Technical Specification Group Services and System Aspects
Meeting #12, Stockholm, Sweden, 18-21 June 2001

TSGS#12(01)0282

Source: **SA5**

Title: **Rel4 CR to Fault Management (32.111-series)**

Document for: **Approval**

Agenda Item: **7.5.3**

item	Doc-1st-Level	Doc-2nd-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Version-New	Workitem
A	SP-010282	S5-010343	32.111-1	003		Rel4	**New features for Fault Management**	B	3.2.0	4.0.0	OAM-FM
B	SP-010282	S5-010344	32.111-2	008		Rel4	**Alarm IRP: IS Rel4 - Addition of feature**	B	3.3.0	4.0.0	OAM-FM
C	SP-010282	S5-010345	32.111-3	009		Rel4	**Alarm IRP: CORBA SS Rel4 - Addition of feature**	B	3.5.0	4.0.0	OAM-FM
D	SP-010282	S5-010345	32.111-4	001		Rel4	**Alarm IRP: CMIP SS Rel4 - Addition of feature**	B	3.1.1	4.0.0	OAM-FM
NOTE: As SA5 had not reviewed the implementation of this CR (**D**) in the CMIP Solution Set, compared with the CORBA SS in CR (**C**), the attached revised TS 32.111-4 is submitted to SA#12 for Information only.											

Dependency of CRs

item	Doc-2nd-Level	Spec	CR	Condition for CR approval
A	S5-010343	32.111-1	003	--
B	S5-010344	32.111-2	008	only when **A** above is approved
C	S5-010345	32.111-3	009	only when **CR32.111-3-008**_S5-010391 is approved ***** (see below) & only when **A** and **B** above are approved
D	S5-010345	32.111-4	001	only when **A** and **B** above are approved

***** **CR32.111-3-008_S5-010391**

Doc-1st-Level	Spec	CR	Phase	Subject	Cat	Version-Current	Version-New	Doc-2nd-Level	Status-2nd-Level	Workitem
SP-010039	32.111-3	008	R99	Probable Cause “Intrusion Detection” is missing	F	3.4.0	3.5.0	S5-010391	Agreed	OAM-FM

Fault Management (OAM-FM) - Status at SA#12:
80% for Approval TSG#12, 5% for TSG#13, 15% re-classified Rel5 (Rel4, 95% complete)NOTE: 5% for TSG#13 - because of the CR (**D**) - see CR32.111-4-001_S5-010339

3GPP TSG-SA5 (Telecom Management) **S5-010343**
Meeting #20, Brighton, UK, 28 May - 1 June 2001 **S5F010107**

CR-Form-v3

CHANGE REQUEST

⌘ **32.111-1** **CR** **003** ⌘ rev **-** ⌘ Current version: **3.2.0** ⌘

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

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

Title:	⌘	New features for Fault Management
Source:	⌘	SA5
Work item code:	⌘	OAM-FM
Date:	⌘	01/06/2001
Category:	⌘	**B**
Release:	⌘	Rel4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	To add two new features ‘partial resynchronisation’ or ‘Itf-N distribution of comments associated to faults’.
Summary of change:	⌘	Add requirement corresponding to ‘partial resynchronization’ in clause 5.3.1. Add a requirement to 5.4 stating that the EM may support the NM with a possibility to distribute comments associated to alarms over Itf-N.
Consequences if not approved:	⌘	The new features will not be supported: It will not be possible to resynchronise only a subset of the alarm list (may give performance problems) and comment attached to alarms will only be handled locally by each IRPManager.

Clauses affected:	⌘	5.3.1, 5.4		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications **X** O&M Specifications	⌘	**32.111-2, 32.111-3, 32.111-4**
Other comments:	⌘			

5.3.1 Retrieval of current alarm information on NM request

The present document defines a flexible, generic synchronisation procedure, which fulfils the following requirements:

- The alarm information provided by means of the synchronisation procedure shall be the same (at least for the mandatory parameters) as the information already available in the alarm list. The procedure shall be able to assign the received synchronisation-alarm information to the correspondent requests, if several synchronisation procedures triggered by one NM run at the same time.
- The procedure shall allow the NM to trigger the start at any time and to recognise unambiguously the end and the successful completion of the synchronisation.
- The procedure shall allow the NM to discern easily between an "on-line" (spontaneous) alarm report and an alarm report received as consequence of a previously triggered synchronisation procedure.
- The procedure shall allow the NM to specify filter criteria in the alignment request (e.g. for a full network or only a part of it.
- The procedure shall support connections to several NM and route the alignment-related information only to the requesting NM.
- During the synchronisation procedure new ("real-time") alarms may be sent at any time to the managing NM.
- If the EM loses confidence to its alarm list and rebuilds it, then the EM shall indicate to the NM that the alarm list have been rebuilt. If the rebuild of the alarm list only concerns alarms for e.g one NE then the EM may indicate that it is only that part of the alarm list that has been rebuilt. In the latter case the NM may use the knowledge that only a specific subset of the alarm list have been rebuilt to perform a partial resynchronization using filters.

If applicable, an alarm synchronisation procedure may be aborted by the requesting NM

5.4 Co-operative alarm acknowledgement on the Itf-N

The acknowledgement of an alarm is a maintenance function that aids the operators in his day to day management activity of his network. An alarm is acknowledged by the operator to indicate he has started the activity to resolve this specific problem. In general a human operator performs the acknowledgement, however a management system (NM or EM) may automatically acknowledge an alarm as well.

The alarm acknowledgement function requires that:

a) All involved OSs have the same information about the alarms to be managed (including the current responsibility for alarm handling).

b) All involved OSs have the capability to send and to receive acknowledgement messages associated to previous alarm reports.

A co-operative alarm acknowledgement means that the acknowledgement performed at EM layer is notified at NM layer and vice versa, thus the acknowledgement-related status of this alarm is the same across the whole management hierarchy. The OSs often gives the operator(s) a possibility to add a comment to an alarm. An OS can have the capability to record more than one comment for each alarm. To make the same alarm look the same in all OSs subscribing to the alarm, it should be possible to distribute the recorded comments in the same way as for the acknowledgement information.

The co-operative alarm acknowledgement on Itf-N shall fulfil the following requirements:

- Acknowledgement messages may be sent in both directions between EMs and NM, containing the following information:
  - Correlation information to the alarm just acknowledged.
  - Acknowledgement history data, including the current alarm state (active | cleared), the time of alarm acknowledgement and, as configurable information, the management system (EM | NM) and the operator in charge of acknowledgement (the parameter operator name or, in case of auto-acknowledgement, a generic system name).
  - Acknowledgement notifications sent to NM shall be filtered with the same criteria applied to the alarms.
- Taking into account the acknowledgement functionality, the above described synchronisation procedure for retrieval of current alarm information on NM request may be extended. Additionally to the requirements defined in subclause 5.3.1, this extended synchronisation procedure relates not only to the active, but also to the "cleared and not acknowledged" alarms, which have still to be managed by the EM.

3GPP TSG-SA5 (Telecom Management) **S5-010344**
Meeting #20, Brighton, UK, 28 May - 1 June 2001 **S5F010104**

CR-Form-v3

CHANGE REQUEST

⌘ **32.111-2** **CR** **008** ⌘ rev **-** ⌘ Current version: **3.3.0** ⌘

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

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

Title:	⌘	Alarm IRP: IS Rel4 - Addition of feature
Source:	⌘	SA5
Work item code:	⌘	OAM-FM
Date:	⌘	01/062001
Category:	⌘	**B**
Release:	⌘	Rel4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 *(GSM Phase 2)*
R96 *(Release 1996)*
R97 *(Release 1997)*
R98 *(Release 1998)*
R99 *(Release 1999)*
REL-4 *(Release 4)*
REL-5 *(Release 5)*

Reason for change: ⌘ Use the new IS document template to document IS features/capabilities. The use of new template eliminates many ambiguities that exist in R99 IS specification. Include specification of new features as specified in 32 111-1. Reflect modification of Release 99 feature.

Summary of change: ⌘ Besides the use of the new template to document the IS features/capabilities, the following lists the technical changes in this version compared to the Release 99 version.

Semantics for Probable Cause

The following quoted text is removed from the cell row "probableCause" and column "comment" of Table 13: Parameter-Attributes of `alarmInformationBody` of Release 99 IS.

"This list is extensive. It is recommended that `IRPAgent` should use the list as is and not to extend it. It is noted that `IRPAgent` can privately (outside the scope of this IRP) define values for `specificProblem` that provides semantics not conveyed by `probableCause`. A special probable cause value (SS specific, e.g. –1) indicates that this alternative is valid. This attribute value shall be single-value and of simple type such as integer or string. See definition in ITU-T Recommendation X.733 [2] clause 8.1.2.1.

"

Add Semantics to Additional Text

The following quoted text is added to the cell row "additionalText" and column "comment" of the Table 13: Parameter-Attributes of `alarmInformationBody of Release 99 IS.`

"

It may provide the identity of the NE (e.g. RNC, Node-B) from which the alarm has been originated. It corresponds to the "user label" attribute of the object class representing the NE in the Generic Network Resource Model.

"

Add optional `setComment()` and `notifyComment()`

An optional setComment() operation is added to allow IRPManager to make comment on one or more AlarmInformation instances. An optional notifiyComment() notification is added to allow IRPAgent to emit notification regarding new comment made to AlarmInformation to subscribed IRPManagers. IRPAgent shall support this notifyComment() if it supports the optional setComment().

Remove the use of Event type and Extended Event Type

Release 99 use the term Event Type to carry semantics of ITU-T defined event type such as "communication error". It uses the term Extended Event Type to carry semantics of 3GPP defined and not ITU-T defined types of event such as "notification of alarm list rebuilt". Release 99 uses Event Type and Extended Event Type in parameters of notifications.

Release 4 removes the use of Event Type and Extended Event Type (as parameters of notifications).

Release 4 uses a notification parameter called `notificationType` that shall be present in all notification. Release 4 uses another notification parameter called `alarmType` that shall be present in all notifications carrying alarm information. The `notificationType` carries semantics of "notification of new alarm", "notification of AlarmList rebuilt", "notification of alarm cleared", etc. A value of notificationType identifies one specific notification such as notifyNewAlarm(), notifyClearedAlarm, notifyObjectCreation, etc. The `alarmType` are the ITU-T defined event types that are related to alarm condition such as "communication alarm".

Support Partial Alarm List Rebuilt

In Release 99, the IRPAgent has no possibility to indicate the alarm information of a specific MO and its subordinate MOs has been rebuilt. Release 99 IRPManager has no way of knowing which part of the MO sub-tree whose alarm information is no longer reliable. The IRPManager must invoke getAlarmList() to retrieve the whole alarm list covering all MOs.

In Release 4, the IRPAgent uses the MOC/MOI (of the notifyAlarmListRebuilt) to indicate that the alarm information of the MO (indicated by the MOC/MOI) and its subordinate MOs has been rebuilt. On reception of this notifyAlarmListRebuilt(), the IRPManager is aware that the alarm information of a specific MO sub-tree has been rebuilt and can act accordingly.

Add optional operations `getOperationProfile()` and `getNotificationProfile()`

This set of optional operations allows IRPManager, at run time, to identify IRPAgent supported operations and the corresponding supported parameters. It also allows the IRPManager to identify the IRPAgent supported notifications and their supported parameters.

These operations are specified for ManagedGenericIRP IOC in 32.112 Generic IRP Management: IS. They are inherited by this IS.

Note on presentation of the above changes:

The use of the new IS specification template necessitates extensive non-technical textual changes to the Alarm IRP: IS Release 99. The tracking of these changes proved to be a bit much for the word processor. It is difficult to maintain the change bar, indicating changes to the old text, while maintaining all the new text without hanging the word processor.

Consequently, in the Alarm IRP: IS Release 4 draft, you will find only the clean text and no change bar and no deleted text (as file 32111-2-331.doc).

Consequences if not approved:	⌘	Ambiguities in R99 remain. Addition of new feature and functional modification of R99 features will not be realised.

Clauses affected:	⌘	All clauses.		
Other specs affected:	⌘		Other core specifications	⌘
			Test specifications	
		X	O&M Specifications	**32.111-3, 32.111-4**
Other comments:	⌘	This CR is dependent on the approval of CR32.111-1-003_S5-010343.doc		

3GPP TS 32.111-2 V3.3.1 (2001-06)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Telecommunication Management;
Fault Management;
Part 2: Alarm Integration Reference Point: Information Service
(Release 4)

Keywords
Fault Management, Alarms

3GPP

Postal address

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

Internet
http://www.3gpp.org

Contents

Foreword

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

The present document is part 2 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects, as identified below:

Part 1: "3G Fault Management Requirements";

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

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

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

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

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

Introduction

The present document is part of a set of TSs which describe the requirements and information model necessary for the Telecommunication Management (TM) of 3G systems. The TM principles and TM architecture are specified in 3GPP TS 32.101 [6] and 3GPP TS 32.102 [7].

A 3G system is composed of a multitude of Network Elements (NE) of various types and, typically, different vendors inter-operate in a co-ordinated manner in order to satisfy the network users' communication requirements.
The occurrence of failures in a NE may cause a deterioration of this NE's function and/or service quality and will, in severe cases, lead to the complete unavailability of the NE. In order to minimise the effects of such failures on the Quality Of Service (QOS) as perceived by the network users it is necessary to:

- detect failures in the network as soon as they occur and alert the operating personnel as fast as possible;
- isolate the failures (autonomously or through operator intervention), i.e. switch off faulty units and, if applicable, limit the effect of the failure as much as possible by reconfiguration of the faulty NE/adjacent NEs;
- if necessary, determine the cause of the failure using diagnosis and test routines; and,
- repair/eliminate failures in due time through the application of maintenance procedures.

This aspect of the management environment is termed "Fault Management" (FM). The purpose of FM is to detect failures as soon as they occur and to limit their effects on the network Quality of Service (QOS) as far as possible. The latter is achieved by bringing additional/redundant equipment into operation, reconfiguring existing equipment/NEs, or by repairing/eliminating the cause of the failure.

Fault Management (FM) encompasses all of the above functionalities except commissioning/decommissioning of NEs and potential operator triggered reconfiguration (these are a matter of Configuration Management (CM), see [13]).

FM also includes associated features in the Operations System (OS), such as the administration of alarm list, the presentation of operational state information of physical and logical devices/resources/functions, and the provision and analysis of the alarm and state history of the network.

1 Scope

The present document defines the Alarm Integration Reference Point (IRP) Information Service (IS), which addresses the alarm surveillance aspects of Fault Management (FM), applied to the N Interface [???].

The purpose of the AlarmIRP is to define an interface through which a "system" (typically a Network Element Manager or a Network Element) can communicate alarm information for its managed objects to one or several Manager Systems (typically Network Management Systems).

