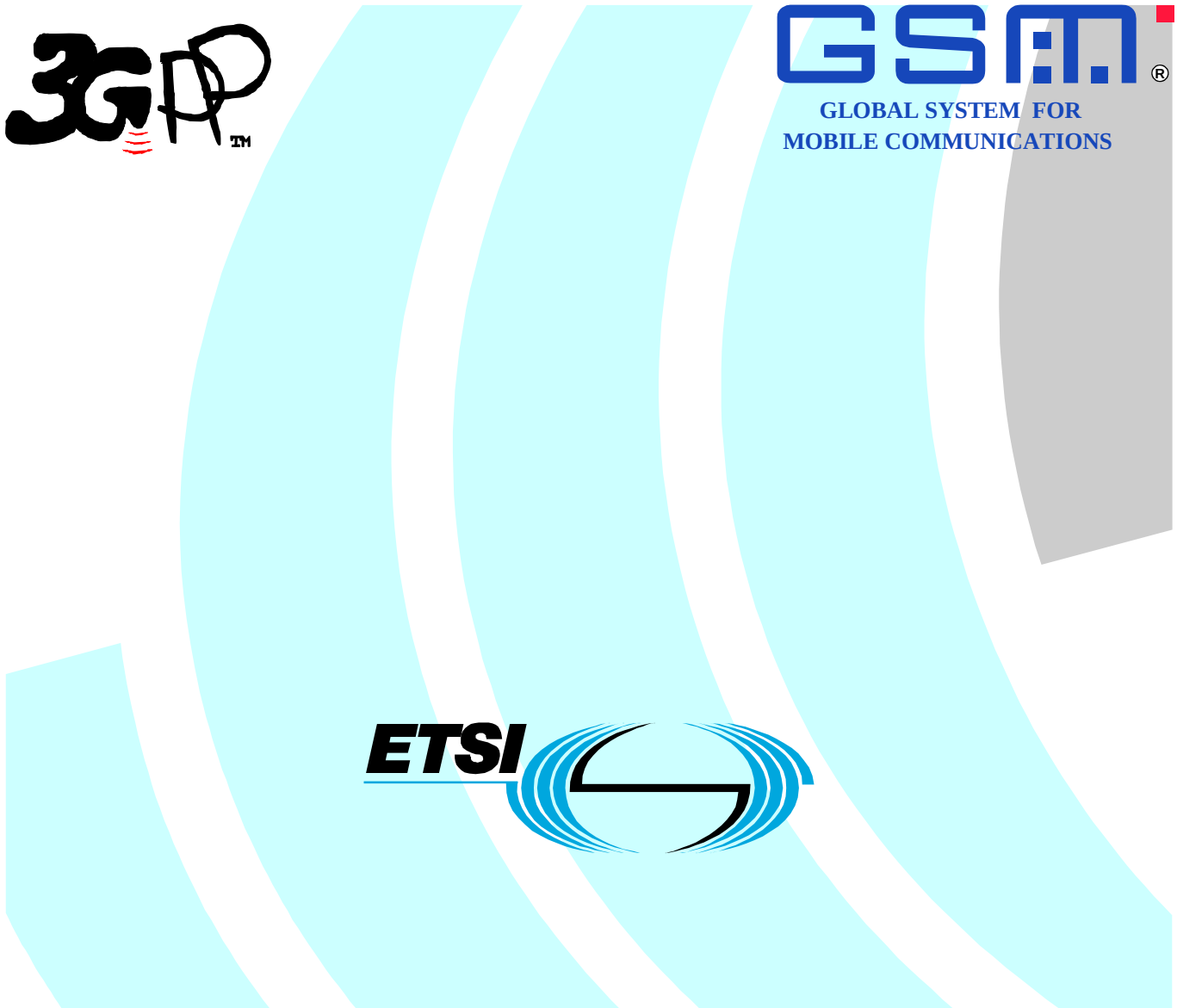
ETSI TS 132 111-4 V6.5.0 (2005-06)

Technical Specification

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
Telecommunication management;
Fault Management;
Part 4: Alarm Integration Reference Point (IRP):
Common Management Information Protocol (CMIP)
Solution Set (SS)
(3GPP TS 32.111-4 version 6.5.0 Release 6)



Reference
RTS/TSGS-0532111-4v650

Keywords
GSM, UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

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

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp .

Contents

Foreword

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

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.

Introduction

The present document is part of a TS-family covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects; Telecommunication management; as identified below:

32.111-1 "Fault Management; Part 1: 3G fault management requirements".

32.111-2 "Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)".

32.111-3 "Fault Management; Part 3: Alarm Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS)".

32.111-4 "Fault Management; Part 4: Alarm Integration Reference Point (IRP): Common Management Information Protocol (CMIP) Solution Set (SS)".

32.111-5 "Fault Management; Part 5: Alarm Integration Reference Point (IRP): eXtensible Markup Language (XML) definitions".

1 Scope

The present document 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.

This Solution Set specification is related to 3GPP TS 32.111-2 (V6.3.X).

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. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TS 32.302: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Information Service (IS)".

[2] ITU-T Recommendation X.710: "Information technology - Open Systems Interconnection - Common Management Information Service".

[3] ITU-T Recommendation X.711: "Information technology - Open Systems Interconnection - Common Management Information Protocol: Specification".

[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: "Stage 2 and Stage 3 description for the Q3 interface - Alarm Surveillance".

[8] 3GPP TS 32.111-1: "Telecommunication management; Fault Management; Part 1: 3G fault management requirements".

[9] 3GPP TS 32.111-2: "Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)".

[10] 3GPP TS 32.304: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Common Management Information Protocol (CMIP) Solution Set (SS)".

[11] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[12] ITU-T Recommendation X.736: "Information technology - Open Systems Interconnection - Systems Management: Security alarm reporting function".

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 Definition 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 (3GPP TS 32.111-2 [9]) for the CMIP interface.

4.1 Architectural aspects

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

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

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

4.1.1 Reporting new alarms

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

For this purpose the standardised alarm notifications defined in ITU-T Recommendations X.721 [4], X.733 [5] and X.736 [12] are used.

4.1.2 Reporting changed alarms

Although in the Alarm IRP Information Service (3GPP TS 32.111-2 [9]) there is a notification specifically defined to report the event of alarm attribute changes, on the CMIP interfaces such events are reported according to ITU-T Recommendations X.721 [4], X.733 [5] and X.736 [12], i.e. the original alarm is first cleared (by means of a clear alarm notification) and then a new alarm notification with the changed parameter values is generated by the Agent.

4.1.3 Reporting cleared alarms

On the CMIP interfaces the clearing of alarms is reported by the Agent to the Managers in accordance with the mechanisms defined in ITU-T Recommendation X.733 [5], X.736 [12] and ITU-T Recommendation Q.821 [7].

4.1.4 Acknowledgment of alarms

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

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

- one action to acknowledge alarms (*acknowledgeAlarms*);
- one action to unacknowledge alarms (*unacknowledgeAlarms*);
- ITU-T Recommendation X.721 [4] compliant alarm notifications to inform Managers about changes of acknowledgment state.

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

4.1.5 Management of comments associated to alarms

This feature provides the NM and EM 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. This implies that a synchronisation of the comments between the EM and NM shall be possible. 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 a`larmControl` includes

- one action (*setComment*) allowing the NM operator to add a comment to one or several alarms;
- ITU-T Recommendation X.721 [4] compliant alarm notifications to inform the IRPManagers about changes of alarm related comments. Such notifications are generated by the Agent towards all connected Managers either if the comment is made by an NM operator (i.e. after the completion of a previous *setComment* request) or if the comment is made by an EM operator.

4.1.6 Alignment of alarm conditions over the Itf-N

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

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

- The Manager sends to the Agent a *getAlarmList* request containing the following information:
  - *alarmAckState*, used to select the alarms from the Agent's alarm list for the current alignment (e.g. all active alarms).
  - *baseObjectClass*, *baseObjectInstance*, identifies the part of the alarm list to be uploaded.
  - *destination*, identifying the destination to which event reports that have passed the filter conditions are sent.
  - *filter*, this optional parameter defines the conditions an alarm notification shall fulfil in order to be forwarded to the Manager. It applies only for the current alignment request.
- After evaluation of the request, the Agent first generates an *alignmentId* value, which unambiguously identifies this alignment process. This value is used by the Manager to correlate alarm reports to the corresponding alignment requests, in case this Manager issues several alarm alignments in parallel.
- The Agent creates a temporary Event Forwarding Discriminator (EFD) instance for the purpose of this alarm alignment, using the parameters *destination* and *filter* received in the request. If the *filter* parameter is absent in the alarm synchronisation request, all alarm notifications are forwarded to the Manager through this EFD, taking into account both the *filter* constraint currently active for the event reporting to the manager having invoked the synchronisation request and the value of the parameter *alarmAckState*.
  The filter is set by the Agent automatically in order to forward only those alarm notifications containing, at the beginning of the field *additionalText*, the string "(ALIGNMENT-<alignmentId>)". The filter must also forward the notification *notifyAlarmAlignmentEnd* indicating the end of the alarm alignment process. The alarm alignment end notifications of other alignment processes shall be filtered out using the *alignmentId* carried by the event information parameter of *notifyAlarmAlignmentEnd*.
- 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 and the *filter* parameter, the Agent inserts, at the beginning of the field *additionalText*, the string "(ALIGNMENT-<alignmentId>)".
- Depending on the event being reported, the *additionalInformation* field of every alarm notification shall carry the parameters *ackTimeParameter*, *ackStateParameter*, *ackUserIdParameter*, *ackSystemIdParameter*, *clearUserIdParameter*, *clearSystemIdParameter*, *commentsParameter*, *alarmRaisedTimeParameter* or *alarmClearedTimeParameter*.
- According to ITU-T Recommendation X.734 [6], the Agent forwards these alarm notifications towards all EFDs.