The Alarm IRP IS defines the semantics of alarms and the interactions visible across the reference point in a protocol neutral way. It defines the semantics of the operations and notifications visible in the IRP. It does not define the syntax or encoding of the operations, notifications and their parameters.

2 References

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

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

[1] ITU-T Recommendation Q821: "Stage 2 and Stage 3 description for the Q3 interface – Alarm surveillance".

[2] ITU-T Recommendation X.733 (02/92): "Information technology - Open Systems Interconnection - Systems management: Alarm Reporting Function".

[3] ITU-T Recommendation X.721: "Information Technology - Open Systems Interconnection - Structure Of Management Information: Definition Of Management Information".

[4] GSM 12.11 version 6.2.0 Release 1997: "Fault management of the Base Station System (BSS)".

[5] 3GPP TS 32.301-2: "Notification IRP: Information Service".

[6] 3GPP TS 32.101: "3G Telecom Management principles and high level requirements".

[7] 3GPP TS 32.102: "3G Telecom Management architecture".

[8] 3GPP TS 32.300: "Name Convention for Managed Objects".

[9] 3GPP TS 32.111-1: "3G Fault Management".

[10] 3GPP TS 32.620-2: "Generic Network Resource Model (NRM), Release 4".

[11] ITU-T Recommendation M.3100 (07/95): "Generic network information model".

[12] ITU-T Recommendation X.720: "Management Information Model".

[13] 3GPP TS 32.620-2 : "Generic Network Resources IRP : Network Resource Model".

[14] 3GPP TS 32.112-2 : "Generic IRP Management : Information Service".

3 Definitions and abbreviations

3.1 Definitions

In addition to the terms and definitions defined in 3GPP TS 32.111-1 [9], the following definitions apply to this document:

Event: It is an occurrence that is of significance to network operators, the NEs under surveillance and Network Management applications. Events do not have state.

IRPManager: defined in 3GPP TS 32.102 [7].

IRP document version number string: The IRP document version number (sometimes called "**IRPVersion**") string is used to identify a particular IRP solution set specification. It is derived using the following rule. Take the 3GPP document version number on the front page of the solution set specification, such as "3GPP TS 32.106-3 V3.2.0 (2000-12)". Discard the leading "3GPP TS ". Discard all characters after and including the last period. Eliminate leading and trailing spaces. Reduce multiple consecutive spaces with one space. Express the resultant in a string. Capitalised the string. For example, if the 3GPP document version number is "3GPP TS 32.106-3 V3.2.0 (2000-12)", then the IRP document version number shall be "32.106 V3.2".

Matching-Criteria-Attributes: It identifies a set of ITU-T Recommendation X.733 [2] defined attributes. Notifications carrying identical values for these attributes are considered to be carrying alarm information related to (a) the same network resource and (b) the same alarmed condition. The matching-criteria-attributes are: `objectInstance, eventType, probableCause and specificProblem`, if present.

Notification: It refers to the transport of events from `IRPAgent` to `IRPManager`. In this IRP, notifications are used to carry alarm information from `IRPAgent` to `IRPManager`.

`IRPAgent`: defined in 3GPP TS 32.102 [7].

3.2 Abbreviations

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

CCITT	The International Telegraph and Telephone Consultative Committee
CMIP	Common Management Information Protocol
DN	Distinguished Name
EM	Element Manager
IOC	Information Object Class
IRP	Integration Reference Point
ITU-T	International Telecommunication Union, Telecommunication Sector
MO	Managed Object
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NM	Network Manager

OS	Operations System
OSI	Open System Interconnection
RDN	Relative Distinguished Name
SS	Solution Set
UML	Unified Modelling Language

4 Basic aspects

4.1 Background

Integration Reference Points (IRPs) are the means within 3G Telecom Management (TM) for specifying interoperable points of information exchange between systems and applications.

3GPP TS 32.101 [6] and 32.102 [7] contain background and introductory information about the IRP concept.

4.2 System Overview

The following figures identify system contexts of this document in terms of implementations called `IRPAgent` and `IRPManager`.

"`IRPManager`" depicts a process that interacts with `IRPAgent` for the purpose of receiving alarms via this IRP. Examples of `IRPManager` can be Network Management Systems and Alarm viewing devices (such as a local craft terminal). `IRPAgent` implements and supports the Alarm IRP.
`IRPAgent` can be one Network Element (NE) (see figure 2) or it can be one Element Manager (EM) with one or more NEs (see figure 1). In the latter case, the interfaces (represented by a thick dotted line) between the EM and the NEs are not subject of this IRP. Whether EM and NE share the same hardware system is not relevant to this document either. By observing the interaction across the Alarm IRP, one cannot deduce if EM and NE are integrated in a single system or if they run in separate systems.

As indicated in figure 1 and figure 2, the subject document need to be complemented with the Notification IRP [5] (to allow `IRPManager` to subscribe to notifications issued by `IRPAgent` and (optionally) product-specific resource models describing the MOs maintained by the `IRPAgent`).

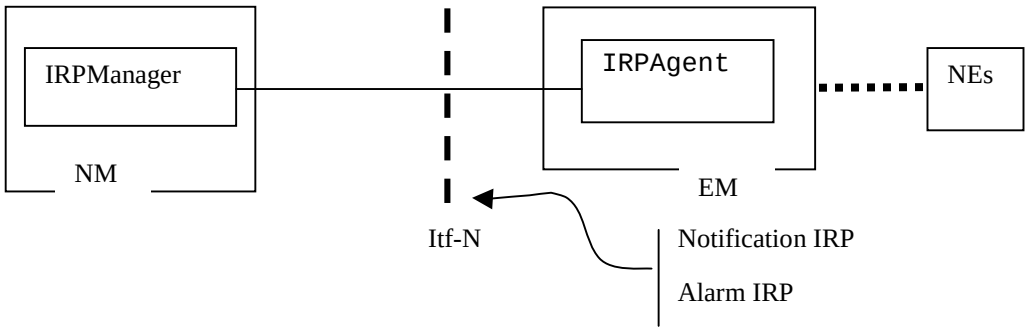


Figure 1: System Context A

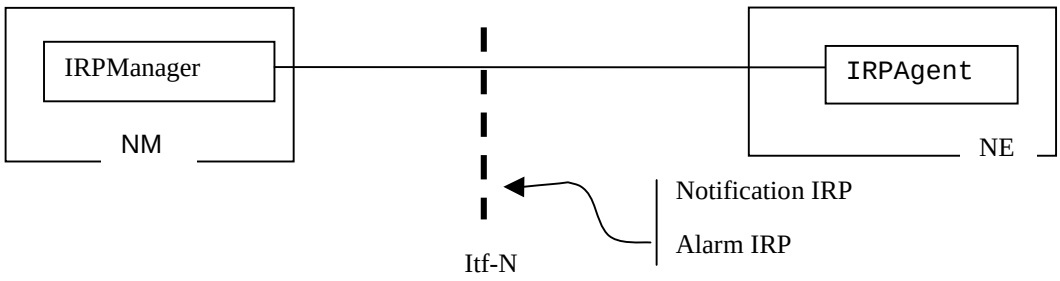


Figure 2: System Context B

5 Information Object Classes

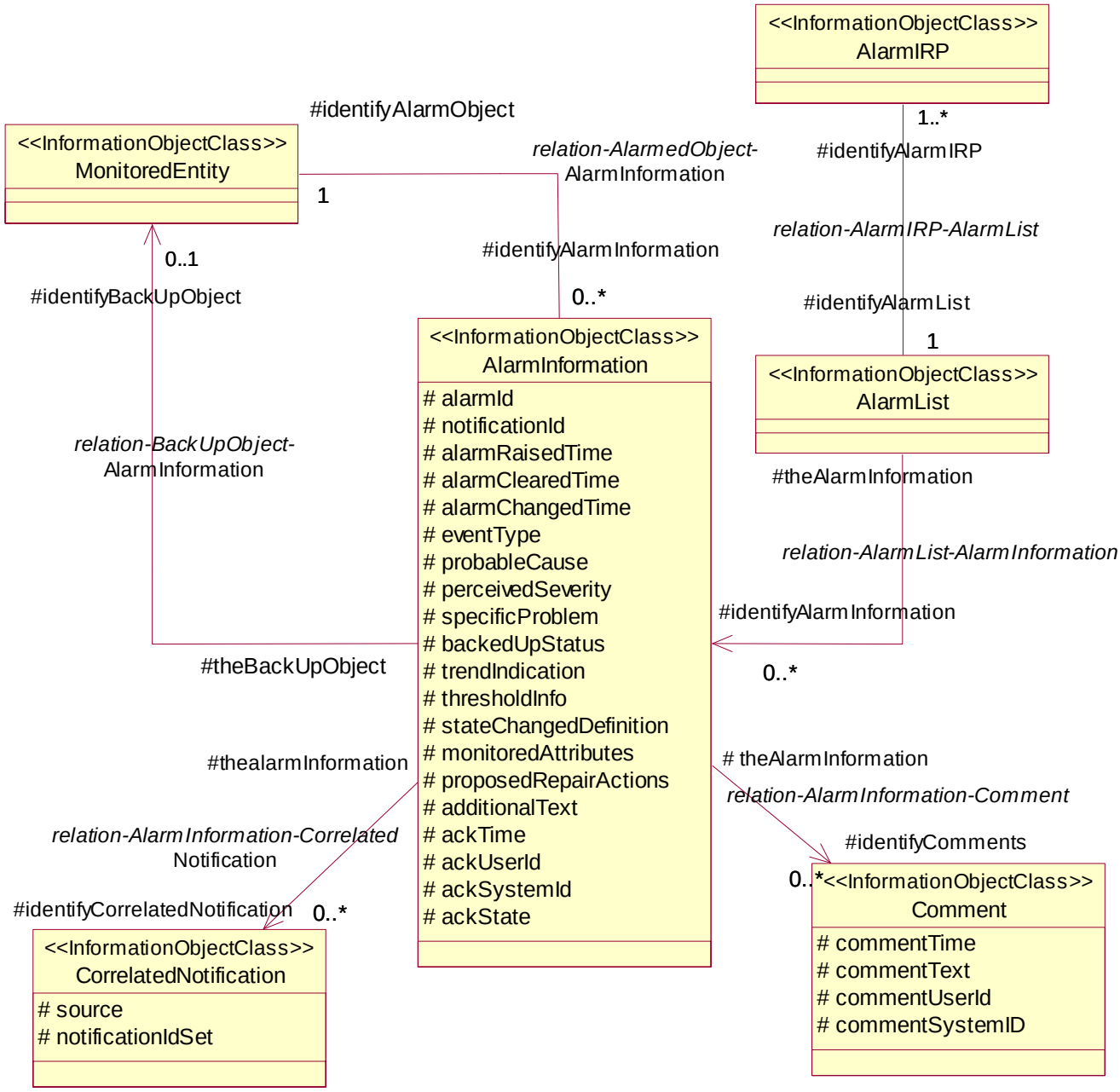
5.1 Information entities imported and local label

Label reference	Local label
32.301-2 [5], information object class, NotificationIRP	`NotificationIRP`
32.301-2 [5], interface, `notificationIRPNotification`	notificationIRPNotification
32.620-2 [10], information object class, `IRPAgent`	`IRPAgent`
32.620-2 [10], information object class, `ManagedGenericIRP`	`ManagedGenericIRP`

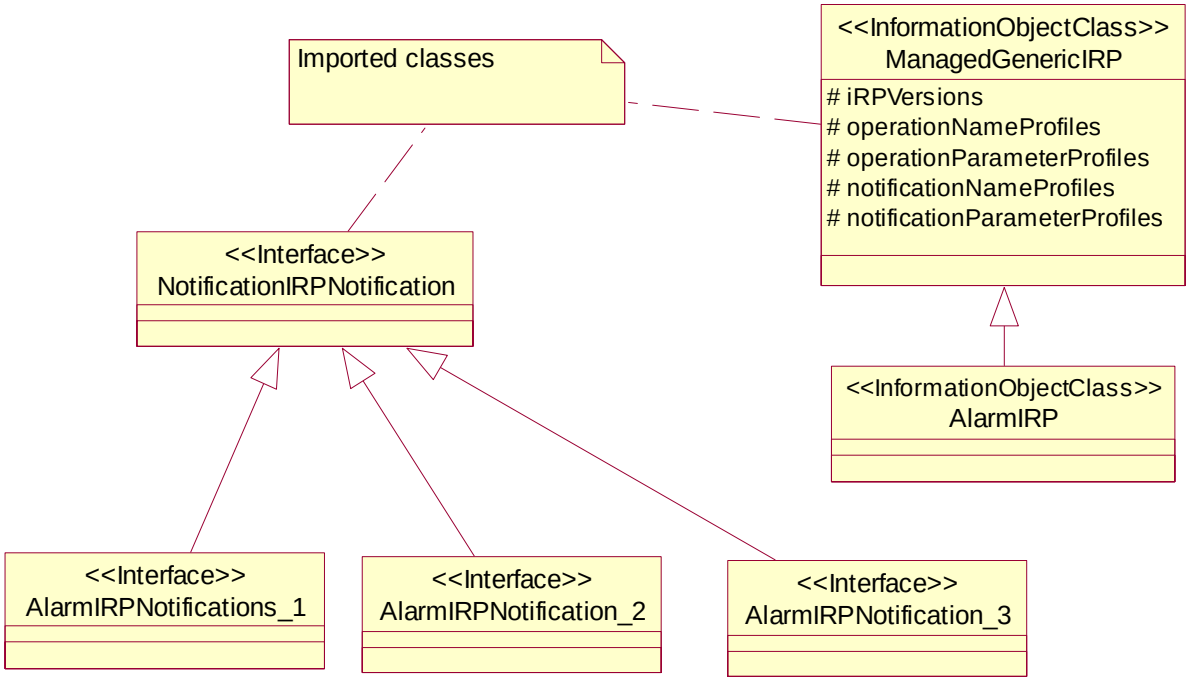
5.2 Class diagram

This sub-clause introduces the set of information object classes (IOCs) that encapsulate information within the `IRPAgent`. The intent is to identify the information required for the AlarmIRP Agent implementation of its operations and notification emission. This sub-clause provides the overview of all support object classes in UML. Subsequent sub-clauses provide more detailed specification of various aspects of these support object classes.

5.2.1 Attributes and relationships



5.2.2 Inheritance



5.3 Information Object Class Definitions

5.3.3 `AlarmInformation`

5.3.1.1 Definition

`AlarmInformation` contains information about alarm condition of an alarmed `MonitoredEntity`.