  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.
- At the end of the alarm alignment process the Agent shall send the dedicated notification *notifyAlarmAlignmentEnd* in order to indicate the end of the current alignment process (unambiguously identified by the *alignmentId*). In case the alarm list is empty or no alarm matches the criteria defined by the *alarmAckState* parameter and the *filter* parameter the notification *notifyAlarmAlignmentEnd* shall be emitted directly after the agent has send the *getAlarmList* response.
- The temporary EFD of the current alarm alignment process shall forward only alarm alignment end notifications carrying in the event information field the *alignmnentId* of this alignment process. All other alarm alignment end notifications shall be filtered out.

- Each NMC has to set the filter of its permanent EFD instance in order to block the notifyAlarmAlignmentEnd notification (otherwise the NMC would receive this notification twice: Once by the temporary EFD, once by the permanent)

- In case of several alignments running in parallel, each NMC has to evaluate the alignmentId value of every received notifyAlarmAlignmentEnd notification (passed via all "temporary" EFD instances) and to ignore those notifications containing alignmentId values that do not correspond to one of its own alignments.

- After sending the notification *notifyAlarmAlignmentEnd* the Agent automatically deletes the temporary EFD instance (see figure 1).

At the end of the alarm conditions alignment the acknowledgement state and the comments assigned to each alarm are implicitly synchronised between the IRPAgent and the IRPManager that has requested the alignment.

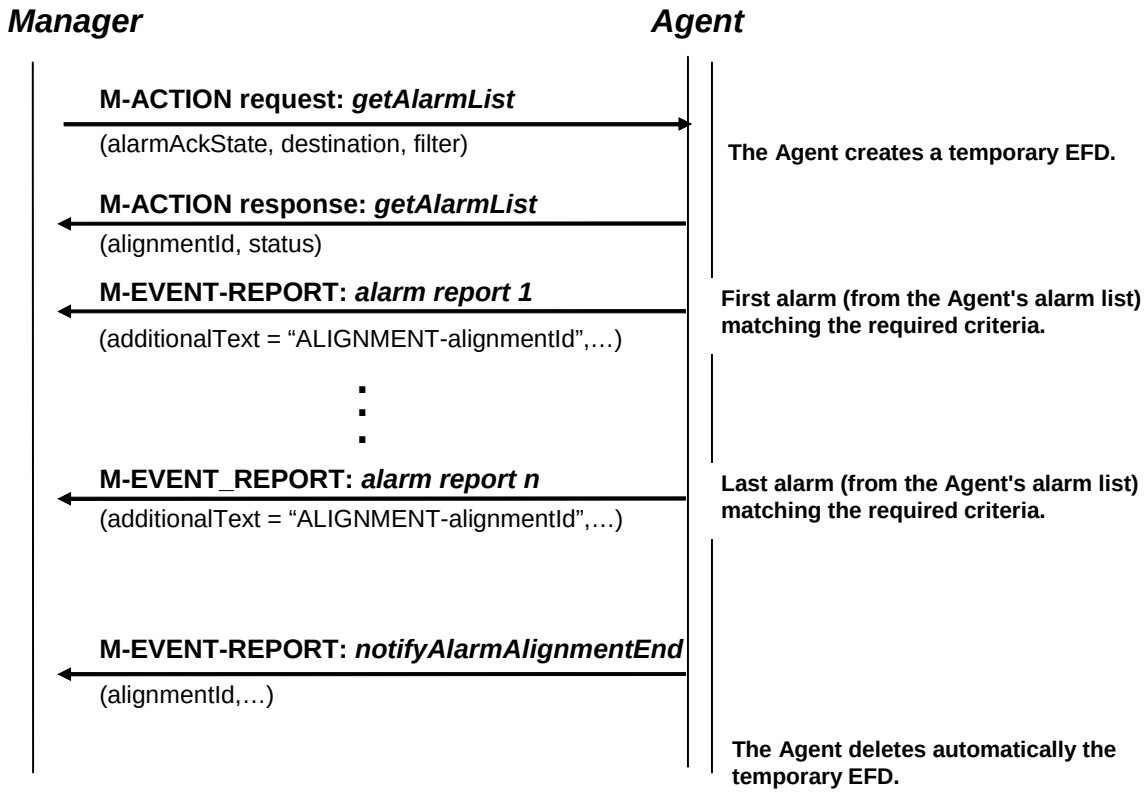


Figure 1: Alignment arrow diagram

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.

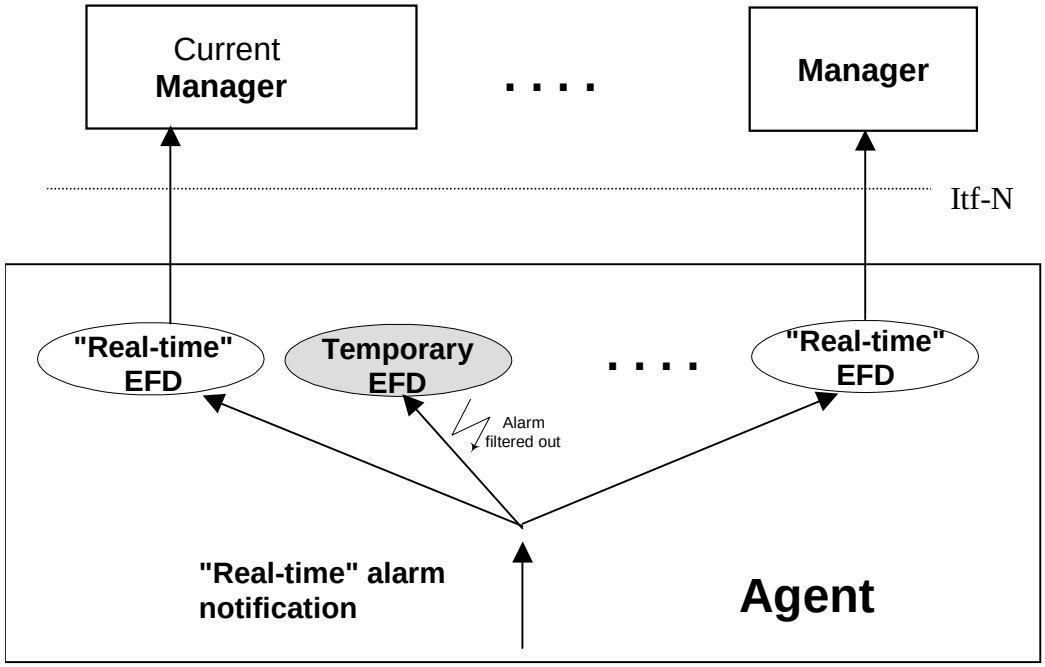


Figure 2: Treatment of "real time" alarms

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.

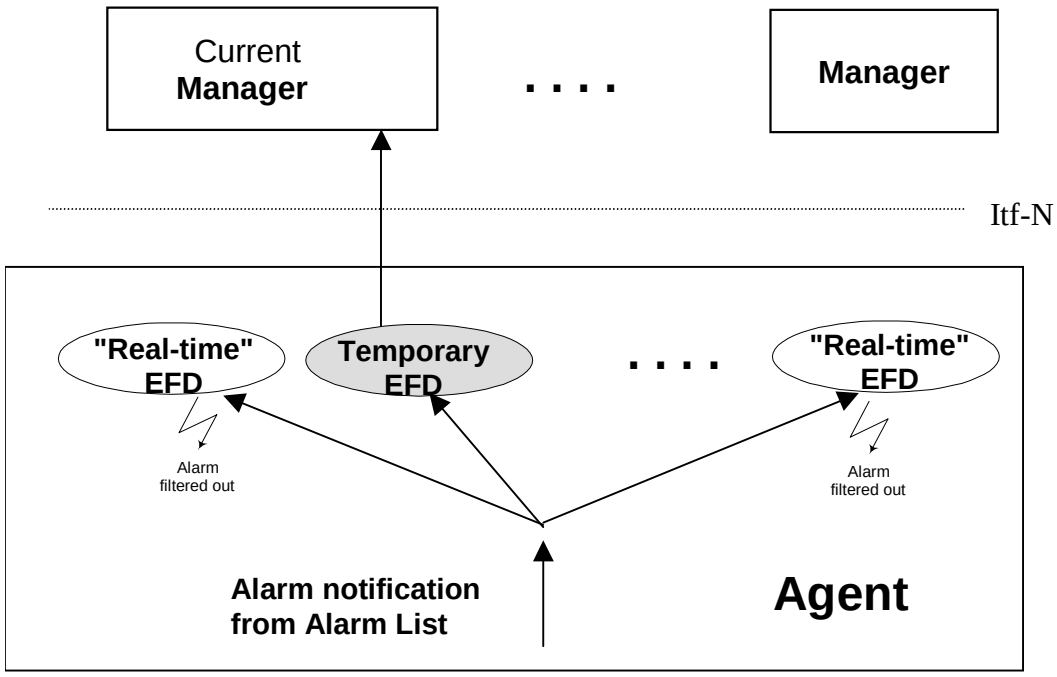


Figure 3: Treatment of "alignment" alarms

It is possible to abort an ongoing alarm alignment process by invoking the action *abortGetAlarmList*. Also in this case the notification *notifyAlarmAlignmentEnd* is emitted.

4.2 Mapping

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

4.2.1 Mapping of Information Object Classes

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

4.2.2 Mapping of 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

IS Interface	IS Operation	CMIP SS Equivalent	Qualifier
AlarmIRPOperations_1	acknowledgeAlarms	CMISE M-ACTION service, action type: acknowledgeAlarms	M
	getAlarmList	CMISE M-ACTION service, action type: getAlarmList environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationsAlarm ITU-T X.721 [4] integrityViolation ITU-T X.721 [4] operationalViolation ITU-T X.721 [4] physicalViolation ITU-T X.721 [4] securityServiceOrMechanismViolation ITU-T X.721 [4] timeDomainViolation ITU-T X.721 [4] CMISE M-EVENT-REPORT service, event type: notifyAlarmAlignmentEndR0602	M
	Method to abort an ongoing alarm alignment process	abortGetAlarmList	M
AlarmIRPOperations_2	getAlarmCount	CMISE M-ACTION service, action type: getAlarmCount	O
AlarmIRPOperations_3	unacknowledgeAlarms	CMISE M-ACTION service, action type: unacknowledgeAlarms	O
AlarmIRPOperations_4	setComment	CMISE M-ACTION service, action type: setComment	O
AlarmIRPOperations_5	clearAlarms	CMISE M-ACTION service, action type: clearAlarms	O
GenericIRPVersionOperation	getIRPVersion	CMISE M-ACTION service, action type: getAlarmIRPVersion	M
GenericIRPProfileOperation	getNotificationProfile	CMISE M-ACTION service, action type: getAlarmIRPNotificationProfile	O
	getOperationProfile	CMISE M-ACTION service, action type: getAlarmIRPOperationProfile	O

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

4.2.3 Mapping of Operation Parameters

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

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

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

Table 2: Parameter mapping of the operation *acknowledgeAlarms*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationAndSeverityReferenceList	IN	M-ACTION parameter 'Action information' (AckOrUnackAlarmsInfo): alarmReferenceList (note)	M
ackUserId	IN	M-ACTION parameter 'Action information' (AckOrUnackAlarmsInfo): ackUserId	M
ackSystemId	IN	M-ACTION parameter 'Action information' (AckOrUnackAlarmsInfo): ackSystemId	O
badAlarmInformationReferenceList	OUT	M-ACTION parameter 'Action reply' (AckOrUnackAlarmsReply): errorAlarmReferenceList	M
status	OUT	M-ACTION parameter 'Action reply' (AckOrUnackAlarmsReply): status	M
NOTE: severity verification not required in CMIP solution set.			

Table 3: Parameter mapping of the operation *getAlarmCount*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
filter	IN	M-ACTION parameter 'Action information' (GetAlarmCountInfo): filter	O
alarmAckState	IN	M-ACTION parameter 'Action information' (GetAlarmCountInfo): alarmAckState	O
criticalCount	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): criticalCount	M
majorCount	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): majorCount	M
minorCount	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): minorCount	M
warningCount	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): warningCount	M
indeterminateCount	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): indeterminateCount	M
clearedCount	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): clearedCount	M
status	OUT	M-ACTION parameter 'Action reply' (GetAlarmCountReply): status	M

Table 4: Parameter mapping of the operation *getAlarmList*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
filter	IN	M-ACTION parameter 'Action information' (GetAlarmListInfo): filter	O
alarmAckState	IN	M-ACTION parameter 'Action information' (GetAlarmListInfo): alarmAckState	O
baseObjectClass	IN	M-ACTION parameter 'Action information' (GetAlarmListInfo): baseObjectClass	O
baseObjectInstance	IN	M-ACTION parameter 'Action information' (GetAlarmListInfo): baseObjectInstance	O
--	IN	M-ACTION parameter 'Action information' (GetAlarmListInfo): destination (see note 1)	M
alarmInformationList	OUT	sequence of alarm notifications, see subclause 4.1.6	M
status	OUT	M-ACTION parameter 'Action reply' (GetAlarmListReply): status	M
--	OUT	M-ACTION parameter 'Action reply' (GetAlarmListReply): alignmentId (see note 2)	M
NOTE 1: Destination is a CMIP specific parameter and is determined by the Manager. NOTE 2: AlignmentId is a CMIP specific parameter and is determined by the Agent.			

Table 5: Parameter mapping of the operation *getAlarmIRPVersion*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
versionNumberSet	OUT	M-ACTION parameter 'Action reply' (GetAlarmIRPVersionReply): versionNumberList	M
status	OUT	M-ACTION parameter 'Action reply' (GetAlarmIRPVersionReply): status	M

Table 6: Parameter mapping of the operation *getOperationProfile*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
irpVersion	IN	M-ACTION parameter 'Action information': irpVersionNumber	M
operationNameProfile	OUT	M-ACTION parameter 'Action reply' (GetOperationProfileReply): operationNameProfile	M
operationParameterProfile	OUT	M-ACTION parameter 'Action reply' (GetOperationProfileReply): operationParameterProfile	M
status	OUT	M-ACTION parameter 'Action reply' (GetOperationProfileReply): status	M

Table 7: Parameter mapping of the operation *getNotificationProfile*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
irpVersion	IN	M-ACTION parameter 'Action information': irpVersionNumber	M
notificationNameProfile	OUT	M-ACTION parameter 'Action reply' (GetNotificationProfileReply): notificationNameProfile	M
notificationParameterProfile	OUT	M-ACTION parameter 'Action reply' (GetNotificationProfileReply): notificationParameterProfile	M
status	OUT	M-ACTION parameter 'Action reply' (GetNotificationProfileReply): status	M

Table 8: Parameter mapping of the operation *setComment*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationReferenceList	IN	M-ACTION parameter 'Action information' (SetCommentInfo): alarmReferenceList	M
commentUserId	IN	M-ACTION parameter 'Action information' (SetCommentInfo): commentUserId	M
commentSystemId	IN	M-ACTION parameter 'Action information' (SetCommentInfo): commentSystemId	O
commentText	IN	M-ACTION parameter 'Action information' (SetCommentInfo): commentText	M
badAlarmInformationReferenceList	OUT	M-ACTION parameter 'Action reply' (SetCommentReply): errorAlarmReferenceList	M
status	OUT	M-ACTION parameter 'Action reply' (SetCommentReply): status	M

Table 9: Parameter mapping of the operation *unacknowledgeAlarms*

IS Parameter	IN/OUT	CMIP SS Equivalent	Qualifier
alarmInformationReferenceList	IN	M-ACTION parameter 'Action information' (AckOrUnackAlarmsInfo): alarmReferenceList	M
ackUserId	IN	M-ACTION parameter 'Action information' (AckOrUnackAlarmsInfo): ackUserId	M
ackSystemId	IN	M-ACTION parameter 'Action information' (AckOrUnackAlarmsInfo): ackSystemId	O
badAlarmInformationReferenceList	OUT	M-ACTION parameter 'Action information' (AckOrUnackAlarmsReply): errorAlarmReferenceList	M
status	OUT	M-ACTION parameter 'Action information' (AckOrUnackAlarmsReply): status	M

Table 10: Parameter mapping of the operation *clearAlarms*

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

4.2.4 Mapping of Notifications

Table 11 maps the Notifications defined in the Information Service of the Alarm IRP to the equivalent Notifications of the CMIP solution set for the Alarm IRP. The CMIP Notifications are qualified as Mandatory (M) or Optional (O).

Table 11: Mapping of Notifications

IS Notification	CMIP SS Equivalent		Qualifier
notifyNewAlarm	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationsAlarm integrityViolation operationalViolation physicalViolation securityServiceOrMechanismViolation timeDomainViolation	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] 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
notifyChangedAlarm	notifyClearedAlarm notifyNewAlarm which are in turn mapped into environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationsAlarm integrityViolation operationalViolation physicalViolation securityServiceOrMechanismViolation timeDomainViolation	 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] 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 communicationsAlarm integrityViolation operationalViolation physicalViolation securityServiceOrMechanismViolation timeDomainViolation	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] 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 communicationsAlarm integrityViolation operationalViolation physicalViolation securityServiceOrMechanismViolation timeDomainViolation	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] 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	notifyAlarmListRebuiltR0602		M
notifyComments	environmentalAlarm equipmentAlarm qualityofServiceAlarm processingErrorAlarm communicationsAlarm integrityViolation operationalViolation physicalViolation securityServiceOrMechanismViolation timeDomainViolation	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] 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
notifyPotentialFaultyAlarmList	notifyPotentialFaultyAlarmListR0602		O

4.2.5 Mapping of Notification Parameters

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

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

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

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

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

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

- If the alarm type is equal to 'Communications Alarm' the notification *communicationsAlarm* is emitted;
- If the alarm type is equal to 'Processing Error Alarm' the notification *processingErrorAlarm* is emitted;
- If the alarm type is equal to 'Environmental Alarm' the notification *environmentalAlarm* is emitted;
- If the alarm type is equal to 'Quality of Service Alarm' the notification *qualityofServiceAlarm* is emitted;
- If the alarm type is equal to 'Equipment Alarm' the notification *equipmentAlarm* is emitted.
- If the alarm type is equal to 'Integrity Violation ' the notification *integrityViolation* is emitted.
- If the alarm type is equal to 'Operational Violation ' the notification *operationalViolation* is emitted.
- If the alarm type is equal to 'Physical Violation ' the notification *physicalViolation* is emitted.
- If the alarm type is equal to 'Security Violation ' the notification *securityServiceOrMechanismViolation* is emitted.
- If the alarm type is equal to 'Time Domain Violation ' the notification *timeDomainViolation* is emitted.