One `IRPAgent` is related to at most one `AlarmList`. The `IRPAgent` or its related AlarmIRP or the related `AlarmList` assigns an identifier, called `alarmId`, to each `AlarmInformation` in the `AlarmList`. An `alarmId` unambiguously identifies one `AlarmInformation` in the `AlarmList`.

5.3.1.2 Attribute

Attribute name	Support Qualifier
`alarmId`	M
`notificationId (note 1)`	M
`alarmRaisedTime`	M
`alarmClearedTime`	M
`alarmChangedTime`	O
`eventType`	M
`probableCause`	M
`perceivedSeverity`	M
`specificProblem`	O
`backedUpStatus`	O
`trendIndication`	O
`thresholdInfo`	O
`stateChangedDefinition`	O
`monitoredAttributes`	O
`proposedRepairActions`	O
`additionalText`	O

ackTime	M
ackUserId	M
ackSystemId	O
ackState	M

Note 1: This attribute may be "retired/removed" in Release 5 when Log IRP is introduced. Its removal implies that information carried in this attribute is no longer made accessible to IRPManager via the getAlarmList().

5.3.1.3 State diagram

Alarms have states. The alarm state information is captured in AlarmInformation in AlarmList.

The solid circle icon represents the Start State. The double circle icon represents the End State. In this state, the alarm is Cleared and acknowledged. The AlarmInformation shall not be accessible via the IRP and is removed from the AlarmList.

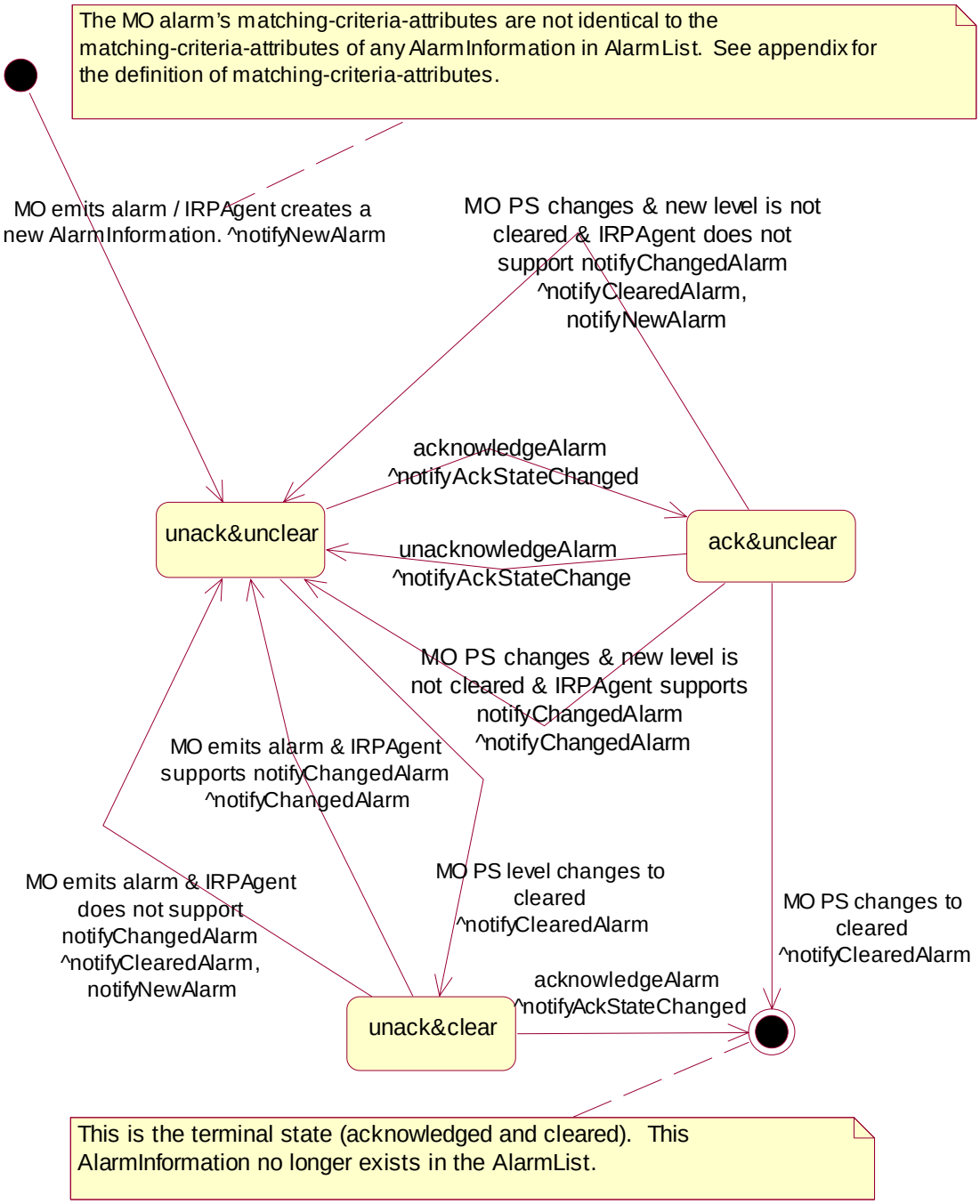
Note the state diagram uses " X / Y ^ Z " to label the arc that indicates state transition. The meanings of X, Y and Z are:

- X identifies the triggering event
- Y identifies the action of IRPAgent because of the triggering event
- Z is the notification to be emitted by IRPAgent because of the triggering event

Note that acknowledgeAlarm^notifyAckStateChanged and the unacknowledgeAlarm^notifyAckStateChange refer to cases when the request of the IRPManager is successful for the AlarmInformation concerned. They do not refer to the cases when the request is a failure since in the failure cases, no state transition would occur.

Note that, to reduce cluttering to the diagram, the setComment^notifyComment is not included in the figure. One transition should be applied from unack&unclear to itself. Similarly, another transition should be applied from ack&unclear to itself. Another one is from unack&clear to itself.

Note that "PS" used in the state diagram stands for "perceived severity".


The MO alarm's matching-criteria-attributes are not identical to the matching-criteria-attributes of any AlarmInformation in AlarmList. See appendix for the definition of matching-criteria-attributes.
MO emits alarm / IRPAgent creates a new AlarmInformation. ^notifyNewAlarm
MO PS changes & new level is not cleared & IRPAgent does not support notifyChangedAlarm ^notifyClearedAlarm, notifyNewAlarm
acknowledgeAlarm ^notifyAckStateChanged
unack&unclear
unacknowledgeAlarm ^notifyAckStateChange
ack&unclear
MO PS changes & new level is not cleared & IRPAgent supports notifyChangedAlarm ^notifyChangedAlarm
MO emits alarm & IRPAgent supports notifyChangedAlarm ^notifyChangedAlarm
MO emits alarm & IRPAgent does not support notifyChangedAlarm ^notifyClearedAlarm, notifyNewAlarm
MO PS level changes to cleared ^notifyClearedAlarm
MO PS changes to cleared ^notifyClearedAlarm
acknowledgeAlarm ^notifyAckStateChanged
unack&clear
This is the terminal state (acknowledged and cleared). This AlarmInformation no longer exists in the AlarmList.

5.3.2 AlarmList

5.3.2.1 Definition

IRPAgent maintains an AlarmList. It contains all currently active alarms (i.e. AlarmInformation whose perceivedSeverity is not Cleared) and alarms that are Cleared but not yet acknowledged.

5.3.2.2 Attribute

There is no additional attribute defined for this IOC besides those inherited.

5.3.3 AlarmIRP

5.3.3.1 Definition

AlarmIRP is the representation of the alarm management capabilities specified by this specification. This IOC inherits from ManagedGenericIRP IOC specified in [14].

5.3.4 Comment

5.3.4.1 Definition

Comment contains commentary and associated information such as the time when the commentary is made.

5.3.4.2 Attribute

Attribute Name	Support Qualifier
commentTime	M
commentText	M
commentUserId	M
commentSystemId	O

5.3.5 CorrelatedNotification

5.3.5.1 Definition

It identifies one MonitoredEntity. For that MonitoredEntity identified, a set of notification identifiers is also identified. One or more CorrelatedNotification instances can be related to an AlarmInformation. In this case, the information of the AlarmInformation is said to be correlated to information carried in the notifications identified by the CorrelatedNotification instances. See further definition of correlated notification in ITU-T Recommendation X.733 [2] clause 8.1.2.9.

The meaning of correlation is dependent on the type of notification itself. See the comment column of the correlatedNotification input parameter for each type of notification, such as notifyNewAlarm.

Notification carries AlarmInformation. The AlarmInformation instances referred to by the correlatedNotification may or may not exist in the AlarmList. For example, the AlarmInformation carried by the identified notification may have been acknowledged and Cleared and therefore, no longer exist in the AlarmList.

5.3.5.2 Attribute

Attribute Name	Support Qualifier
source	M
notificationIdSet	M

5.3.6 MonitoredEntity

5.3.6.1 Definition

It encapsulates a subset of information of an IOC that can emit alarms. It can also encapsulate a subset of information of an IOC that serves as the back up object.

5.3.6.2 Attribute

There is no attribute for this IOC.

5.4 Information relationships definition

5.4.1 relation-AlarmIRP-AlarmList (M)

5.4.1.1 Definition

This represents the relationship between AlarmIRP and AlarmList.

5.4.1.2 Role

Name	Definition
identifyAlarmIRP	It represents the capability to obtain the identities of one or more AlarmIRP.
identifyAlarmList	It represents the capability to obtain the identify of one AlarmList.

5.4.1.3 Constraint

There is no constraint for this relationship.

5.4.2 relation-AlarmList-AlarmInformation (M)

5.4.2.1 Definition

This represents the relationship between AlarmList and AlarmInformation.

5.4.2.2 Role

Name	Definition
theAlarmInformation	It represents the AlarmInformation.
identifyAlarmInformation	It represents a capability to obtain the information contained in AlarmInformation.

5.4.2.3 Constraint

Name	Definition
inv_ hasAlarmInformation1	No AlarmInformation playing the role of theAlarmInformation shall have its perceivedSeverity = "cleared" and its ackState = "acknowledged".
inv_ hasAlarmInformation2	The alarmId of all AlarmInformation instances playing the role of theAlarmInformation are distinct.

5.4.3 relation-AlarmInformation-Comment (M)

5.4.3.1 Definition

This represents the relationship between AlarmInformation and Comment.

5.4.3.2 Role

Name	Definition
theAlarmInformation	It represents the AlarmInformation.
identifyComment	It represents a capability to obtain the information contained in Comment.

5.4.3.3 Constraint

There is no constraint.

5.4.4 relation-AlarmInformation-CorrelatedNotification (M)

5.4.4.1 Definition

This represents the relationship between AlarmInformation and CorrelatedNotification.

5.4.4.2 Role

Name	Definition
theAlarmInformation	It represents the AlarmInformation.
identifyCorrelatedNotification	It represents a capability to obtain the information contained in CorrelatedNotification.

5.4.4.3 Constraint

There is no constraint.

5.4.5 relation-AlarmedObject-AlarmInformation (M)

5.4.5.1 Definition

This represents the relationship between MonitoredEntity and AlarmInformation.

5.4.5.2 Role

Name	Definition
identifyAlarmedObject	It represents the capability to obtain the identification, in terms of objectClass and objectInstance, of alarmed network resource.
identifyAlarmInformation	It represents the capability to obtain the identities of AlarmInformation.

5.4.5.3 Constraint

Name	Definition
inv_relation-AI-ME	All AlarmInformation involved in this relationship with the same MonitoredEntity shall have at least one different value in the following attributes: eventType, probableCause and specificProblem.

5.4.6 relation-backUpObject-AlarmInformation (O)

5.4.6.1 Definition

The relationship represents the relationship between AlarmInformation and the backUpObject.

5.4.6.2 Role

Name	Definition
identifyBackUpObject	It represents a capability to obtain the identification, in terms of objectClass and objectInstance, of the backUpObject.

5.4.6.3 Constraint

Name	Definition
inv_identifyBackUpObject	This relationship is present if and only if the AlarmInformation.backedUpStatus attribute is present and is indicating true.

5.5 Information attribute definition

5.5.1 Definition and legal values

Name	Definition	Legal Values
alarmId	It identifies one AlarmInformation in the AlarmList.	
notificationId	It identifies the notification that carries the AlarmInformation.	
alarmRaised	It indicates the date and time when the alarm is first raised by	All values indicating valid time.

Name	Definition	Legal Values
Time	the alarmed resource.	
alarmChanged Time	It indicates the last date and time when the AlarmInformation is changed by the alarmed resource. Changes to AlarmInformation caused by invocations of the IRPManager would not change this date and time.	All values indicating valid time.
alarmCleared Time	It indicates the date and time when the alarm is Cleared.	All values indicating valid time.
eventType	It indicates the type of event. See Annex A for information on event type.	See Annex A.
probableCause	It qualifies alarm and provides further information than eventType. See Annex B for a complete listing.	See Annex B.
perceived Severity	It indicates the relative level of urgency for operator attention.	Critical, Major, Minor, Warning, Indeterminate, Cleared: see ITU-T Recommendation X.733 [2]. This IRP does not recommend the use of indeterminate.
specific Problem	It provides further qualification on the alarm than probableCause. This attribute value shall be single-value and of simple type such as integer or string. See definition in ITU-T Recommendation X.733 [2] clause 8.1.2.2.	Provided by vendor.
backedUp Status	It indicates if an object (the MonitoredEntity) has a back up. See definition in ITU-T Recommendation X.733 [2] clause 8.1.2.4.	All values that carry the semantics of backedUpStatus defined by ITU-T X.733 [2] clause 8.1.2.4.
trend Indication	It indicates if some observed condition is getting better, worse, or not changing.	"Less severe", "no change", "more severe": see definition in ITU-T Recommendation X.733 [2] clause 8.1.2.6.
thresholdInfo	It indicates the direction of threshold crossing.	"Up direction", "down direction": see definitions in ITU-T Recommendation X.733 [2] clause 8.1.2.7.
stateChange Definition	It indicates MO attribute value changes. See definition in ITU-T Recommendation X.733 [2] clause 8.1.2.10.	
monitored Attributes	It indicates MO attributes whose value changes are being monitored. See definition in ITU-T Recommendation X.733 [2] clause 8.1.2.11.	
proposed RepairActions	It indicates proposed repair actions. See definition in ITU-T Recommendation X.733 [2] clause 8.1.2.12.	
additional Text	It carries semantics that is outside the scope of this IRP specification. It may provide the identity of the NE (e.g. RNC, Node-B) from which the alarm has been originated. It corresponds to the "user label" attribute of the object class representing the NE in the Generic Network Resource Model [10]. It can contain further information on the alarm.	N/A
ackTime	It identifies the time of last operation acknowledgeAlarms or unacknowledgeAlarms.	All values that indicate valid time that are later than that carried in alarmRaisedTime.