Also the IS parameter *alarmId* is not mapped directly to a parameter in the CMIP SS. This is not required because an alarm is identified unambiguously by the notification identifier of the notification reporting the alarm the first time and, if the notification identifier is not unique across the IRPAgent, by the instance of the managed object emitting this notification. Notifications referring to an alarm already reported (e.g. *notifyClearedAlarm*, *notifyAckStateChanged*, *notifyComments*) do so by specifying in the M-EVENT REPORT parameter 'Event information': *correlatedNotifications* (ITU-T Recommendations X.721 [4], X.733 [5] and X.736 [12]) the notification identifier of the notification having reported the new alarm and, if required, the instance of the object having emitted this notification.

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

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

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

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

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

Table 13: Parameter mapping of the notification *notifyClearedAlarm*

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

Table 14: Parameter mapping of the notification *notifyAckStateChanged*

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

Table 15: Parameter mapping of the notification *notifyAlarmListRebuilt*

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

Table 16: Parameter mapping of the notification *notifyComments*

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

Table 17: Parameter mapping of the notification *notifyPotentialFaultyAlarmList*

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

-- 5 GDMO Definitions

--Please do not remove the '—' in front of the headline numbering, as it is the CMIP code

--for a comment. This way the whole chapter can be put directly into a compiler.

-- 5.1 Managed Object Classes

-- 5.1.1 alarmControl

alarmControlR0602 **MANAGED OBJECT CLASS**

DERIVED FROM

"Rec. X.721 | ISO/IEC 10165-2 : 1992":top;

CHARACTERIZED BY

alarmControlBasicPackageR0602,

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",

alarmPotentialFaultyAlarmListPackageR0602 **PRESENT IF** "an instance supports it",

alarmClearPackage **PRESENT IF** "an instance supports it";

REGISTERED AS {ts32-111AlarmObjectClass 10602};

-- 5.2 Packages

-- 5.2.1 alarmControlBasicPackage

alarmControlBasicPackageR0602 **PACKAGE**

BEHAVIOUR

alarmControlBasicPackageR0602Behaviour;

ATTRIBUTES

alarmControlId GET,

alarmsCountSummary GET;

ACTIONS

getAlarmList,

abortGetAlarmList;

NOTIFICATIONS

notifyAlarmListRebuiltR0602,

notifyAlarmAlignmentEndR0602;

REGISTERED AS {ts32-111AlarmPackage 10602};

alarmControlBasicPackageR0602Behaviour **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 'notifyAlarmListRebuilt' 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.

The notification 'notifyAlarmAlignmentEnd' is sent by the Agent to the Manager to inform that the alarm alignment process identified by the 'alignmentId' is completed.";

-- 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,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": integrityViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": operationalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": physicalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": securityServiceOrMechanismViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": timeDomainViolation;

REGISTERED AS {ts32-111AlarmPackage 3};

alarmAcknowledgementPackageBehaviour **BEHAVIOUR**

DEFINED AS

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

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

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

-- 5.2.4 alarmUnacknowledgementPackage

alarmUnacknowledgementPackage **PACKAGE**

BEHAVIOUR

alarmUnacknowledgementPackageBehaviour;

ACTIONS

unacknowledgeAlarms;

NOTIFICATIONS

"Rec. X.721 | ISO/IEC 10165-2 : 1992": communicationsAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": environmentalAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": equipmentAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": processingErrorAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": qualityofServiceAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": integrityViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": operationalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": physicalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": securityServiceOrMechanismViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": timeDomainViolation;

REGISTERED AS {ts32-111AlarmPackage 4};

alarmUnacknowledgementPackageBehaviour **BEHAVIOUR**

DEFINED AS

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

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

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

-- 5.2.5 alarmCommentPackage

alarmCommentPackage **PACKAGE**

BEHAVIOUR

alarmCommentPackageBehaviour;

ACTIONS

setComment;

NOTIFICATIONS

"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,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": integrityViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": operationalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": physicalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": securityServiceOrMechanismViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": timeDomainViolation;

REGISTERED AS {ts32-111AlarmPackage 5};

alarmCommentPackageBehaviour **BEHAVIOUR**

DEFINED AS

"This package has been defined to allow the management of comments related to alarms.

The action *setComment* allows the IRPManager to add a comment to one or several alarms. Also the IRPAgent may add comments to alarms.

ITU-T Recommendation X.721 [4] compliant alarm notifications are generated once a comment is added to an alarm. The information in all comments associated to an alarm is carried in the attribute *additionalInformation*.";

-- 5.2.6 alarmIRPVersionPackage

alarmIRPVersionPackage **PACKAGE**

BEHAVIOUR

alarmIRPVersionPackageBehaviour;

ATTRIBUTES

supportedAlarmIRPVersions GET;

ACTIONS

getAlarmIRPVersion;

REGISTERED AS {ts32-111AlarmPackage 6};

alarmIRPVersionPackageBehaviour **BEHAVIOUR**

DEFINED AS

"This package has been defined to allow the Manager to get information about the Alarm IRP versions supported by the Agent.

The attribute 'supportedAlarmIRPVersions' indicates all versions of the Alarm IRP currently supported by the Agent.

The action 'getAlarmIRPVersion' may be invoked by the Manager to get information about the Alarm IRP versions supported by the Agent. Such Alarm IRP versions must compatible to each other. This means that the Manager may use any one of such Alarm IRP versions";

-- 5.2.7 alarmProfilePackage

alarmProfilePackage **PACKAGE**

BEHAVIOUR

alarmProfilePackageBehaviour;

ACTIONS

getAlarmIRPOperationProfile,

getAlarmIRPNotificationProfile;

REGISTERED AS {ts32-111AlarmPackage 7};

alarmProfilePackageBehaviour **BEHAVIOUR**

DEFINED AS

"This package has been defined to allow the Manager to get detailed information about the profile of Alarm IRP.

The action 'getOperationProfile' is invoked by the Manager to get detailed information about the operations supported by Alarm IRP.

The action 'getNotificationProfile' is invoked by the Manager to get detailed information about the notifications supported by Alarm IRP.";

-- 5.2.8 alarmPotentialFaultyAlarmListPackage

alarmPotentialFaultyAlarmListPackageR0602 **PACKAGE**

BEHAVIOUR

alarmPotentialFaultyAlarmListPackageR0602Behaviour;

NOTIFICATIONS

notifyPotentialFaultyAlarmListR0602;

REGISTERED AS {ts32-111AlarmPackage 80602};

alarmPotentialFaultyAlarmListPackageR0602Behaviour **BEHAVIOUR**

DEFINED AS

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

-- 5.2.9 alarmClearPackage

alarmClearPackage **PACKAGE**

BEHAVIOUR

alarmClearPackageBehaviour;

ACTIONS

clearAlarms;

REGISTERED AS {ts32-111AlarmPackage 9};

alarmClearPackageBehaviour **BEHAVIOUR**

DEFINED AS

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

-- 5.2.10 x721AlarmNotificationsPackage

x721AlarmNotificationsPackage **PACKAGE**

BEHAVIOUR

x721AlarmNotificationsPackageBehaviour;

NOTIFICATIONS

"Rec. X.721 | ISO/IEC 10165-2 : 1992": communicationsAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": environmentalAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": equipmentAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": processingErrorAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": qualityofServiceAlarm,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": integrityViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": operationalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": physicalViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": securityServiceOrMechanismViolation,

"Rec. X.721 | ISO/IEC 10165-2 : 1992": timeDomainViolation;

REGISTERED AS {ts32-111AlarmPackage 10};

x721AlarmNotificationsPackageBehaviour **BEHAVIOUR**

DEFINED AS

"This package contains all alarm notifications defined in ITU-T X.721.";

-- 5.3 Actions

-- 5.3.1 acknowledgeAlarms (M)

acknowledgeAlarms **ACTION**

BEHAVIOUR

acknowledgeAlarmsBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.AckOrUnackAlarmsInfo;

WITH REPLY SYNTAX

TS32-111-4TypeModule.AckOrUnackAlarmsReply;

REGISTERED AS {ts32-111AlarmAction 1};

acknowledgeAlarmsBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' field contains the following data:

- *alarmReferenceList*

  This parameter contains a set of MOI (Managed Object Instance) and *notificationIdentifier*. Each pair identifies unambiguously in the scope of the Agent an alarm (previously received by the NM) that have to be now acknowledged. MOI can be absent if scope of uniqueness of notificationIdentifier is across the IRPAgent.

- *ackUserId*

  It contains the name of the operator who acknowledged the alarm or a generic name (dependent on the operational concept). It may have also the value NULL.

- *ackSystemId*

  It indicates the management system where the acknowledgment is triggered. It may have also the value NULL.

The 'Action response' contains the following data:

- *status*

  This parameter contains the results of the NM acknowledgement action. Possible values: noError (0, all alarms found and ack state changed according to the manager request), ackPartlySuccessful (some alarms not found / not changeable, see next parameter), error (value indicates the reason why the complete operation failed).

- *errorAlarmReferenceList*

  This parameter (significant only if *status* = ackPartlySuccessful) contains the list of moi (managed object instance) and notificationIdentifier pairs of the alarms which could not be acknowledged and, for each alarm, also the reason of the error.";

-- 5.3.2 getAlarmCount (O)

getAlarmCount **ACTION**

BEHAVIOUR

getAlarmCountBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.GetAlarmCountInfo;

WITH REPLY SYNTAX

TS32-111-4TypeModule.GetAlarmCountReply;

REGISTERED AS {ts32-111AlarmAction 2};

getAlarmCountBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' field contains the following data:

- *alarmAckState*

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

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

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

- *filter*

  The handling of this optional parameter is as follows:

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

The 'Action response' is composed of:

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

-- 5.3.3 getAlarmList (M)

getAlarmList **ACTION**

BEHAVIOUR

getAlarmListBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.GetAlarmListInfo;

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.

- *baseObjectClass*

  This parameter carries the object class of the managed object instance identified by the *baseObjectInstance* parameter.

- *baseObjectInstance*

  This parameter carries the DN of a certain managed object instance. Only alarm information instances related to this managed object and its subordinate objects shall be provided.

- *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 the string '(ALIGNMENTEND-alignmentId)' to allow a Manager to recognise that this alarm report is sent due to a previous getAlarmList request.
- If available, the data related to the acknowledgment history (i.e. ackState, ackTime, ackUserId, ackSystemId) are provided in the field *additionalInformation*.

  Further details about the implementation of this operation are provided in the 'Introduction'.";

-- 5.3.4 setComment (O)

setComment **ACTION**

BEHAVIOUR

setCommentBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.SetCommentInfo;

WITH REPLY SYNTAX

TS32-111-4TypeModule.SetCommentReply;

REGISTERED AS {ts32-111AlarmAction 4};

setCommentBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' field contains:

- *alarmReferenceList*

  Contains a list of alarm identifiers to which the comment must be associated.

- *commentUserId*

  Contains the identity of the NM User that invokes this operation.

- *commentSystemId*

  Contains the identity of the NM that invokes this operation.

- *commentText*

  Contains the text of the comment.

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

- *errorAlarmReferenceList*

  List of pair of alarmId and failure reason.

- *status*

  It contains the results of the NM action. Possible values: actionSucceeded (0), actionPartiallyFailed (12) or another value indicating the reason of the error.";

-- 5.3.5 getAlarmIRPVersion (M)

getAlarmIRPVersion **ACTION**

BEHAVIOUR

getAlarmIRPVersionBehaviour;

MODE

CONFIRMED;

WITH REPLY SYNTAX

TS32-111-4TypeModule.GetAlarmIRPVersionReply;

REGISTERED AS {ts32-111AlarmAction 5};

getAlarmIRPVersionBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' field contains no data.

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

- *versionNumbersList*

  It defines a list of Alarm IRP versions supported by the Agent. A list containing no element, i.e. a NULL list means that the concerned Agent doesn't support any version of the Notification IRP.

- *status*

  It contains the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).";

-- 5.3.6 getAlarmIRPNotificationProfile (O)

getAlarmIRPNotificationProfile **ACTION**

BEHAVIOUR

getAlarmIRPNotificationProfileBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.IRPVersionNumber;

WITH REPLY SYNTAX

TS32-111-4TypeModule.GetNotificationProfileReply;

REGISTERED AS {ts32-111AlarmAction 6};

getAlarmIRPNotificationProfileBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' contains the following data:

- *irpVersionNumber*

  This mandatory parameter identifies the Alarm IRP version.

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

- *notificationNameProfile*

  It contains a list of notification names, i.e. a NULL list means that the Alarm IRP doesn't support any notification.

- *notificationParameterProfile.*

  It contains a set of elements, each element corresponds to a notification name and is composed by a set of parameter names.

- *status*

  It contains the results of this action. Possible values: noError (0), error (the value indicates the reason of the error).";

-- 5.3.7 getAlarmIRPOperationProfile (O)

getAlarmIRPOperationProfile **ACTION**

BEHAVIOUR

getAlarmIRPOperationProfileBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.IRPVersionNumber;

WITH REPLY SYNTAX

TS32-111-4TypeModule.GetOperationProfileReply;

REGISTERED AS {ts32-111AlarmAction 7};

getAlarmIRPOperationProfileBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' contains the following data:

- *irpVersionNumber*

  This mandatory parameter identifies the Alarm IRP version.

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

- *operationNameProfile*

  It contains a list of operation names.

- *operationParameterProfile.*

  It contains a set of elements, each element corresponds to an operation name and is composed by a set of parameter names.

- *status*

  It contains the results of this action. Possible values: noError (0), error (the value indicates the reason of the error).";

-- 5.3.8 unacknowledgeAlarms (O)

unacknowledgeAlarms **ACTION**

BEHAVIOUR

unacknowledgeAlarmsBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.AckOrUnackAlarmsInfo;

WITH REPLY SYNTAX

TS32-111-4TypeModule.AckOrUnackAlarmsReply;

REGISTERED AS {ts32-111AlarmAction 8};

unacknowledgeAlarmsBehaviour **BEHAVIOUR**

DEFINED AS

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

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

The 'Action information' field contains the following data:

- *alarmReferenceList*

  This parameter contains a set of MOI (Managed Object Instance) and *notificationIdentifier pair*. Each of them identifies unambiguously in the scope of the Agent an alarm (previously acknowledged by the NM) that have to be now unacknowledged. MOI can be absent if scope of uniqueness of notificationIdentifier is across the IRPAgent.

- *ackUserId*

  It contains the name of the operator who unacknowledged the alarm or a generic name (dependent on the operational concept). It may have also the value NULL. Note that only the user who previously acknowledged the alarm is allowed to un-acknowledge it later.

- *ackSystemId*

  It indicates the management system where the acknowledgment is triggered. It may have also the value NULL. Note that the un-acknowledgement is allowed only at the management system where previously the acknowledgement took place.

The 'Action response' contains the following data:

- *status*

  This parameter contains the results of the NM un-acknowledgement action. Possible values: noError (0, all alarms found and ack state changed according to the manager request), unackPartlySuccessful (some alarms not found / not changeable, see next response parameter), error (value indicates the reason why the complete operation failed).

- *errorAlarmReferenceList*

  This parameter (significant only if *status* = unackPartlySuccessful) contains the list of MOI (Managed Object Instance) and notificationIdentifier pairs of the alarms which could not be unacknowledged and, for each alarm, also the reason of the error. MOI can be absent if scope of uniqueness of notificationIdentifier is across the IRPAgent. ";

-- 5.3.9 clearAlarms (O)

clearAlarms **ACTION**

BEHAVIOUR

clearAlarmsBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.ClearAlarmsInfo;

WITH REPLY SYNTAX

TS32-111-4TypeModule.ClearAlarmsReply;

REGISTERED AS {ts32-111AlarmAction 9};

clearAlarmsBehaviour **BEHAVIOUR**

DEFINED AS

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

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

- *alarmReferenceList*

  This mandatory parameter identifies the alarms to be cleared. Each alarm is identified by the notification identifier of the notification that reported the alarm the first time and, if the notification identifier is not unique across the IRPAgent, by the instance of the managed object that emitted this notification.

- *clearUserId*

  This mandatory parameter identifies the user that has invoked the *clearAlarms* operation.

- *clearSystemId*

  This optional parameter identifies the system on which the IRPManager, where the *clearAlarms* operation has been invoked, is running. This parameter may be absent.

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

- *errorAlarmReferenceList*

  This mandatory parameter identifies alarms that are specified in the *alarmReferenceList*, but which could not be cleared. The alarms are specified by the notification identifier of the notification that reported the alarm the first time and, if required, the instance of the managed object that emitted this notification. In addition to this, the parameter specifies for every alarm that could not be cleared the error reason. If all alarms specified in the *alarmReferenceList* exist and could be cleared, this parameter contains no information. If the operation failed completely due to a general error, this parameter is not significant.

- *status*

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

-- 5.3.10 abortGetAlarmList (M)

abortGetAlarmList **ACTION**

BEHAVIOUR

abortGetAlarmListBehaviour;

MODE

CONFIRMED;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.AbortGetAlarmListInfo;

WITH REPLY SYNTAX

TS32-111-4TypeModule.AbortGetAlarmListReply;

REGISTERED AS {ts32-111AlarmAction 10};

abortGetAlarmListBehaviour **BEHAVIOUR**

DEFINED AS

"This action is invoked by the IRPManager to abort an ongoing alarm alignment process.The M-ACTION request parameter 'Action information' *AbortGetAlarmListInfo* is composed of the following fields:

- *alignmentIdReferenceList*

  This parameter specifies the alarm alignment processes to be aborted. Each alarm alignment process is identified by its *alignmentId*.

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

- *errorAlignmentIdReferenceList*

  This mandatory parameter identifies alarm alignment processes that are specified in the *alignmentIdReferenceList*, but which could not be aborted. In addition to this, the parameter specifies for every process that could not be aborted the error reason. If all alarm alignment processes specified in the *alignmentIdReferenceList* exist and could be aborted, this parameter contains no information. If the operation failed completely due to a general error, this parameter is not significant.

- *status*

  This mandatory parameter provides informations about the result of the operation. If all alarm alignment processes specified in the *alignmentIdReferenceList* exist and are aborted, the value *noError* (0) is returned. If some processes specified do not exist or could not be aborted, the value *abortGetAlarmListPartlySuccessful* (16) is returned. In this case the parameter *errorAlignmentIdReferenceList* provides additional information. If the operation failed completely due to a general error, this parameter returns the error reason.";

-- 5.4 Notifications

-- 5.4.1 notifyAlarmListRebuilt (M)

notifyAlarmListRebuiltR0602 **NOTIFICATION**

BEHAVIOUR

notifyAlarmListRebuiltR0602Behaviour;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.NotifyAlarmListRebuiltInfo

AND ATTRIBUTE IDS

rebuiltObjectClass rebuiltObjectClass,

rebuiltObjectInstance rebuiltObjectInstance;

REGISTERED AS {ts32-111AlarmNotification 10602};

notifyAlarmListRebuiltBehaviour **BEHAVIOUR**

DEFINED AS

"This notification is used by the Agent to inform the NM that the alarm list has been rebuilt.