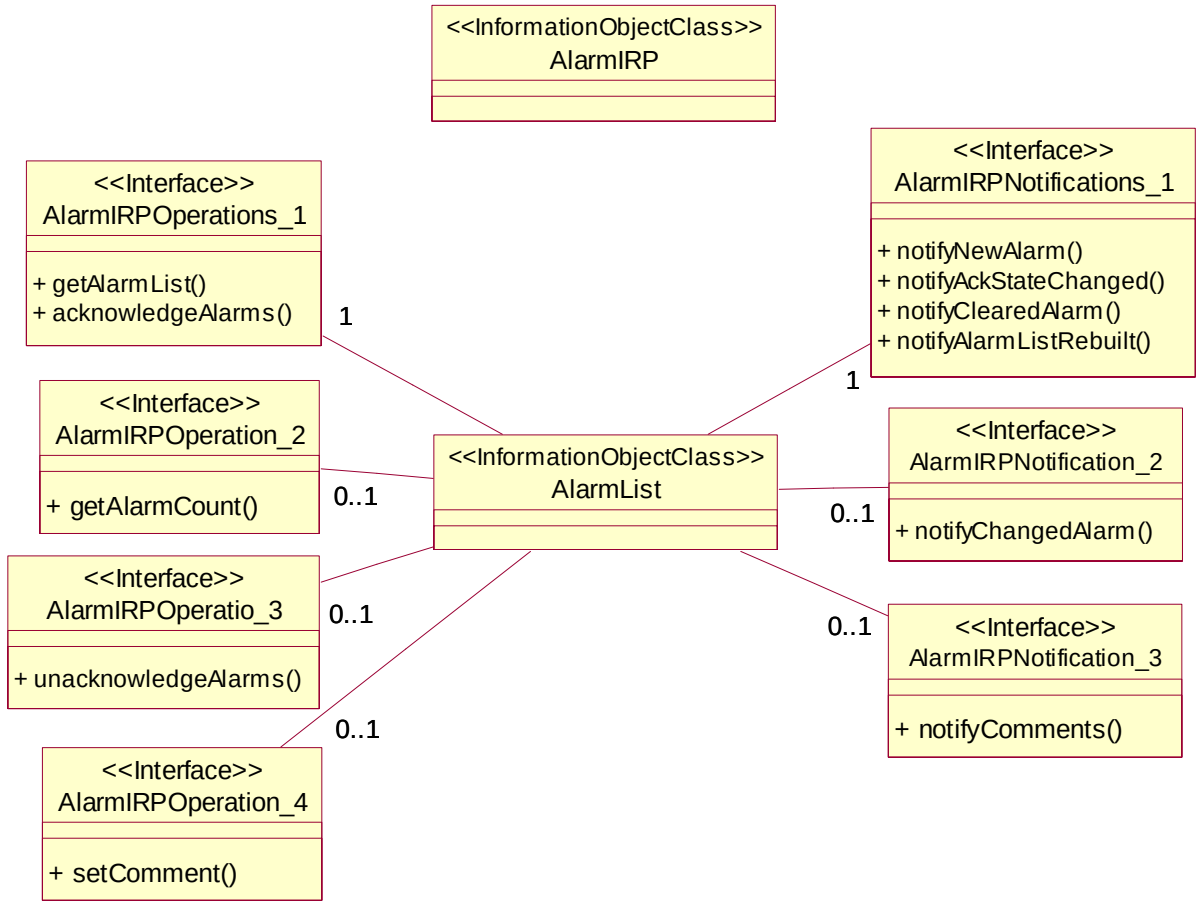
Name	Definition	Legal Values
ackUserId	It identifies the last user who has change the Acknowledgement State via operation acknowledgeAlarms or unacknowledgeAlarms. It can be used to identify the human operator such as “John Smith” or it can identify a group, such as “Team Six”, or it can contain no information such as “”.	N/A
ackSystemId	It identifies the system in which IRPManager, that invokes the acknowledgeAlarms or unacknowledgeAlarms operation, runs.	N/A
ackState	It identifies the Acknowledgement State of the alarm.	Acknowledged: the alarm has been acknowledged. Unacknowledged: the alarm has been unacknowledged or the alarm has never been acknowledged.
commentTime	It carries the time when a comment is made via setComment operation.	
commentText	It carries the textual comment made via setComment operation.	
commentUserId	It carries the identification of the user who made the comment via setComment operation.	
commentSystemId	It carries the identification of the system in which the the IRPManager runs. That IRPManager supports the user that made the comment.	
source	It identifies one MonitoredEntity.	All values that carry the semantics of DN.
notificationIdSet	It carries one or more notification identifiers.	

5.5.2 Constraints

Name	Definition
inv_alarmChangedTime	Time indicated shall be later than that carried in alarmRaisedTime.
inv_alarmClearedTime	Time indicated shall be later than that carried in alarmRaisedTime.
inv_ackTime	Time indicated shall be later than that carried in alarmRaisedTime.
inv_notificationId	NotificationIds shall be chosen to be unique across all notifications of a particular managed object (representing the NE) throughout the time that alarm correlation is significant. The algorithm by which alarm correlation is accomplished is outside the scope of this IRP.

6 Interface Definition

6.1 Class diagram



6.2 Generic rules

Rule 1: each operation with at least one input parameter supports a pre-condition valid_input_parameter which indicates that all input parameters shall be valid with regards to their information type. Additionally, each such operation supports an exception operation_failed_invalid_input_parameter which is raised when pre-condition valid_input_parameter is false. The exception has the same entry and exit state.

Rule 2: Each operation with at least one optional input parameter supports a set of pre-conditions supported_optional_input_parameter_xxx where "xxx" is the name of the optional input parameter and the pre-condition indicates that the operation supports the named optional input parameter. Additionally, each such operation supports an exception operation_failed_unsupported_optional_input_parameter_xxx which is raised when (a) the pre-condition supported_optional_input_parameter_xxx is false and (b) the named optional input parameter is carrying information. The exception has the same entry and exit state.

Rule 3: each operation shall support a generic exception operation_failed_internal_problem that is raised when an internal problem occurs and that the operation cannot be completed. The exception has the same entry and exit state.

6.3 Interface AlarmIRPOperations_1

6.3.1 acknowledgeAlarms (M)

6.3.1.1 Definition

The IRPManager invokes this operation to acknowledge one or more alarms.

6.3.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
alarm Information ReferenceList	M	List of AlarmInformation.alarm Id	It carries one or more identifiers identifying AlarmInformation instances in AlarmList.
AckUserId	M	AlarmInformation.ackUs erId	It identities the user acknowledging the alarm.
ackSystemId	O	AlarmInformation.ackSy stemId	It identifies the processing system on which the subject IRPManager runs. It may contain no information implying that IRPManager does not wish this information be kept in AlarmInformation in AlarmList.

6.3.1.3 Output Parameters

Name	Qualifier	Matching Information	Comment
badAlarm Information ReferenceList	M	List of pair of AlarmInformation.alarm Id and failure reason.	If allAlarmsAcknowledged is true, it contains no information. If someAlarmAcknowledged is true, then it contains identifications of AlarmInformation that are (a) present in input parameter AlarmInformationReferenceList but are absent in the AlarmList; or (b) present in input parameter AlarmInformationReferenceList and are present in the AlarmList but the Acknowledgement Information (see note below table) has not changed, in contrast to IRPManager's request.
status	M	ENUM (OperationSucceeded, OperationFailed, OperationPartiallySucceeded)	If someAlarmAcknowledged is true, status = OperationPartiallySuceeded. If allAlarmsAcknowledged is true, status = OperationSucceeded. If operation_failed is true, status = OperationFailed.

Note: Acknowledgement Information is defined as the information contained in AlarmInformation.ackTime, AlarmInformation.ackUserId, AlarmInformaton.ackSystemId, AlarmInformation.ackState.

6.3.1.4 Pre-condition

atLeastOneValidId.

Assertion Name	Definition
atLeastOneValidId	The AlarmInformationReferenceList contains at least one identifier that identifies one AlarmInformation in AlarmList and that this identified AlarmInformation shall have its ackState indicating "unacknowledged".

6.3.1.5 Post-condition

someAlarmAcknowledged OR allAlarmsAcknowledged.

Assertion Name	Definition
someAlarmAcknowledged	At least one but not all AlarmInformation identified in input parameter AlarmInformationReferenceList has been acknowledged. Acknowledgement of an AlarmInformation means that the ackState attribute has been set to "acknowledged", that ackUserId, ackSystemId attributes of this AlarmInformation have been set to the values provided as input parameter and that the time of acknowledgeAlarms operation has been registered in ackTime attribute.
allAlarmsAcknowledged	All AlarmInformation identified in input parameter have been acknowledged. Acknowledgement of an AlarmInformation means that the ackState attribute has been set to "acknowledged", that ackUserId, ackSystemId attributes of this AlarmInformation have been set to the values provided as input parameter and that the time of acknowledgeAlarms operation has been registered in ackTime attribute.

6.3.1.6 Exceptions

Name	Definition
operation_failed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.

6.3.2 getAlarmList (M)

6.3.2.1 Definition

IRPManager requests IRPAgent to provide the list of AlarmInformation instances in AlarmList.

There are two modes of operation. One mode is synchronous. In this mode, the list of AlarmInformation instances in AlarmList is returned synchronously with the operation. The other mode is asynchronous. In this mode, the list of AlarmInformation instances is returned via notifications. In asynchronous mode of operation, the only information returned synchronously is the status of the operation. The mode of operation to be used is determined by means outside the scope of specification. To use asynchronous mode, the IRPManager must have established a subscription with the IRPAgent notificationIRP via the subscribe operation specified in [5].

6.3.2.2 Input Parameters

Name	Qualifier	Information Type	Comment
alarmAckState	O	ENUM (all alarms, all active alarms, all active and acknowledged alarms, all active and unacknowledged, all Cleared and unacknowledged alarms, all unacknowledged)	It carries a constraint. The IRPAgent shall apply it on AlarmInformation instances in AlarmList when constructing its output parameter AlarmInformationList.
filter	O	N/A	It carries a filter constraint. The IRPAgent shall apply it

			on AlarmInformation instances in AlarmList when constructing its output parameter AlarmInformationList.

6.3.2.3 Output Parameters

Name	Qualifier	Matching Information	Comment
AlarmInformationList	M	List of AlarmInformation.	It carries AlarmInformation in AlarmList. Case when synchronous mode of operation is used: (a) The IRPAgent shall apply the constraints expressed in alarmAckState and filter to AlarmInformation instances when constructing this output parameter. Case when asynchronous mode of operation is used (i.e., this output parameter is conveyed via notifications): (a) If the filter parameter is present, the IRPAgent shall apply the constraint when constructing this output parameter. Furthermore, if the alarmAckState constraint is present, the IRPAgent shall apply that constraint as well. The filter constraint, if any, that is currently active in the notification channel is not used for the construction of this output parameter. (b) If the filter parameter is absent, the IRPAgent shall apply the filter constraint currently active in the notification channel when constructing this output parameter. If the alarmAckState constraint is present, the IRPAgent shall apply that constraint as well.
status	M	ENUM (OperationSucceeded, OperationFailed)	If allAlarmInformationReturned is true, status = OperationSucceeded. If operation_failed is true, status = OperationFailed.

6.3.2.4 Pre-condition

There is no pre-condition.

6.3.2.5 Post-condition

allAlarmInformationReturned.

Assertion Name	Definition
allAlarmInformationReturned	All AlarmInformation that satisfy the constraints expressed in input parameters filter and alarmAckState and are present in the AlarmList at the moment of this operation invocation are returned. All AlarmInformation in AlarmList remains unchanged as the result of this operation.

6.3.2.6 Exceptions

Assertion Name	Definition
operation_failed	**Condition:** At least one input parameter is invalid or the pre-condition is false or the post-condition is not ture. **Returned Information:** The output parameter status. **Exit state:** Entry state.

6.4 Interface AlarmIRPOperation_2

6.4.1 getAlarmCount (O)

6.4.1.1 Definition

An IRPManager wishes to know the amount of AlarmInformation kept in the AlarmList. The IRPManager requests the counts via this operation. Possible usage is for IRPManager to find out the number of AlarmInformation in AlarmList before invoking getAlarmList operation.

6.4.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
filter	O	N/A	It carries a filter constraint. The operation shall apply it when counting the AlarmInformation instances in AlarmList. Case when synchronous mode of operation is used for getAlarmList: (a) If this parameter is present, the operation shall count the AlarmInformation instances which satisfy both (a) this filter constraint and (b) the condition set by input parameter alarmAckState. (b) If this parameter is absent, the operation shall count all AlarmInformation instances that satisfy the condition set by input parameter alarmAckState. Case when asynchronous mode of operation is used for getAlarmList: (a) If this parameter is present, the operation shall count all AlarmInformation instances that satisfy this filter constraint and the condition set by input parameter alarmAckState. (b) If this parameter is absent, the operation shall count AlarmInformation instances that satisfy (a) the filter constraint currently active in the notification channel established between the IRPManager and the IRPAgent that is equipped with NotificationIRP capabilities and (b) the condition set by input parameter alarmAckState.
alarmAckState	O	ENUM (all alarms, all active alarms, all active and acknowledged alarms, all active and unacknowledged, all	It carries a constraint. The operation shall apply it on AlarmInformation instances in AlarmList when counting.

		cleared and unacknowledged alarms, all unacknowledged)	

6.4.1.3 Output Parameters

Name	Qualifier	Matching Information	Comment
criticalCount, majorCount, minorCount, warningCount, indeterminateCount, clearedCount	M	N/A	They carry the number of AlarmInformation in AlarmList that has the following properties. Case when synchronous mode of operation is used: (a) The operation shall apply the constraints expressed in alarmAckState and filter to AlarmInformation instances when counting. Case when asynchronous mode of operation is used (i.e., this output parameter is conveyed via notifications): (a) If the filter parameter is present, the operation shall apply the constraint when counting. Furthermore, if the alarmAckState constraint is present, the operation shall apply that constraint as well. The filter constraint, if any, that is currently active in the notification channel is not used for the counting. (b) If the filter parameter is absent, the operation shall apply the filter constraint currently active in the notification channel when counting. If the alarmAckState constraint is present, the operation shall apply that constraint as well.
status	M	ENUM (OperationSucceeded, OperationFailed)	If allAlarmInformationCounted is true, status = OperationSucceeded. If operation_failed is true, status = OperationFailed.

6.4.1.4 Pre-condition

There are no pre-conditions.

6.4.1.5 Post-condition

allAlarmInformationCounted.

Assertion Name	Definition
allAlarmInformationCounted	All AlarmInformation that satisfy the constraints expressed in input parameters filter and alarmAckState and are present in the AlarmList at the moment of this operation invocation are counted and the result returned. All AlarmInformation in AlarmList remains unchanged as the result of this operation.