The 'Event Information' field contains the following data:

- *notificationIdentifier*

  This ITU-T X.721 standardised parameter, together with MOI (Managed Object Instance), unambiguously identifies this notification.

- *rebuiltObjectClass*

  This parameter carries the IRPAgent MOC when the entire AlarmList has been rebuilt. It carries a different MOC when the AlarmList has been partially rebuilt.

- *rebuiltObjectInstance*

  This parameter carries DN of the IRPAgent when the entire AlarmList has been rebuilt. It carries the DN of another MOI when the AlarmList has been partially rebuilt and only the MOIs subordinate of this rebuilt MOI may be affected by this partial rebuilt.

- *reason*

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

- *alarmListAlignmentRequirement*

  This parameter indicates, if the IRPManager has to align its alarm list with the IRPAgent. Absence of this parameter means, that an alignment is required. ";

-- 5.4.2 notifyPotentialFaultyAlarmList (O)

notifyPotentialFaultyAlarmListR0602 **NOTIFICATION**

BEHAVIOUR

notifyPotentialFaultyAlarmListR0602Behaviour;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.NotifyPotentialFaultyAlarmListInfo

AND ATTRIBUTE IDS

potentialFaultyObjectClass potentialFaultyObjectClass,

potentialFaultyObjectInstance potentialFaultyObjectInstance;

REGISTERED AS {ts32-111AlarmNotification 30602};

notifyPotentialFaultyAlarmListR0602Behaviour **BEHAVIOUR**

DEFINED AS

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

The 'Event information' field contains the following data:

- *potentialFaultyObjectClass*

  This parameter specifies together with the parameter *potentialFaultyObjectInstance* the unreliable alarm information instances in the alarm list.

  If this parameter carries the MOC of the IRPAgent, then the entire alarm list is unreliable.

  If this parameter carries the MOC of another MO, then only a part of the alarm list is unreliable. The mechanism for identifying the unreliable part is described below.

- *potentialFaultyObjectInstance*

  This parameter specifies together with the parameter *potentialFaultyObjectClass* the unreliable alarm information instances in the alarm list.

  If *potentialFaultyObjectClass* carries the MOC of the IRPAgent, the this parameter carries the DN of the IRPAgent and the entire alarm list is unreliable.

  If *potentialFaultyObjectClass* carries the MOC of another MO, then this parameter carries the DN of an instance of this class. All alarm information instances representing alarms raised by this MOI and its subordinates may be unreliable in this case.

- *notificationIdentifier*

  This parameter specifies the notification identifier (ITU-T X.733 [5]), which, together with the instance of the object emitting this notification, unambiguously identifies this notification.

- *reason*

  This parameter specifies the reason why the IRPAgent has lost confidence in the integrity of its alarm list and needs to rebuild it.";

-- 5.4.3 notifyAlarmAlignmentEnd (M)

notifyAlarmAlignmentEndR0602 **NOTIFICATION**

BEHAVIOUR

notifyAlarmAlignmentEndR0602Behaviour;

WITH INFORMATION SYNTAX

TS32-111-4TypeModule.NotifyAlarmAlignmentEndInfoR0602

AND ATTRIBUTE IDS

notificationIdentifier "Rec. X.721 | ISO/IEC 10165-2 : 1992":notificationIdentifier,

alignmentId alignmentId,

alarmAlignmentEndStatus alarmAlignmentEndStatus;

REGISTERED AS {ts32-111AlarmNotification 40602};

notifyAlarmAlignmentEndR0602Behaviour **BEHAVIOUR**

DEFINED AS

"This notification is used by the Agent to inform the NM that the alarm alignment related to the current *alignmentId* value is completed or has been aborted before completion by *abortGetAlarmList*.

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.

- *alignmentId*

  The parameter is defined by the Agent (in the getAlarmList response) and identifies unambiguously the current alarm alignment process. It allows the Manager to distinguish between alarm reports sent as consequence of several own alignment requests triggered in parallel.";

-- 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.";

-- 5.5.4 rebuiltObjectClass

rebuiltObjectClass **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX

TS32-111-4TypeModule.ObjectClass;

MATCHES FOR

EQUALITY;

BEHAVIOUR

rebuiltObjectClassBehaviour;

REGISTERED AS {ts32-111AlarmAttribute 40602};

rebuiltObjectClassBehaviour **BEHAVIOUR**

DEFINED AS

"The rebuiltObjectClass attribute type is specified to allow filtering of the rebuiltObjectClass parameter in the notification notifyAlarmListRebuilt.";

-- 5.5.5 rebuiltObjectInstance

rebuiltObjectInstance **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX

TS32-111-4TypeModule.ObjectInstance;

MATCHES FOR

EQUALITY;

BEHAVIOUR

rebuiltObjectInstanceBehaviour;

REGISTERED AS {ts32-111AlarmAttribute 50602};

rebuiltObjectInstanceBehaviour **BEHAVIOUR**

DEFINED AS

"The rebuiltObjectInstance attribute type is specified to allow filtering of the rebuiltObjectInstance parameter in the notification notifyAlarmListRebuilt.";

-- 5.5.6 potentialFaultyObjectClass

potentialFaultyObjectClass **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX

TS32-111-4TypeModule.ObjectClass;

MATCHES FOR

EQUALITY;

BEHAVIOUR

potentialFaultyObjectClassBehaviour;

REGISTERED AS {ts32-111AlarmAttribute 60602};

potentialFaultyObjectClassBehaviour **BEHAVIOUR**

DEFINED AS

"The potentialFaultyObjectClass attribute type is specified to allow filtering of the potentialFaultyObjectClass parameter in the notification notifyPotentialFaultyAlarmList.";

-- 5.5.7 potentialFaultyObjectInstance

potentialFaultyObjectInstance **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX

TS32-111-4TypeModule.ObjectInstance;

MATCHES FOR

EQUALITY;

BEHAVIOUR

potentialFaultyObjectInstanceBehaviour;

REGISTERED AS {ts32-111AlarmAttribute 70602};

potentialFaultyObjectInstanceBehaviour **BEHAVIOUR**

DEFINED AS

"The potentialFaultyObjectInstance attribute type is specified to allow filtering of the rebuiltObjectInstance parameter in the notification notifyPotentialFaultyAlarmList.";

-- 5.5.8 alignmentId

alignmentId **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX

TS32-111-4TypeModule.AlignmentId;

MATCHES FOR

EQUALITY;

BEHAVIOUR

alignmentIdBehaviour;

REGISTERED AS {ts32-111AlarmAttribute 80602};

alignmentIdBehaviour **BEHAVIOUR**

DEFINED AS

"The alignmentId attribute type is specified to allow filtering of the alignmentId parameter in the notification notifyAlarmAlignmentEnd.";

-- 5.5.9 alarmAlignmentEndStatus

alarmAlignmentEndStatus **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX

TS32-111-4TypeModule.AlarmAlignmentEndStatus;

MATCHES FOR

EQUALITY;

BEHAVIOUR

alarmAlignmentEndStatusBehaviour;

REGISTERED AS {ts32-111AlarmAttribute 90602};

alarmAlignmentEndStatusBehaviour **BEHAVIOUR**

DEFINED AS

"The alarmAlignmentEndStatus attribute type is specified to allow filtering of the alarmAlignmentEndStatus parameter in the notification notifyAlarmAlignmentEnd.";

-- 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 is carried by *additionalInformation* in alarm notifications reporting the acknowledgement/unacknowledgement of an alarm or in case these are emitted for alarm synchronisation purposes.. If present, it informs the IRPManager 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 is carried by *additionalInformation* in alarm notifications reporting the acknowledgement/unacknowledgement of an alarm or in case these are emitted for alarm synchronisation purposes.. If present, it informs the IRPManager 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 is carried by *additionalInformation* in alarm notifications reporting the acknowledgement/unacknowledgement of an alarm or in case these are emitted for alarm synchronisation purposes.. If present, it informs the IRPManager 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 is carried by *additionalInformation* in alarm notifications reporting the acknowledgement/unacknowledgement of an alarm or in case these are emitted for alarm synchronisation purposes. If present, it informs the IRPManager about the identifier of the user who acknowledged the present alarm.";

-- 5.6.5 clearUserIdParameter

clearUserIdParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule .AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.UserId;

BEHAVIOUR

clearUserIdParameterBehaviour;

REGISTERED AS {ts32-111AlarmParameter 5};

clearUserIdParameterBehaviour **BEHAVIOUR**

DEFINED AS

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

-- 5.6.6 clearSystemIdParameter

clearSystemIdParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule.AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.UserId;

BEHAVIOUR

clearSystemIdParameterBehaviour;

REGISTERED AS {ts32-111AlarmParameter 6};

clearSystemIdParameterBehaviour **BEHAVIOUR**

DEFINED AS

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

-- 5.6.7 commentsParameter

commentsParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule.AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.AlarmComments;

BEHAVIOUR

commentsParameterBehaviour;