6.4.1.6 Exceptions

Name	Definition
operation_failed	**Condition:** the pre-condition is false or the post-condition is true. **Returned Information:** The output parameter status. **Exit state:** Entry state.

6.5 Interface AlarmIRPOperation_3

6.5.1 unacknowledgeAlarms (O)

6.5.1.1 Definition

IRPManager invokes this operation to remove acknowledgement information kept in one or more AlarmInformation instances.

6.5.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
alarmInformationReferenceList	M	List of AlarmInformation.alarmId	It carries one or more identifiers identifying AlarmInformation in AlarmList.
ackUserId	M	AlarmInformation.ackUserId	It identities the user that invokes this operation.
ackSystemId	O	AlarmInformation.ackSystemId	It identifies the processing system on which the subject IRPManager runs.

6.5.1.3 Output Parameters

Name	Qualifier	Matching Information	Comment
badAlarmInformationReferenceList	M	List of pair of AlarmInformation.alarmId and the failure reason.	If allAlarmsUnacknowledged is true, it contains no information. If someAlarmUnacknowledged is true, then it contains identifications of AlarmInformation that are (a) present in input parameter AlarmInformationReferenceList but are absent in the AlarmList; or (b) present in input parameter AlarmInformationReferenceList and are present in the AlarmList but the Acknowledgement Information (see note below table) has not changed, in contrast to IRPManager's request.
status	M	ENUM (OperationSucceeded, OperationFailed, OperationPartiallySucceeded)	If someAlarmUnacknowledged is true, status = OperationPartiallySuceeded. If allAlarmsUnacknowledged is true, status = OperationSucceeded. If operation_failed is true, status = OperationFailed.

Note: Acknowledgement Information is defined as the information contained in `AlarmInformation.ackTime`, `AlarmInformation.ackUserId`, `AlarmInformaton.ackSystemId` and `AlarmInformation.ackState`.

6.5.1.4 Pre-condition

`atLeastOneValidId AND validUserId&SystemId.`

Assertion Name	Definition
`atLeastOneValidId`	The `AlarmInformationReferenceList` contains at least one identifier that identifies one `AlarmInformation` in `AlarmList` and that this identified `AlarmInformation` shall have its `ackState` indicating "acknowledged".
`validUserId&SystemId`	The values of `ackUserId` and `ackSystemId` attributes of the `AlarmInformation` must be the same as the ones provided as input parameters. The `AlarmInformation` is identified by the input parameter `AlarmInformationReferenceList`.

6.5.1.5 Post-condition

`someAlarmUnacknowledged` OR `allAlarmsUnacknowledged`.

Assertion Name	Definition
`someAlarmUnacknowledged`	At least one but not all `AlarmInformation` identified in input parameter `alarmListReferenceList` has been unacknowledged. This means that the ackState attribute has been set to "unacknowledged", that ackTime, ackUserId, ackSystemId attributes of this `AlarmInformation` have been set to containing no information.
`allAlarmsUnacknowledged`	All `AlarmInformation` identified in input parameter have been unacknowledged. This means that the ackState attribute has been set to "unacknowledged", that ackTime, ackUserId, ackSystemId attributes of this `AlarmInformation` have been set to contain no information.

6.5.1.6 Exceptions

Name	Definition
`operation_failed`	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.

6.6 Interface `AlarmIRPOperation_4`

6.6.1 `setComment` (O)

6.6.1.1 Definition

The `IRPManager` invokes this operation to record a comment in one or more `AlarmInformation` instances in AlarmList.

6.6.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
AlarmInformationReferenceList	M	List of AlarmInformation.alarmId	It carries one or more identifiers identifying AlarmInformation instances in the AlarmList.
commentUserId	M	The Comment.commentUserId where Comment is involved in relation-AlarmInformation-Comment with an AlarmInformation.	
commentSystemId	O	The Comment.commentSystemId where Comment is involved in relation-AlarmInformation-Comment with an AlarmInformation.	
commentText	M	The comment.commentText where Comment is involved in relation-AlarmInformation-Comment with an AlarmInformation.	

6.6.1.3 Output Parameter

Name	Qualifier	Matching Information	Comment
badAlarmInformationReferenceList	M	List of pair of AlarmInformation.alarmId and the failure reason.	If allUpdated is true, it contains no information. If someUpdated is true, then it contains identifications of AlarmInformation that are not present in AlarmList or that they are present, but AlarmInformation.comments has not changed, in contrast to IRPManager's request.
Status	M	ENUM(Operation succeeded, Operation failed, Operation partially failed)	If allUpdated is true, then status = OperationSsucceeded. If someUpdated is true, then status = OperationPartiallyFailed. If exception operationFailed is raised, then status = OperationFailed.

6.6.1.4 Pre-condition

atLeastOneValidId.

Assertion Name	Properties
atLeastOneValidId	The AlarmInformationReferenceList contains at least one identifier that identifies one AlarmInformation in AlarmList.

6.6.1.5 Post-condition

`allUpdated OR someUpdated`.

Assertion Name	Properties
`allUpdated`	The `AlarmInformation.comment` of all alarms identified by the input parameter `AlarmInformationReferenceList` has been updated. The input parameter `commentText`, `commentUserId` and `commentSystemId` are added to the `AlarmInformation.comment`. The time of the operation invocation is captured in the `AlarmInformation.comment` as well. To make it possible to add the new `comment`, the `IRPAgent` may remove one or more old `comment` previously held by `AlarmInformation.comments`.
`someUpdated`	The `AlarmInformation.`comment attribute of at least one but not all alarms identified by the input parameter `AlarmInformationReferenceList` has been updated. The input parameter `commentText`, `commentUserId` and `commentSystemId` are added to the `AlarmInformation.comment`. The time of the operation invocation is captured in the `AlarmInformation.comment` as well. To add a new `Comment`, it may be necessary to remove one or more old Comment instances being held. The `commentTime` of the removed `Comment` instances shall be older than that of the remaining `Comment` instances.

6.6.1.6 Exceptions

Name	Properties
`operation_failed`	**Condition:** the pre-condition is false or the post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.

6.7 Interface `AlarmIRPNotifications_1`

This specification does not specify methods for IRPManager to detect alarm loss. The use of `alarmId` to detect alarm loss is an arrangement made between `IRPAgent` and IRPManager. This arrangement is outside the scope of this specification. For example, `IRPAgent` may use integer sequence (e.g. 1, 2, 3, 4, 5, …) as `alarmId instances` for its alarms. Based on this knowledge, IRPManager can detect alarm loss. This kind of arrangement may not be possible for all SS.

This specification does not specify how `IRPAgent` can determine if IRPManager has received alarms correctly. Not all SSs provide such capability.

This document does not specify methods for `IRPManager` and `IRPAgent` to recover alarm loss. The only mechanism recommended to deal with alarm loss is the use of `getAlarmList` operation. This document does not specify conditions under which `IRPManager` should invoke this operation.

6.7.1 `notifyNewAlarm` (M)

6.7.1.1 Definition

A new `AlarmInformation` has been added in the `AlarmList`. The subscribed `IRPManager` instances are notified of this fact if the added `AlarmInformation` satisfies the current filter constraint of their subscription.

6.7.1.2 Input Parameters

Parameter Name	Qualifier	Matching Information	Comment
objectClass	M,F	MonitoredEntity.objectClass where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the new AlarmInformation.	
objectInstance	M,F	MonitoredEntity.objectInstance where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the new AlarmInformation.	
notificationId	M	This carries the semantics of notification identifier.	
eventTime	M,F	AlarmInformation.alarmRaisedTime	
systemDN	C,F	IRPAgent.systemDN where the IRPAgent is related to the AlarmIRP that is related to this AlarmList.	It carries the DN of the IRPAgent.
notificationType	M,F	"notifyNewAlarm".	
probableCause	M,F	AlarmInformation.probableCause	
perceivedSeverity	M,F	AlarmInformation.perceivedSeverity	
alarmType	M, F	AlarmInformation.eventType	
specificProblem	O	AlarmInformation.specificProblem	
correlatedNotifications	O	The set of CorrelatedNotification related to this AlarmInformation.	
backedUpStatus	O	AlarmInformation.backedUpStatus	
backUpObject	O	MonitoredEntity.objectInstance where the MonitoredEntity is identified by relation-BackUpObject-AlarmInformation of the new AlarmInformation.	It carries the DN of the back up object.
trendIndication	O	AlarmInformation.trendIndication	
thresholdInfo	O	AlarmInformation.thresholdInfo	

Parameter Name	Qualifier	Matching Information	Comment
		Info	
stateChangeDefinition	O	AlarmInformation.stateChange	
monitoredAttributes	O	AlarmInformation.monitoredAttributes	
proposedRepairActions	O	AlarmInformaton.proposedRepairActions	
additionalText	O	AlarmInformation.additionalText	
alarmId	M	AlarmInformation.alarmId	

6.7.1.3 Triggering Event

6.7.1.3.1 From-state

noMatchedAlarm.

Assertion Name	Definition
noMatchedAlarm	AlarmList does not contain an AlarmInformation that has the following properties: • Its matching-criteria-attributes values are identical to that of the newly generated network alarm and • It is involved in relation-AlarmObject-AlarmInformation with the same MonitoredEntity as the one identified by the newly generated network alarm.

6.7.1.3.2 To-state

newAlarmInAlarmList.

Assertion Name	Definition
newAlarmInAlarmList	AlarmList contains an AlarmInformation holding information conveyed by the newly generated network alarm. This AlarmInformation is involved in relation-AlarmObject-AlarmInformation with the same MonitoredEntity as the one identified by the newly generated network alarm. The following attributes of the AlarmInformation shall be populated with information in the newly generated alarm. alarmId, notificationId, alarmRaisedTime, eventType, probableCause, perceivedSeverity. The following attributes of the same AlarmInformation shall be populated with information in the newly generated alarm if the information is present (in the newly generated alarm) and if the attribute is supported. specificProblem, backedUpStatus, trendIndication, thresholdInfo, stateChangedDefinition, monitoredAttributes, proposedRepairActions, additionalText.

6.7.2 notifyAckStateChanged (M)

6.7.2.1 Definition

The subscribed IRPManager instances are notified regarding changes in alarm Acknowledgement State. The AlarmInformation carried in the notification shall satisfy the current filter constraint of the subscription.

The notification shall contain all parameters that are filterable and are present in the original (related) notifyNewAlarm notification.

6.7.2.2 Input Parameters

Parameter Name	Qualifier	Matching Information	Comment
objectClass	M,F	MonitoredEntity.objectClass where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the AlarmInformation.	
objectInstance	M,F	MonitoredEntity.objectInstance where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the AlarmInformation.	
notificationId	M	This carries the semantics of notification identifier.	
eventTime	M,F	AlarmInformation.ackTime	
systemDN	C,F	IRPAgent.systemDN	
notificationType	M,F	"notifyAckStateChanged"	
probableCause	M,F	AlarmInformation.probableCause	
perceived Severity	M,F	AlarmInformation.perceivedSeverity	
alarmType	M,F	AlarmInformation.eventType	
alarmId	M	AlarmInformation.alarmId	
ackTime	M	AlarmInformation.ackTime	
ackState	M	AlarmInformation.ackState	
ackUserId	M	AlarmInformation.ackUserId	
ackSystemId	O	AlarmInformation.ackSystemId	

6.7.2.3 Triggering Event

6.7.2.3.1 From-state

alarmInformationExists.

Assertion Name	Definition
alarmInformationExists	The AlarmInformation exists in AlarmList.

6.7.2.3.2 To-state

alarmAckStateHasChanged.

Assertion Name	Definition
alarmAckStateHasChanged	The AlarmInformation.ackState of the AlarmInformation identified by from-state assertion alarmInformationExists have been updated. Specifically, the following attributes of the subject AlarmInformation are updated. notificationId, ackTime, ackUserId, ackState, ackSystemId.

6.7.3 notifyClearedAlarm (M)

6.7.3.1 Definition

IRPAgent notifies The subscribed IRPManager is notified of alarm clearing if the subject AlarmInformation satisfies the optional filter constraint expressed in the subscribe operation.

The notification shall contain all parameters that are filterable and are present in the original (related) notifyNewAlarm notification.

6.7.3.2 Input Parameters

Parameter Name	Qualifier	Matching Information	Comment
objectClass	M,F	MonitoredEntity.objectClass where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the new AlarmInformation.	
objectInstance	M,F	MonitoredEntity.objectInstance where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the new AlarmInformation.	
notificationId	M	This carries the semantics of notification identifier.	
eventTime	M,F	AlarmInformation.alarmClearedTime	
systemDN	C,F	IRPAgent.systemDN where the IRPAgent is related to the AlarmIRP that is related to this AlarmList.	
notificationType	M,F	"notifyClearedAlarm"	
probableCause	M,F	AlarmInformation.probableCause	
perceivedSeverity	M,F	AlarmInformation.perceivedSeverity	Its value shall indicate Cleared.
alarmType	M,F	AlarmInformation.eventType	
correlated Notifications	O	The set of CorrelatedNotification related to this AlarmInformation.	It contains references to other AlarmInformation

Parameter Name	Qualifier	Matching Information	Comment
		related to this AlarmInformation.	instances whose perceivedSeverity levels are Cleared as well. In this way, perceivedSeverity level of multiple AlarmInformation instances can be Cleared by one notification.
alarmId	M	AlarmInformation.alarmId	

6.7.3.3 Triggering Event

6.7.3.3.1 From-state

alarmMatched AND alarmCleared.

Assertion Name	Definition
alarmMatched	The matching-criteria-attributes of the newly generated network alarm have values that are identical (matched) with ones in one AlarmInformation in AlarmList and the perceivedSeverity of the matched AlarmInformation is not Cleared.
alarmCleared	The perceivedSeverity of the newly generated network alarm is Cleared.

6.7.3.3.1 To-state

AlarmInformationCleared_1.

Assertion Name	Definition
AlarmInformationCleared_1	The following attributes of the subject AlarmInformation are updated. perceivedSeverity (updated to Cleared),alarmClearedTime.

6.7.4 notifyAlarmListRebuilt (M)

6.7.4.1 Definition

The IRPAgent or its related AlarmIRP maintains an AlarmList. They can lose confidence in the integrity of its AlarmList. Under this condition, IRPAgent or its related AlarmIRP or the related AlarmList shall invoke notifyAlarmListRebuilt notification after the AlarmList has been rebuilt.

The IRPAgent can also invoke notifyAlarmListRebuilt notification indicating that part of the AlarmList has been rebuilt. In this case, the notification carries the managed object (MO) instance indicating that the AlarmList only have been rebuilt for alarms concerning this MO and its subordinate MOs. Furthermore, this notification indicates that there is no rebuilt going on for superior MOs of this MO.