REGISTERED AS {ts32-111AlarmParameter 7};

commentsParameterBehaviour **BEHAVIOUR**

DEFINED AS

"This parameter is carried by *additionalInformation* in alarm notifications reporting the addition of a Comment or in case these are emitted for alarm synchronisation purposes. If present, it informs the IRPManager about the comments assigned to an alarm. Every single comment includes the following data: *commentText, commentTime, commentUserId* and (optionally) *commentSystemId*.";

-- 5.6.8 alarmRaisedTimeParameter

alarmRaisedTimeParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule.AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.AlarmRaisedTime;

BEHAVIOUR

alarmRaisedTimeParameterBehaviour;

REGISTERED AS {ts32-111AlarmParameter 80603};

alarmRaisedTimeParameterBehaviour **BEHAVIOUR**

DEFINED AS

"This parameter is carried by *additionalInformation* in alarm notifications in case these are emitted for alarm synchronisation purposes. If present, it informs the IRPManager about the time the present alarm has been raised.";

-- 5.6.9 alarmClearedTimeParameter

alarmClearedTimeParameter **PARAMETER**

CONTEXT

TS32-111-4TypeModule.AlarmInfo.additionalInformation;

WITH SYNTAX

TS32-111-4TypeModule.AlarmClearedTime;

BEHAVIOUR

alarmClearedTimeParameterBehaviour;

REGISTERED AS {ts32-111AlarmParameter 90603};

alarmClearedTimeParameterBehaviour **BEHAVIOUR**

DEFINED AS

"This parameter is carried by *additionalInformation* in alarm notifications in case these are emitted for alarm synchronisation purposes. If present, it informs the IRPManager about the time the present alarm has been cleared.";

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 notification(10)}



-- Start of 3GPP SA5 own definitions


AbortGetAlarmListInfo ::= SEQUENCE
  {
  alignmentIdReferenceList      SET OF INTEGER
  }

AbortGetAlarmListReply ::= SEQUENCE
  {
  errorAlignmentIdReferenceList      SET OF ErrorInfoAbortGetAlarmList,
  status                        ErrorCauses
  }

AckErrorList ::= SET OF ErrorInfo

AlarmReference ::= SEQUENCE
  {
  moi                     ObjectInstance OPTIONAL, -- absent if scope of uniquness of
                                         -- notificationId is across IRPAgent
  notificationIdentifier       NotificationIdentifier
  }

AckOrUnackAlarmsInfo ::= SEQUENCE
  {
  alarmReferenceList          SET OF AlarmReference,
  ackUserId                  UserId,
  ackSystemId                 SystemId OPTIONAL
  }

AckOrUnackAlarmsReply ::= SEQUENCE
```

```
  {
  status                  ErrorCauses,
  errorAlarmReferenceList     AckErrorList
  }

AckState ::= ENUMERATED
  {
  acknowledged    (0),
  unacknowledged  (1)
  }

AckTime ::= GeneralizedTime

AlarmAlignmentEndStatus ::= ENUMERATED
  {
  successfulCompletion  (0),   -- the alarm alignment has been completed successfully
  aborted               (1),   -- the alarm alignment has been aborted via the invocation
                               -- of the operation abortGetAlarmList
  error               (255)    -- the alarm alignment has been aborted due to an internal error
  }

AlarmChoice ::= ENUMERATED
  {
  allAlarms               (0),
  allActiveAlarms         (1),
  allActiveAndAckAlarms   (2),
  allActiveAndUnackAlarms (3),
  allClearedAndUnackAlarms (4),
  allUnackAlarms          (5)
  }

AlarmClearedTime ::= GeneralizedTime

AlarmComments ::= SET OF SingleAlarmComment
```

AlarmRaisedTime ::= GeneralizedTime

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
  }
```

AlarmListAlignmentRequirement ::= ENUMERATED

```
  {
  alignmentRequired      (0),   -- An alarm alignment is required.
  alignmentNotRequired   (1)    -- An alarm alignment is not required.
  }
```

AlignmentId ::= INTEGER

ClearAlarmsInfo ::= SEQUENCE

```
  {
  alarmReferenceList      SET OF AlarmReference,
  clearUserId             UserId,
  clearSystemId           SystemId OPTIONAL
  }
```

ClearAlarmsReply ::= SEQUENCE

```
  {
  status                    ErrorCauses,
  errorAlarmReferenceList      ClearErrorList
  }
```

ClearErrorList ::= SET OF ErrorInfo

CommentText ::= GraphicString

CommentTime ::= GeneralizedTime

ErrorCauses ::= ENUMERATED

```
{
noError                    (0), -- operation / notification successfully performed
wrongFilter                (1), -- the value of the filter parameter is not valid
wrongAlarmAckState         (2), -- the value of the alarmAckState parameter (e.g.
                           -- getAlarmCount) is not valid
ackPartlySuccessful        (3), -- acknowledgment request partly successful
unackPartlySuccessful      (4), -- unacknowledgment request partly successful
wrongAlarmReference        (5), -- alarm identifier used in the alarm reference list not
                           -- found (e.g. in case of acknowledgement request)
wrongAlarmReferenceList    (6), -- the alarm reference list (e.g. in case of
                           -- acknowledgement request) is empty or completely wrong
alarmAlreadyAck            (7), -- alarm to be acknowledged is already in this state
alarmAlreadyUnack          (8), -- alarm to be acknowledged is already in this state
wrongUserId                (9), -- the user identifier in the unacknowledgement operation
                           -- is not the same as in the previous
                           -- acknowledgementAlarms request
wrongSystemId              (10), -- the system identifier in the unacknowledgement
                           -- operation is not the same as in the previous
                           -- acknowledgementAlarms request
alarmAckNotAllowed         (11), -- current management system not allowed to acknowledge the
                           -- alarm (e.g. due to acknowledgement competence rules)
setCommentPartlySuccessful (12), -- the setComment action partly successful (e.g. some
                           -- alarmId are not in the alarmList)
clearAlarmsPartlySuccessful (13), -- only some alarms to be cleared could be cleared
clearAlarmsNotAllowed      (14), -- current management system not allowed to clear the alarm
clearAlarmsAlarmAlreadyCleared (15), -- alarm to be cleared is already cleared
abortGetAlarmListPartlySuccessful (16), -- only some alarm alignment processes to be aborted
                           -- could be aborted
```

```
  abortGetAlarmListNotAllowed          (17), -- current management system not allowed to abort
                                       -- alarm alignment processes
  abortGetAlarmListProcessNotExist     (18), -- alarm alignment process to be aborted does
                                       -- not exist
  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
  }

ErrorInfoAbortGetAlarmList ::= SEQUENCE
  {
  alignmentId      INTEGER,
  reason           ErrorCauses
  }

GeneralObjectId ::= INTEGER

GetAlarmCountInfo ::= 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
 }

GetAlarmListInfo ::= SEQUENCE
  {
  alarmAckState           AlarmChoice OPTIONAL,
  baseObjectClass         ObjectClass OPTIONAL,      -- ITU-T X.711
  baseObjectInstance      ObjectInstance OPTIONAL,   -- ITU-T X.711
  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

```
NotifyAlarmAlignmentEndInfoR0602 ::= SEQUENCE
  {
  notificationIdentifier      NotificationIdentifier,       -- ITU-T X.721
  alignmentId                 AlignmentId,
  alarmAlignmentEndStatus     AlarmAlignmentEndStatus OPTIONAL
  }

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

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

OperationList ::= SET OF OperationName

OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

ReasonAlarmListRebuilt ::= ENUMERATED

```
{
agentNetworkEntityCommunicationError   (0),
agentRestart                    (1),
indeterminate                   (2)
}
```

ReasonPotentialFaultyAlarmList ::= ENUMERATED

```
{
communicationErrorNEAgent   (0), -- A communication error between a NE and the agent has occured.
agentRestart            (1), -- The agent has restarted and not yet updated its alarm list.
indeterminate           (2)  -- The reasn could not be determined.
}
```

SetCommentInfo ::= SEQUENCE

```
{
alarmReferenceList      SET OF AlarmReference,
commentUserId           UserId,
commentSystemId         [2] SystemId OPTIONAL,
commentText             CommentText
}
```