6.7.4.2 Input Parameters

Parameter Name	Qualifier	Matching Information	Comment
objectClass	M,F	It carries the IRPAgent.objectClass or alternatively, the object class of another MO.	If it carries the IRPAgent.objectClass, then all AlarmInformation instances in the AlarmList may have been rebuilt. If it carries the object class of another MO, then all AlarmInformation instances related to the MO and its subordinate MOs may have been rebuilt. The AlarmInformation instances not related to the subject MO and its subordinate MOs are not subject to rebuilt.
objectInstance	M,F	It carries the IRPAgent.iRPAgentId or alternatively, the id of another MO.	If objectClass carries the IRPAgent.objectClass, then this parameter carries the RDN of the IRPAgent whose AlarmList has been rebuilt. If objectClass carries the object class of another MO, then this parameter carries the RDN of the MO instance indicating that the AlarmList only have been rebuilt for alarms concerning that MO and its subordinate MOs.
notificationId	M	This carries the semantics of notification identifier.	
eventTime	M,F	It carries the time when the IRPAgent has rebuilt the AlarmList successfully.	
systemDN	C,F	IRPAgent.systemDN where the IRPAgent is related to the AlarmIRP that is related to this AlarmList.	
notificationType	M,F	"notifyAlarmListRebuilt".	
reason	M	"indeterminate". Other values can be added.	It carries the reason why the IRPAgent has rebuilt the AlarmList.

6.7.4.3 Triggering Event

6.7.4.3.1 From-state

alarmListRebuilt.

Assertion Name	Definition
alarmListRebuilt	IRPAgent loses confidence in part or whole of its AlarmList. IRPAgent has initiated procedure to repair its AlarmList.

6.7.4.3.2 To-state

alarmListRebuilt_2.

Assertion Name	Definition
alarmListRebuilt_2	IRPAgent rebuilt the whole or part of AlarmList.

6.8 Interface AlarmIRPNotification_2

6.8.1 notifyChangedAlarm (O)

6.8.1.1 Definition

The subscribed IRPManager instances are notified regarding changes in AlarmInformation in AlarmList. This notification is only triggered by a change in perceivedSeverity attribute value (except to the value "Cleared"). The AlarmInformation carried in the notification shall satisfy the current filter constraint of the subscription.

The notification shall contain all parameters that are filterable and are present in the original (related) notifyNewAlarm notification.

6.8.1.2 Input Parameters

Parameter Name	Qualifier	Matching Information	Comment
objectClass	M,F	MonitoredEntity.objectClass where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the new AlarmInformation.	
objectInstance	M,F	MonitoredEntity.objectInstance where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the new AlarmInformation.	
notificationId	M	This carries the semantics of notification identifier.	
eventTime	M,F	AlarmInformation.alarmChangedTime	
systemDN	C,F	IRPAgent.systemDN where the IRPAgent is related to the AlarmIRP that is related to this AlarmList.	
notificationType	M,F	"notifyChangedAlarm"	
probableCause	M,F	AlarmInformation.probableCause	
perceivedSeverity	M,F	AlarmInformation.perceivedSeverity	
alarmType	M,F	AlarmInformation.eventType	
alarmId	M	AlarmInformation.alarmId	

6.8.1.3 Triggering Event

6.8.1.3.1 From-state

alarmMatched AND alarmNotCleared AND alarmChanged.

Assertion Name	Definition
`alarmMatched`	The matching-criteria-attributes of the newly generated network alarm has values that are identical (matches) with ones in one `AlarmInformation` in AlarmList.
`alarmNotCleared`	The `perceivedSeverity` of the newly generated network alarm is not `Cleared`.
`alarmChanged`	The `perceivedSeverity` of the newly generated network alarm and of the matched `AlarmInformation` are different.

6.8.1.3.2 To-state

`informationUpdate.`

Assertion Name	Definition
`informationUpdate`	• The `AlarmInformation` identified in `alarmMatched` in from-state has been updated according to the following rules : `perceivedSeverity` is updated; • `notificationId` is updated; • `alarmChangedTime` is updated; • ackTime, ackUserId and ackSystemId are updated to contain no information; • ackState is updated to “unacknowledged”;

6.9 Interface `AlarmIRPNotification_3`

6.9.1 `notifyComments` (O)

6.9.1.1 Definition

The subscribed `IRPManager` instances are notified regarding to the addition of `Comment,` as a consequence of successful completion of `setComment` operation, in `AlarmInformation` instances in `AlarmList`. The `AlarmInformation` carried in the notification shall satisfy the current filter constraint of the subscription.

The notification shall contain all parameters that are filterable and are present in the original (related) `notifyNewAlarm` notification.

IRPAgent shall support this notification if it supports the operation `setComment`.

6.9.1.2 Input Parameters

Parameter Name	Qualifier	Matching Information	Comment
objectClass	M,F	MonitoredEntity.objectClass where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the AlarmInformation.	
objectInstance	M,F	MonitoredEntity.objectInstance where the MonitoredEntity is identified by the relation-AlarmedObject-AlarmInformation of the AlarmInformation.	
notificationId	M	This carries the semantics of notification identifier.	
eventTime	M,F	AlarmInformation.alarmChangedTime	
systemDN	C,F	IRPAgent.systemDN	
notificationType	M,F	"notifyComments"	
alarmType	M,F	AlarmInformation.eventType	
probableCause	M,F	AlarmInformation.probableCause	
perceived Severity	M,F	AlarmInformation.perceivedSeverity	
comments	M	The set of Comment instances involved in relationship with this AlarmInformation.	
alarmId	M	AlarmInformation.alarmId	

6.9.1.3 Triggering Events

6.9.1.3.1 From-state

alarmInformationExists.

Assertion Name	Definition
alarmInformationExists	The AlarmInformation is in AlarmList.

6.9.1.3.2 To-state

commentInserted.

Assertion Name	Definition
commentInserted	One Comment has been created and it is involved in a relationship with the AlarmInformation identified by from-state assertion alarmInformationExists. The following attributes of the newly created Comment shall be populated. commentTime (set to setComment operation completion time), commentText, commentUserId and commentSystemId.

Annex A (normative): Event Types

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

Event type is defined in 3GPP TS 32.301-2 [5]. The table below lists some of the event types referred to in this document.

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

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

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

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

TableA.1: Event Types

Event Types	Explanation
Communications Alarm	An alarm of this type is associated with the procedure and/or process required conveying information from one point to another (ITU-T Recommendation X.733 [2]).
Processing Error Alarm	An alarm of this type is associated with a software or processing fault (ITU-T Recommendation X.733 [2]).
Environmental Alarm	An alarm of this type is associated with a condition related to an enclosure in which the equipment resides (ITU-T Recommendation X.733 [2]).
Quality of Service Alarm	An alarm of this type is associated with degradation in the quality of a service (ITU-T Recommendation X.733 [2]).
Equipment Alarm	An alarm of this type is associated with an equipment fault (ITU-T Recommendation X.733 [2]).

Annex B (normative): Probable Causes

This appendix lists probable causes and their corresponding event types.

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

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

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

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

M.3100 Probable cause	Event type
Rectifier High Voltage	Environmental
Rectifier Low F Voltage	Environmental
Ventilation System Failure	Environmental
Enclosure Door Open	Environmental
Explosive Gas	Environmental
Fire	Environmental
Flood	Environmental
High Humidity	Environmental
High Temperature	Environmental
High Wind	Environmental
Ice Build Up	Environmental
Intrusion Detection	Environmental
Low Fuel	Environmental
Low Humidity	Environmental
Low Cable Pressure	Environmental
Low Temperature	Environmental
Low Water	Environmental
Smoke	Environmental
Toxic Gas	Environmental
Storage Capacity Problem	Processing error
Memory Mismatch	Processing error
Corrupt Data	Processing error
Out Of CPU Cycles	Processing error
Software Environment Problem	Processing error
Software Download Failure	Processing error

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

X.733 Probable Cause	Event type
Adapter Error	Equipment
Application Subsystem Failure	Processing error
Bandwidth Reduction	Quality of service
Call Establishment Error	Communications
Communication Protocol Error	Communications
Communication Subsystem Failure	Communications
Configuration or Customizing Error	Processing error
Congestion	Quality of service
Corrupt Data	Processing error
CPU Cycles Limit Exceeded	Processing error
Data Set or Modem Error	Equipment
Degraded Signal	Communications
DTE-DCE Interface Error	Communications
Enclosure Door Open	Environmental
Equipment Malfunction	Equipment
Excessive Vibration	Environmental
File Error	Processing error
Fire Detected	Environmental
Flood Detected	Environmental
Framing Error	Communications
Heating or Ventilation or Cooling System Problem	Environmental
Humidity Unacceptable	Environmental
Input/Output Device Error	Equipment
Input Device Error	Equipment
LAN Error	Communications
Leak Detection	Environmental
Local Node Transmission Error	Communications
Loss of Frame	Communications
Loss of Signal	Communications
Material Supply Exhausted	Environmental
Multiplexer Problem	Equipment
Out of Memory	Processing error
Output Device Error	Equipment
Performance Degraded	Quality of service
Power Problem	Equipment

X.733 Probable Cause	Event type
Pressure Unacceptable	Environmental
Processor Problem	Equipment
Pump Failure	Environmental
Queue Size Exceeded	Quality of service
Receive Failure	Equipment
Receiver Failure	Equipment
Remote Node Transmission Error	Communications
Resource at or Nearing Capacity	Quality of service
Response Time Excessive	Quality of service
Re-transmission Rate Excessive	Quality of service
Software Error	Processing error
Software Program Abnormally Terminated	Processing error
Software Program Error	Processing error
Storage Capacity Problem	Processing error
Temperature Unacceptable	Environmental
Threshold Crossed	Quality of service
Timing Problem	Equipment
Toxic Leak Detected	Environmental
Transmit Failure	Equipment
Transmitter Failure	Equipment
Underlying Resource Unavailable	Processing error
Version Mismatch	Processing error

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

GSM 12.11 Probable Cause	Event Type
A-bis to BTS interface failure	Equipment
A-bis to TRX interface failure	Equipment
Antenna problem	Equipment
Battery breakdown	Equipment
Battery charging fault	Equipment
Clock synchronisation problem	Equipment
Combiner problem	Equipment
Disk problem	Equipment
Equipment failure	Equipment
Excessive receiver temperature	Equipment
Excessive transmitter output power	Equipment
Excessive transmitter temperature	Equipment
Frequency hopping degraded	Equipment
Frequency hopping failure	Equipment
Frequency redefinition failed	Equipment
Line interface failure	Equipment
Link failure	Equipment
Loss of synchronisation	Equipment
Lost redundancy	Equipment
Mains breakdown with battery back-up	Equipment
Mains breakdown without battery back-up	Equipment
Power supply failure	Equipment
Receiver antenna fault	Equipment
Receiver Failure	Equipment
Receiver multicoupler failure	Equipment
Reduced transmitter output power	Equipment
Signal quality evaluation fault	Equipment
Timeslot hardware failure	Equipment
Transceiver problem	Equipment
Transcoder problem	Equipment
Transcoder or rate adapter problem	Equipment
Transmitter antenna failure	Equipment
Transmitter antenna not adjusted	Equipment
Transmitter failure	Equipment
Transmitter low voltage or current	Equipment
Transmitter off frequency	Equipment
Database inconsistency	Processing error

GSM 12.11 Probable Cause	Event Type
File system call unsuccessful	Processing error
Input parameter out of range	Processing error
Invalid parameter	Processing error
Invalid pointer	Processing error
Message not expected	Processing error
Message not initialised	Processing error
Message out of sequence	Processing error
System call unsuccessful	Processing error
Timeout expired	Processing error
Variable out of range	Processing error
Watch dog timer expired	Processing error
Cooling system failure	Environmental
External equipment failure	Environmental
External power supply failure	Environmental
External transmission device failure	Environmental
Fan failure	Environmental
High humidity	Environmental
High temperature	Environmental
Intrusion detected	Environmental
Low humidity	Environmental
Low temperature	Environmental
Smoke detected	Environmental
Excessive Error Rate	Quality of service
Reduced alarm reporting	Quality of service
Reduced event reporting	Quality of service
Reduced logging capability	Quality of service
System resources overload	Quality of service
Broadcast channel failure	Communications
Connection establishment error	Communications
Invalid message received	Communications
Invalid MSU received	Communications
LAPD link protocol failure	Communications
Local alarm indication	Communications
Remote alarm indication	Communications
Routing failure	Communications
SS7 protocol failure	Communications
Transmission error	Communications

Table 20 identifies probable causes that are defined by more than one standard. This is for information only.

Table B.4: Duplicated Probable Causes

Duplicated Probable Cause	GSM 12.11	X.721 X.733	M.3100	Event Type
Call Establishment Failure (X.721/X.733) Call Setup Failure (M.3100)		X	X	Communications
Degraded Signal		X	X	Communications
Framing Error		X	X	Communications
Loss of Frame		X	X	Communications
Loss of Signal		X	X	Communications
Equipment Failure (GSM 12.11) Equipment Malfunction (X.721/X.733)	X	X		Equipment
Multiplexer Problem		X	X	Equipment
Power Problem		X	X	Equipment
Processor Problem		X	X	Equipment
Receiver Failure	X	X	X	Equipment
Timing Problem		X	X	Equipment
Transmitter Failure	X	X	X	Equipment
Enclosure Door Open		X	X	Environmental
Fan Failure (GSM 12.11) Cooling Fan Failure (M.3100)	X		X	Environmental
Fire Detected (X.721/X.733) Fire (M.3100)		X	X	Environmental
Flood Detected (X.721/X.733) Flood (M.3100)		X	X	Environmental
High Humidity	X		X	Environmental
High Temperature	X		X	Environmental
Intrusion Detected (GSM 12.11) Intrusion Detection (X.736/M.3100)	X		X	Environmental
Low Humidity	X		X	Environmental
Low Temperature	X		X	Environmental
Pump Failure		X	X	Environmental
Smoke Detected (GSM 12.11) Smoke (M.3100)	X		X	Environmental
Storage Capacity Problem		X	X	Processing Error
Excessive Bit Error Rate (M.3100) Excessive Error Rate (GSM12.11)	X		X	
Corrupt Data		X	X	Processing Error

Annex C (informative): Examples of using notifyChangedAlarm

This annex describes a number of valid and invalid interactions governing the case when IRPAgent is reporting a specific fault of a particular network resource whose alarm severity level changes from, e.g. “Critical” to “Minor” and then to “Cleared”.

In the following examples:

ni is notificationId,

moc is managedObjectClass,

moi is managedObjectInstance,

et is eventType,

pc is probableCause,

sp is specificProblem,

ps is perceivedSeverity and

ai is alarmId.