SetCommentReply ::= SEQUENCE

```
{
```

```
  errorAlarmReferenceList      SET OF ErrorInfo,
  status                       ErrorCauses
  }

SingleAlarmComment ::= SEQUENCE
  {
  commentText          CommentText,
  commentTime          CommentTime,
  commentUserId        UserId,
  commentSystemId      SystemId OPTIONAL
  }

SystemId ::= GraphicString

SupportedAlarmIRPVersions ::= SET OF IRPVersionNumber

UserId ::= GraphicString

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

Annex A (informative): List of assigned Object Identifiers

This annex provides a list with all object identifiers that have been assigned in TS 32.111-4 in Release 5 up to V5.7.0 and in Release 6 up to the latest version. These object identifiers shall not be assigned to new objects.

Basic Object Name	Name and OID of the current TS Version	Name and OIDs of previous TS Versions
	Managed Object Classes	
alarmControl	Name: alarmControlR0602 OID : ts32-111AlarmObjectClass 10602	Name: alarmControl OID : ts32-111AlarmObjectClass 1
	Packages	
alarmControlBasicPackage	Name: alarmControlBasicPackageR0602 OID : ts32-111AlarmPackage 10602	Name: alarmControlBasicPackage OID : ts32-111AlarmPackage 1
alarmCountPackage	Name: alarmCountPackage OID : ts32-111AlarmPackage 2	--
alarmAcknowledgementPackage	Name: alarmAcknowledgementPackage OID : ts32-111AlarmPackage 3	--
alarmUnacknowledgementPackage	Name: alarmUnacknowledgementPackage OID : ts32-111AlarmPackage 4	--
alarmCommentPackage	Name: alarmCommentPackage OID : ts32-111AlarmPackage 5	--
alarmIRPVersionPackage	Name: alarmIRPVersionPackage OID : ts32-111AlarmPackage 6	--
alarmProfilePackage	Name: alarmProfilePackage OID : ts32-111AlarmPackage 7	--
alarmPotentialFaultyAlarmListPackage	Name: alarmPotentialFaultyAlarmListPackageR0602 OID : ts32-111AlarmPackage 80602	Name: alarmPotentialFaultyAlarmListPackage OID : ts32-111AlarmPackage 8
alarmClearPackage	Name: alarmClearPackage OID : ts32-111AlarmPackage 9	--
x721AlarmNotificationsPackage	Name: x721AlarmNotificationsPackage OID : ts32-111AlarmPackage 10	--
	Actions	
acknowledgeAlarms	Name: acknowledgeAlarms OID : ts32-111AlarmAction 1	--
getAlarmCount	Name: getAlarmCount OID : ts32-111AlarmAction 2	--
getAlarmList	Name: getAlarmList OID : ts32-111AlarmAction 3	--
setComment	Name: setComment OID : ts32-111AlarmAction 4	--
getAlarmIRPVersion	Name: getAlarmIRPVersion OID : ts32-111AlarmAction 5	--
getAlarmIRPNotificationProfile	Name: getAlarmIRPNotificationProfile OID : ts32-111AlarmAction 6	--
getAlarmIRPOperationProfile	Name: getAlarmIRPOperationProfile OID : ts32-111AlarmAction 7	--
unacknowledgeAlarms	Name: unacknowledgeAlarms OID : ts32-111AlarmAction 8	--
clearAlarms	Name: clearAlarms OID : ts32-111AlarmAction 9	--
abortGetAlarmList	Name: abortGetAlarmList OID : ts32-111AlarmAction 10	--
	Notifications	
notifyAlarmListRebuilt	Name: notifyAlarmListRebuiltR0602 OID : ts32-111AlarmNotification 10602	Name: notifyAlarmListRebuilt OID : ts32-111AlarmNotification 1
notifyComments	--	Name: notifyComments OID : ts32-111AlarmNotification 2
notifyPotentialFaultyAlarmList	Name: notifyPotentialFaultyAlarmListR0602 OID : ts32-111AlarmNotification 30602	Name: notifyPotentialFaultyAlarmList OID : ts32-111AlarmNotification 3
notifyAlarmAlignmentEnd	Name: notifyAlarmAlignmentEndR0602 OID : ts32-111AlarmNotification 40602	Name: notifyAlarmAlignmentEnd OID : ts32-111AlarmNotification 4
	Attributes	
alarmControlId	Name: alarmControlId OID : ts32-111AlarmAttribute 1	--
alarmsCountSummary	Name: alarmsCountSummary OID : ts32-111AlarmAttribute 2	--
supportedAlarmIRPVersions	Name: supportedAlarmIRPVersions OID : ts32-111AlarmAttribute 3	--

rebuiltObjectClass	Name: rebuiltObjectClass OID : ts32-111AlarmAttribute 40602	--
rebuiltObjectInstance	Name: rebuiltObjectInstance OID : ts32-111AlarmAttribute 50602	--
potentialFaultyObjectClass	Name: potentialFaultyObjectClass OID : ts32-111AlarmAttribute 60602	--
potentialFaultyObjectInstance	Name: potentialFaultyObjectInstance OID : ts32-111AlarmAttribute 70602	--
alignmentId	Name: alignmentId OID : ts32-111AlarmAttribute 80602	--
alarmAlignmentEndStatus	Name: alarmAlignmentEndStatus OID : ts32-111AlarmAttribute 90602	--
Parameters		
ackStateParameter	Name: ackStateParameter OID : ts32-111AlarmParameter 1	--
ackSystemIdParameter	Name: ackSystemIdParameter OID : ts32-111AlarmParameter 2	--
ackTimeParameter	Name: ackTimeParameter OID : ts32-111AlarmParameter 3	--
ackUserIdParameter	Name: ackUserIdParameter OID : ts32-111AlarmParameter 4	--
clearUserIdParameter	Name: clearUserIdParameter OID : ts32-111AlarmParameter 5	--
clearSystemIdParameter	Name: clearSystemIdParameter OID : ts32-111AlarmParameter 6	--
commentsParameter	Name: commentsParameter OID : ts32-111AlarmParameter 7	--
alarmRaisedTimeParameter	Name: alarmRaisedTimeParameter OID : ts32-111AlarmParameter 80603	--
alarmClearedTimeParameter	Name: alarmClearedTimeParameter OID : ts32-111AlarmParameter 90603	--
Name Bindings		

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2000	SA_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	SA_08	SP-000254	0005	--	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	SA_12	SP-010282	0001	--	Alarm IRP: CMIP SS Rel4 - Addition of feature. As SA5 had not reviewed this part, it is submitted to SA#12 for Information only.	3.1.1	--
Sep 2001	SA_13	SP-010470	0001	1	Addition of features	3.1.1	4.0.0
Dec 2001	SA_14	SP-010640	0003	--	Change of qualifier for setComment and notifyComment	4.0.0	4.1.0
Dec 2001	SA_14	SP-010640	0004	--	Addition of missing parameter in notifyComments	4.0.0	4.1.0
Mar 2002	SA_15	SP-020028	0005	--	Addition of "perceivedSeverity" as parameter to "acknowledgeAlarms" operation (CMIP SS)	4.1.0	4.2.0
Mar 2002	SA_15	--	--	--	Automatic upgrade to Rel-5 (no Rel-5 CR)	4.2.0	5.0.0
Jun 2002	SA_16	SP-020283	0007	--	Correction of errors and ambiguities in the Parameter Mapping Tables and ASN.1 Definitions	5.0.0	5.1.0
Jun 2002	SA_16	SP-020284	0008	--	Addition of the parameter alarmListAlignmentRequirement to the notification notifyAlarmListRebuilt in the CMIP SS (32.111-4)	5.0.0	5.1.0
Jun 2002	SA_16	SP-020284	0009	--	Adding the notification notifyPotentialFaultyAlarmList in the CMIP SS (32.111-4)	5.0.0	5.1.0
Jun 2002	SA_16	SP-020284	00100	--	Introduction of SS (32.111-4) to IS (32.111-2) relation and correction of Foreword	5.0.0	5.1.0
Sep 2002	SA_17	SP-020480	0011	--	Alignment with 32.111-2 on Alarm Clearance Functionality	5.1.0	5.2.0
Dec 2002	SA_18	SP-020751	0013	--	Add the additionalInformation parameter in notifyNewAlarms to the Alarm IRP CMIP SS (Alignment with Information Service in Rel-5 32111-2)	5.2.0	5.3.0
Dec 2002	SA_18	SP-020753	0014	--	Addition of Security Alarm Support to the Alarm IRP CMIP SS (Alignment with Information Service in Rel-5 32111-2)	5.2.0	5.3.0
Mar 2003	SA_19	SP-030063	0016	--	Correction to Alarm Comments- alignment with 32.111-1	5.3.0	5.4.0
Mar 2003	SA_19	SP-030138	0017	--	Add missing x721AlarmNotificationsPackage	5.3.0	5.4.0
Mar 2003	SA_19	SP-030138	0018	--	Corrections to GDMO and ASN.1 definitions in the Alarm IRP CMIP SS	5.3.0	5.4.0
Jun 2003	SA_20	SP-030277	0019	--	Correction of Compilation Errors	5.4.0	5.5.0
Jun 2003	SA_20	SP-030277	00200	--	Addition of missing reasons for the emission of notifyAlarmListRebuilt	5.4.0	5.5.0
Sep 2003	SA_21	SP-030416	0022	--	Correction of syntax error in type SetCommentInfo	5.5.0	5.6.0
Dec 2003	SA_22	SP-030627	0023	--	Add missing parts for the support of security alarms	5.6.0	5.7.0
Dec 2003	SA_22	SP-030627	0024	--	Mapping completion of getAlarmList	5.6.0	5.7.0
Dec 2003	SA_22	SP-030629	0025	--	Align operation getAlarmList with the notification notifyAlarmListRebuilt	5.7.0	6.0.0
Jan 2004	--	--	--	--	Editorial (Tables & CMIP code cosmetics)	6.0.0	6.0.1
Mar 2004	SA_23	SP-040120	0026	--	Addition of a method to abort an ongoing alarm alignment process in the asynchronous mode of the operation getAlarmList	6.0.1	6.1.0
Sep 2004	SA_25	SP-040561	0028	--	Align with the IS 32.111-2 the possibility to apply filters to notification parameters	6.1.0	6.2.0
Dec 2004	SA_26	SP-040791	0029	--	Remove redundant ackTime parameter in notifyAckStateChanged	6.2.0	6.3.0
Mar 2005	SA_27	SP-050021	0031	--	Add missing definition of getAlarmList return value - Align with the IS (TS 32.111-2)	6.3.0	6.4.0
Jun 2005	SA_28	SP-050283	0033	--	Clarification for Parallel Alarm alignments	6.4.0	6.5.0

History

Document history		
V6.3.0	December 2004	Publication
V6.4.0	March 2005	Publication
V6.5.0	June 2005	Publication