EXAMPLE 1: Valid sequence 1 to support the hypothetical case:

(1) NotifyNewAlarm

(ni=1, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Critical)

(2) NotifyChangedAlarm

(ni=2, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Minor)

(3) NotifyClearedAlarm

(ni=3, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Cleared)

EXAMPLE 2: Valid sequence 2 to support the hypothetical case:

(1) NotifyNewAlarm

(ni=1, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Critical)

(2) NotifyClearedAlarm

(ni=2, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Cleared)

(3) NotifyNewAlarm

(ni=3, ai=Y, moc=A, moi=B, et=C, pc=D, sp=E, ps=Minor)

(4) NotifyClearedAlarm

(ni=4, ai=Y, moc=A, moi=B, et=C, pc=D, sp=E, ps=Cleared)

EXAMPLE 3: Invalid sequence 1 to support the hypothetical case:

```
(1) NotifyNewAlarm

    (ni=1, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Critical)

(2) NotifyChangedAlarm

    (ni=2, ai=Y, moc=A, moi=B, et=C, pc=D, sp=E, ps=Minor)

(3) NotifyClearedAlarm

    (ni=3, ai=Y, moc=A, moi=B, et=C, pc=D, sp=E, ps=Cleared)
```

Interaction (2) is illegal since it uses a different `ai` for the same alarm. It should use `ai=X` as in interaction (1).

EXAMPLE 4: Invalid sequence 2 to support the hypothetical case:

```
(1)  NotifyNewAlarm

   (ni=1, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Critical)

(2)  NotifyNewAlarm

   (ni=2, ai=X, moc=A, moi=B, et=C, pc=D, sp=E, ps=Minor)
```

Interaction (2) is illegal since it invokes notifyNewAlarm using same `ai` value. It should use `notifyChangedAlarm` with the same `ai` value.

Annex D (informative): Change history

Change history					
TSG SA#	**Version**	**CR**	**Tdoc SA**	**New Version**	**Subject/Comment**
S_07	2.0.0	-	SP-000012	3.0.0	Approved at TSG SA #7 and placed under Change Control
Mar 2000	3.0.0			3.0.1	Cosmetic
S_08	3.0.1	004	SP-000250	3.1.0	Split of TS - Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)
Sep 2000	3.1.0			3.1.1	Cosmetic
S_09	3.1.1	001	SP-000438	3.2.0	Correction of qualifier for `SystemDN`
S_09	3.1.1	002	SP-000438	3.2.0	Addition of a missing constraint in acknowledgeAlarm operation
S_10	3.2.0	003	SP-000520	3.3.0	Incorrect modifiable attributes
S_10	3.2.0	004	SP-000520	3.3.0	Add acknowledgement information to getAlarmList result
S_10	3.2.0	005	SP-000520	3.3.0	Identification of valid Event Types and Extended Event Types within Notifications
S_10	3.2.0	006	SP-000520	3.3.0	A cleared Alarm shall be given perceived severity "Cleared" and nothing else
S_10	3.2.0	007	SP-000520	3.3.0	Inconsistent behaviour for cleared not yet acknowledged alarms
S_12	3.3.1	008		4.0.0	Alarm IRP: IS Rel4 - Addition of feature

3GPP TSG-SA5 (Telecom Management) **S5-010345**
Meeting #20, Brighton, UK, 28 May - 1 June 2001 **S5F010117**

CR-Form-v3

CHANGE REQUEST

⌘ **32.111-3** **CR** **009** ⌘ rev **-** ⌘ Current version: **3.5.0** ⌘

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

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

Title:	⌘	Alarm IRP: CORBA SS Rel4 - Addition of feature
Source:	⌘	SA5
Work item code:	⌘	OAM-FM
Date:	⌘	01/062001
Category:	⌘	**B**
Release:	⌘	Rel4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change: ⌘ Reflect changes to Alarm IRP: IS Rel4 (32.111-2).

Summary of change: ⌘

Add optional `setComment`() and `notifyComment`().

Remove the use of Extended Event Type. Use notificationType and alarmType for notification parameters. The notificationType is mapped to CORBA structured event header second field. The alarmType is mapped to CORBA structured event header third field.

Replace the use of "1:1" and "1f1" by the "version number" based on the 3GPP issued specification document number.

Use IDL class to encapsulate const string to avoid the generation of large number of classes.

Support Partial Alarm List Rebuilt

Add optional operations `getAlarmIRPOperationProfile`() and `getAlarmIRPNotificationProfile`()

Consequences if not approved: ⌘ There will be no CORBA SS that corresponds to the Alarm IRP: IS Rel4.

Clauses affected: ⌘ All clauses.

Other specs affected: ⌘
☐ Other core specifications ⌘
☐ Test specifications
☐ O&M Specifications

Other comments: ⌘ This CR is dependent on the approval of CR32.111-3-008_S5-010391 & CR32.111-1-003_S5-010343 and CR32.111-2-008_S5-010344

3GPP TS 32.111-3 V3.5.1 (2001-06)

Technical Specification

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

Keywords
Fault Management, Alarms

3GPP

Postal address

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

Internet
http://www.3gpp.org

Contents

Foreword

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

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

Part 1: "3G Fault Management Requirements";

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

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

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

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

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

1 Scope

The present document specifies the CORBA Solution Set (SS) for the IRP whose semantics is specified in Alarm IRP: Information Service (IS) (3G TS 32.111-2 [6]).

Clause 1 to 3 provides background information. Clause 4 provides key architectural features supporting the SS. Clause 5 defines the mapping of operations, notification, parameters and attributes defined in IS to their SS equivalents. Clause 6 describes the notification interface containing the push method. Annex A contains the IDL specification.

2 References

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

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

[1] OMG TC Document telecom/98-11-01: “OMG Notification Service”.

[2] OMG CORBA Services: “Common Object Services Specification, Update: November 22, 1996” (Clause 4 contains the Event Service specification).

[3] 3G TS 32.106-8: “Name Convention for Managed Objects”.

[4] 3G TS 32.106-2: “Notification IRP: Information Service”.

[5] 3G TS 32.106-3: “Notification IRP: CORBA Solution Set”.

[6] 3G TS 32.111-2: “Alarm Integration Reference Point: Information Service”.

3 Definitions and abbreviations

3.1 Definitions

In addition to the terms and definitions defined in TS 32.111-2 [6], there are no additional definitions applicable to the present document.

3.2 Abbreviations

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

CORBA	Common Object Request Broker Architecture
IDL	Interface Definition Language
IRP	Integration Reference Point
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
OMG	Object Management Group
TMN	Telecommunications Management Network
UML	Unified Model Language

3.3 IRP document version number string

The IRP document version number (sometimes called “IRP version” or “version number”) string is used to identify this specification. The string is derived using the following rule.

Take the 3GPP document number on the front page of this specification, such as “3GPP TS 32.106-3 V3.2.0 (2000-12)”. Discard the leading “3GPP TS ”. Discard all characters after and including the last period. Eliminate leading and trailing spaces. Reduce multiple consecutive spaces with one space. Express the resultant in a string. Capitalised the string. For example, if the 3GPP document version number is “3GPP TS 32.106-3 V3.2.0 (2000-12)”, then the IRP document version number shall be “32.106 V3.2”.

This string is returned in `getAlarmIRPVersion` method and is carried in the first field of the notification header of all notifications related to alarm IRP.

4 Architectural Features

The overall architectural feature of Alarm IRP is specified in 3G TS 32.111-2 [6]. This clause specifies features that are specific to the CORBA SS.

4.1 Notification Services

In implementations of CORBA SS, IRPAgent conveys Alarm Information to IRPManager via OMG Notification Service (OMG Notification Service [1]).

OMG Event Service [2] provides event routing and distribution capabilities. OMG Notification Service provides, in addition to Event Service, event filtering and Quality Of Service (QOS) as well.

A necessary and sufficient sub set of OMG Notification Services shall be used to support `AlarmIRPNotifications` notifications as specified in 3G TS 32.111-2 [6].

4.2 Push and Pull Style

OMG Notification Service defines two styles of interaction. One is called push style. In this style, IRPAgent pushes notifications to IRPManager as soon as they are available. The other is called pull style. In this style, IRPAgent keeps the notifications till IRPManager requests for them.

This CORBA SS specifies that support of Push style is Mandatory (M) and that support of Pull style is Optional (O).

4.3 Support multiple notifications in one push operation

For efficiency reasons, IRPAgent may send multiple notifications using one single `push` operation. To pack multiple notifications into one `push` operation, IRPAgent may wait and not invoke the `push` operation as soon as notifications are available. To avoid IRPAgent to wait for an extended period of time that is objectionable to IRPManager, IRPAgent shall implement an IRPAgent wide timer configurable by administrator. On expiration of this timer, IRPAgent shall invoke `push` if there is at least one notification to be conveyed to IRPManager. This timer is re-started after each `push` invocation.

4.4 Filter

IRPAgent shall optionally support alarm filtering based on IRPManager's supplied alarm filter constraints (e.g., as parameter in `subscribe`() of 3G TS 32.106-2 [4]. Alarm filtering can be applied in the following cases:

- It is applicable to alarms emitted by IRPAgent via `AlarmIRPNotifications`. IRPManager supplies alarm filter constraint via the `subscribe` method. This filter is effective during the period of subscription.
- It is applicable to alarms returned by IRPAgent via the `out` parameter of `get_alarm_list` method. IRPManager supplies alarm filter constraint via the `get_alarm_list` method. This filter is effective only for this method invocation.
- It is applicable to the calculation of alarm counts returned by IRPAgent via the `out` parameters of `get_alarm_count` method. IRPManager supplies alarm filter constraint via the `get_alarm_count` method. This filter is effective only for this method invocation.

This SS shall use of filter constraint grammar specified by reference OMG Notification Service [1]. The name of the grammar is called "`EXTENDED_TCL`". See clause 2.4, Default Filter Constraint Language in OMG Notification Service [1]. This SS shall use this grammar only.

5 Mapping

5.1 Operation and Notification mapping

Alarm IRP: IS 3G TS 32.111-2 [6] defines semantics of operation and notification visible across the Alarm IRP. Table 1 indicates mapping of these operations and notifications to their equivalents defined in this SS.

Table 1: Mapping from IS Notification/Operation to SS equivalents

IS Operation/ notification 3G TS 32.111-2 [13]	SS Method	Qualifier
`acknowledgeAlarms`	`acknowledge_alarms`	M
`unacknowledgeAlarms`	`unacknowledge_alarms`	O
`getAlarmList`	`get_alarm_list`	M
`getAlarmIRPVersion`	`get_alarm_IRP_version`	M
`getAlarmCount`	`get_alarm_count`	O
`setComment`	`set_comment`	O
`notifyNewAlarm`	`push_structured_event` Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 6.1	M
`notifyClearedAlarm`	`push_structured_event`See clause 6.1	M
`notifyChangedAlarm`	`push_structured_event`See clause 6.1	M
`notifyAckStateChanged`	`push_structured_event`See clause 6.1	M
`notifyAlarmListRebuilt`	`push_structured_event` `See clause 6.1`	M
`notifyComments`	`push_structured_event`See clause 6.1	O

5.2 Operation parameter mapping

Reference 3G TS 32.111-2 [6] defines semantics of parameters carried in operations across the Alarm IRP. The following set of tables indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

Table 2: Mapping from IS `acknowledgeAlarms` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarmInformation ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list`	M
`ackUserId`	`string ack_user_id`	M
`ackSystemId`	`string ack_system_id`	O
`bad AlarmInformation ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq bad_alarm_information_id_list`	M

`status`	`CommonIRPConstDefs::Signal`
	Exceptions:
	`AcknowledgeAlarms, ParameterNotSupported, InvalidParameter`
		M

Table 3: Mapping from IS `unacknowledgeAlarms` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarm InformationReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list`	M
`ackUserId`	`string ack_user_id`	M
`ackSystemId`	`string ack_system_id`	O
`badAlarm Information ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq bad_alarm_information_id_list`	M

`status`	`CommonIRPConstDefs::Signal`
	Exceptions:
	`UnacknowledgeAlarms, OperationNotSupported, ParameterNotSupported, InvalidParameter`
		M

Table 4: Mapping from IS `getAlarmList` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarmAckState, filter`	`string filter`	O
`alarmInformation List`	Return value of type `AlarmIRPConstDefs::AlarmInformationSeq`	M
`status`	Exceptions: `GetAlarmList, ParameterNotSupported, InvalidParameter`	M

Table 5: Mapping from IS `getAlarmCount` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarmAckState, filter`	`string filter`	O
`criticalCount, majorCount, minorCount, warningCount, indeterminateCount,clearedCount`	`long critical_count, long major_count, long minor_count, long warning_count, long indeterminate_count, long cleared_count`	M
`status`	Exceptions: `GetAlarmCount, OperationNotSupported, ParameterNotSupported, InvalidParameter`	M

Table 6: Mapping from IS `getAlarmIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`versionNumberList`	Return value of type `CommonIRPConstDefs::VersionNumberSet`	M
`status`	Exceptions: `GetAlarmIRPVersion`	M

Table 7: Mapping from IS `setComment` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`AlarmInformation ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list`	M
`commentUserId`	`string comment_user_id`	M
`commentSystemId`	`string comment_system_id`	M
`commentText`	`string comment_text`	M
`badAlarmInformationIdList`	`AlarmIRPConstDefs::BadAlarmInformationIdSe bad_alarm_information_id_list`	
`status`	Exceptions: `CommentAlarms,OperationNotSupported.`	

5.3 Notification parameter mapping

Reference 3G TS 32.111-2 [6] defines semantics of parameters carried in notifications. The following tables indicate the mapping of these parameters to their OMG CORBA Structured Event (defined in OMG Notification Service [1]) equivalents. The composition of OMG Structured Event, as defined in the OMG Notification Service [1], is:

```
Header
      Fixed Header
           domain_name
           type_name
           event_name
      Variable Header
Body
      filterable_body_fields
      remaining_body
```

The following tables list all OMG Structured Event attributes in the second column. The first column identifies the Alarm IRP: IS [6] defined notification parameters.

Table 8: Mapping for `notifyNewAlarm`

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding SS attribute.	`domain_name`		It carries the IRP document version number string. See sub-clause 3.3. It indicates the syntax and semantics of the Structured Event as defined by this specification.
`notification Type`	`type_name`	`M`	This is the `NOTIFY_FM_NEW_ALARM` of `interface NotificationType` of module `NotificationIRPConstDefs`.
`alarmType`	`event_name`	`M`	It identifies one of the following: • communications alarm,

			• processing error alarm, • environmental alarm, • quality of service alarm and • equipment alarm. It is a string. See block of const string definitions encapsulated by `interface AlarmTypes` in the IDL. The strings start with "ET_".
There is no corresponding SS attribute.	`variable Header`		
`managedObjectClass, managedObjectInstance`	One NV pair of `filterable_body_fields`	M	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. Name of NV pair is the `MANAGED_OBJECT_INSTANCE` of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS (3G TS 32.106-3 [5]).
`notificationId`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the NOTIFICATION_ID of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a long. See corresponding table in Notification IRP: CORBA SS (3G TS 32.106-3 [5]).
`eventTime`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the EVENT_TIME of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a `IRPTime`. See corresponding table in Notification IRP: CORBA SS (3G TS 32.106-3 [5]).
`systemDN`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the SYSTEM_DN of `interface AttributeNameValue` of `module` NotificationIRPConstDefs. Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS [5].
`probableCause`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the `PROBABLE_CAUSE` of `interface AttributeNameValue of module AlarmIRPConstDefs`. Value of NV pair is a short defined by `interface ProbableCause`.
`perceivedSeverity`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the PERCEIVED_SEVERITY of `interface AttributeNameValue of module AlarmIRPConstDefs`. Value of NV pair is a short defined by `interface PerceivedSeverity`.
`specificProblem`	One NV pair of `filterable_body_fields`	O	Name of NV pair is the SPECIFIC_PROBLEM of `interface AttributeNameValue` of `module AlarmIRPConstDefs.` Value of NV pair is a string.
`correlatedNotifications`	One NV pair of `filterable_body_fields`	O	Name of NV pair is the CORRELATED_NOTIFICATIONS of `interface AttributeNameValue`. Value of NV pair is a `CorrelatedNotificationSetType.`
`backedUpStatus`	One NV pair of `filterable_body_fields`	O	Name of NV pair is the BACKED_UP_STATUS of `interface AttributeNameValue` of `module AlarmIRPConstDefs` Value of NV pair is a `boolean BackedUpStatusType`.

backUpObject	One NV pair of filterable_body_fields	O	Name of NV pair is the BACKED_UP_OBJECT of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string carrying of DN of the back-up object. See 3G TS 32.106-8 [3] for the DN string representation.
trendIndication	One NV pair of filterable_body_fields	O	Name of NV pair is the TREND_INDICATION of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is an enum TrendIndicationType.
thresholdInfo	One NV pair of filterable_body_fields	O	Name of NV pair is the THRESHOLD_INFO of interface ParameterNameValue of module AlarmIRPConstDefs. Value of NV pair is an enum ThresholdIndicationType.
stateChangeDefinition	One NV pair of filterable_body_fields	O	Name of NV pair is the STATE_CHANGE_DEFINITION of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is an AttributeChangeSetType.
monitoredAttributes	One NV pair of filterable_body_fields	O	Name of NV pair is the MONITORED_ATTRIBUTES of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is an AttributeSetType.
proposedRepairActions	One NV pair of filterable_body_fields	O	Name of NV pair is the PROPOSED_REPAIR_ACTIONS of inteface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
additionalText	One NV pair of filterable_body_fields	O	Name of NV pair is the ADDITIONAL_TEXT of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
alarmId	One NV pair of filterable_body_fields	M	Name of NV pair is the ALARM_ID of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string. If the string is a zero-length string or if this NV pair is absent, the default semantics is that alarmId is a concatenation ofmanagedObjectInstance, eventType, probableCause and specificProblem, if present, of this Structured Event. Since probableCuase is encoded as a short, it shall be converted into string before concatenation. The resultant string shall not contain spaces.
There is no corresponding IS attribute.	remaining_body		

Table 9: Mapping for notifyAckStateChanged

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_ACK_STATE_CHANGED of interface NotificationType of module NotificationIRPConstDefs.
alarmType	event_name	M	See that of notifyNewAlarm.

There is no corresponding IS attribute.	variable Header		
managedObject Class, managedObjectInstance	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
notificationId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
perceivedSeverity	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
ackTime	One NV pair of filterable_body_fields	M	Name of NV pair is the ACK_TIME of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a IRPTime of module ManagedGenericIRPConstDefs.
ackUserId	One NV pair of filterable_body_fields	M	Name of NV pair is the ACK_USER_ID of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
ackSystemId	One NV pair of filterable_body_fields	O	Name of NV pair is the ACK_SYSTEM_ID of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
ackState	One NV pair of filterable_body_fields	M	Name of NV pair is the ACK_STATE of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a short defined by interface AckState of module AlarmIRPConstDefs.
There is no corresponding IS attribute.	remaining_body		

Table 10: Mapping for notifyClearedAlarm

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_CLEARED_ALARM of interface NotificationType of module NotificationIRPConstDefs.

alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
managedObject Class, managedObjectInstance	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
notification Id	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
perceived Severity	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	remaining_ body		

Table 11: Mapping for notifyAlarmListRebuilt

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notification Type	type_name	M	This is the NOTIFY_FM_ALARM_LIST_REBUILT of interface NotificationType of module NotificationIRPConstDefs.
There is no corresponding IS attribute.	event_name	M	Carry an empty string.
There is no corresponding IS attribute.	variable Header		
managedObject Class, managedObjectInstance	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
notification Id	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_ body_fields	O	See that of notifyNewAlarm.

reason	One NV pair of filterable_body_fields	M	It is a string.
There is no corresponding IS attribute.	remaining_body		

Table 12: Mapping for notifyChangedAlarm

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_CHANGED_ALARM of interface NotificationType of module NotificationIRPConstDefs.
alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
notificationId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
perceivedSeverity	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	remaining_body		

Table 13: Mapping for notifyComments

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_CLEARED_ALARM of interface NotificationType of module NotificationIRPConstDefs.

alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
notificationId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
perceivedSeverity	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
comments		M	Name of NV pair is the COMMENTS of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a CommentSet.
There is no corresponding IS attribute.	remaining_body		

6 AlarmIRPNotifications Interface

OMG CORBA Notification push operation is used to realise the notification of AlarmIRPNotifications. All the notifications in this interface are implemented using this push_structured_event method.

6.1 Method push (M)

```
module CosNotifyComm {

      …

      Interface SequencePushConsumer : NotifyPublish {
            void push_structured_events(

                  in CosNotification::EventBatch notifications)

          raises( CosEventComm::Disconnected);

            …

      }; // SequencePushConsumer

      …
```

```
}; // CosNotifyComm
```

NOTE 1: The `push_structured_events` method takes an input parameter of type `EventBatch` as defined in the OMG `CosNotification` module (OMG Notification Service [1]). This data type is the same as a sequence of Structured Events. Upon invocation, this parameter will contain a sequence of Structured Events being delivered to IRPManager by IRPAgent to which it is connected.

NOTE 2: The maximum number of events that will be transmitted within a single invocation of this operation is controlled by IRPAgent wide configuration parameter.

NOTE 3: The amount of time the supplier (IRPAgent) of a sequence of Structured Events will accumulate individual events into the sequence before invoking this operation is controlled by IRPAgent wide configuration parameter as well.

NOTE 4: IRPAgent may push `EventBatch` with only one Structured Event.

Annex A (normative): IDL specification

```
#include "CosNotification.idl"
#include "generic.idl"

#ifndef AlarmIRP_idl
#define AlarmIRP_idl

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

/* ## Module: AlarmIRPConstDefs
This module contains commonly used definitions for Alarm IRP
================================================================
*/
module AlarmIRPConstDefs
{
   /*
   Define the this Alarm IRP version.
   This string is used for the return value of
       get_alarm_IRP_versions().
   It is used as return value of get_notification_categories()
       if the Notification IRP supports the emission of notifications
       defined by this Alarm IRP version.
   It is also used in the domain_name attribute of a structured event
       carrying alarm information defined by this Alarm IRP version.

   See definition "IRP document version number string".
   */
   const string ALARM_IRP_VERSION = "<to be updated using the rule>";

   /*
   This block identifies the alarm types specified for this IRP version.
   These types carry the same semantics as the TMN ITU-T defined event
   types of the same name.
   Their encodings for this version of Alarm IRP are defined here.  Other IRP
   documents, or other versions of Alarm IRP, shall identify their own
   alarm types for their use.  They shall define their encodings
   as well.  Values defined here are unique among themselves.
   */
   interface AlarmType
   {
      const string COMMUNICATIONS_ALARM = "x1";
      const string PROCESSING_ERROR_ALARM = "x2";
      const string ENVIRONMENTAL_ALARM = "x3";
      const string QUALITY_OF_SERVICE_ALARM = "x4";
      const string EQUIPMENT_ALARM = "x5";
   };

   /*
   This block identifies the notification types defined by this
   Alarm IRP version.
   */
   interface NotificationType
   {
      const string NOTIFY_FM_NEW_ALARM = "x1";
      const string NOTIFY_FM_CHANGED_ALARM = "x2";
      const string NOTIFY_FM_ACK_STATE_CHANGED = "x3";
      const string NOTIFY_FM_COMMENT_ADDED = "x4";
      const string NOTIFY_FM_CLEARED_ALARM = "x5";
      const string NOTIFY_FM_ALARM_LIST_REBUILT = "x6";
```

```
};

/*
This block identifies the levels of severity.
*/
interface PerceivedSeverity
{
   const short INDETERMINATE = 1;
   const short CRITICAL = 2;
   const short MAJOR = 3;
   const short MINOR = 4;
   const short WARNING = 5;
   const short CLEARED = 6;
};

/*
This block identifies the probable cause of a reported alarm.
*/
interface ProbableCause
{
   const short ALARM_INDICATION_SIGNAL = 1;
   const short CALL_SETUP_FAILURE = 2;
   const short DEGRADED_SIGNAL_M3100 = 3;
   const short FAR_END_RECEIVER_FAILURE = 4;
   const short FRAMING_ERROR_M3100 = 5;
   const short LOSS_OF_FRAME = 6;
   const short LOSS_OF_POINTER = 7;
   const short LOSS_OF_SIGNAL = 8;
   const short PAYLOAD_TYPE_MISMATCH = 9;
   const short TRANSMISSION_ERROR = 10;
   const short REMOTE_ALARM_INTERFACE = 11;
   const short EXCESSIVE_BIT_ERROR_RATE = 12;
   const short PATH_TRACE_MISMATCH = 13;
   const short UNAVAILABLE = 14;
   const short SIGNAL_LABEL_MISMATCH = 15;
   const short LOSS_OF_MULTI_FRAME = 16;
   const short BACK_PLANE_FAILURE = 51;
   const short DATA_SET_PROBLEM = 52;
   const short EQUIPMENT_IDENTIFIER_DUPLICATION = 53;
   const short EXTERNAL_DEVICE_PROBLEM = 54;
   const short LINE_CARD_PROBLEM = 55;
   const short MULTIPLEXER_PROBLEM_M3100 = 56;
   const short NE_IDENTIFIER_DUPLICATION = 57;
   const short POWER_PROBLEM_M3100 = 58;
   const short PROCESSOR_PROBLEM_M3100 = 59;
   const short PROTECTION_PATH_FAILURE = 60;
   const short RECEIVER_FAILURE_M3100 = 61;
   const short REPLACEABLE_UNIT_MISSING = 62;
   const short REPLACEABLE_UNIT_TYPE_MISMATCH = 63;
   const short SYNCHRONISATION_SOURCE_MISMATCH = 64;
   const short TERMINAL_PROBLEM = 65;
   const short TIMING_PROBLEM_M3100 = 66;
   const short TRANSMITTER_FAILURE_M3100 = 67;
   const short TRUNK_CARD_PROBLEM = 68;
   const short REPLACEABLE_UNIT_PROBLEM = 69;
   const short AIR_COMPRESSOR_FAILURE = 101;
   const short AIR_CONDITIONING_FAILURE = 102;
   const short AIR_DRYER_FAILURE = 103;
   const short BATTERY_DISCHARGING = 104;
   const short BATTERY_FAILURE = 105;
   const short COMMERICAL_POWER_FAILURE = 106;
   const short COOLING_FAN_FAILURE = 107;
   const short ENGINE_FAILURE = 108;
   const short FIRE_DETECTOR_FAILURE = 109;
   const short FUSE_FAILURE = 110;
   const short GENERATOR_FAILURE = 111;
```

```
const short LOW_BATTERY_THRESHOLD = 112;
const short PUMP_FAILURE_M3100 = 113;
const short RECTIFIER_FAILURE = 114;
const short RECTIFIER_HIGH_VOLTAGE = 115;
const short RECTIFIER_LOW_F_VOLTAGE = 116;
const short VENTILATION_SYSTEM_FAILURE = 117;
const short ENCLOSURE_DOOR_OPEN_M3100 = 118;
const short EXPLOSIVE_GAS = 119;
const short FIRE = 120;
const short FLOOD = 121;
const short HIGH_HUMIDITY = 122;
const short HIGH_TEMPERATURE = 123;
const short HIGH_WIND = 124;
const short ICE_BUILD_UP = 125;
const short INTRUSION_DETECTION = 126;
const short LOW_FUEL = 127;
const short LOW_HUMIDITY = 128;
const short LOW_CABLE_PRESSURE = 129;
const short LOW_TEMPERATURE = 130;
const short LOW_WATER = 131;
const short SMOKE = 132;
const short TOXIC_GAS = 133;
const short STORAGE_CAPACITY_PROBLEM_M3100 = 151;
const short MEMORY_MISMATCH = 152;
const short CORRUPT_DATA_M3100 = 153;
const short OUT_OF_CPU_CYCLES = 154;
const short SOFTWARE_ENVIRONMENT_PROBLEM = 155;
const short SOFTWARE_DOWNLOAD_FAILURE = 156;
const short ADAPTER_ERROR = 301;
const short APPLICATION_SUBSYSTEM_FAILURE = 302;
const short BANDWIDTH_REDUCTION = 303;
const short COMMUNICATION_PROTOCOL_ERROR = 305;
const short COMMUNICATION_SUBSYSTEM_FAILURE = 306;
const short CONFIGURATION_OR_CUSTOMIZING_ERROR = 307;
const short CONGESTION = 308;
const short CPU_CYCLES_LIMIT_EXCEEDED = 310;
const short DATA_SET_OR_MODEM_ERROR = 311;
const short DTE_DCE_INTERFACE_ERROR = 313;
const short EQUIPMENT_MALFUNCTION = 315;
const short EXCESSIVE_VIBRATION = 316;
const short FILE_ERROR = 317;
const short HEATING_OR_VENTILATION_OR_COOLING_SYSTEM_PROBLEM = 321;
const short HUMIDITY_UNACCEPTABLE = 322;
const short INPUT_OUTPUT_DEVICE_ERROR = 323;
const short INPUT_DEVICE_ERROR = 324;
const short LAN_ERROR = 325;
const short LEAK_DETECTION = 326;
const short LOCAL_NODE_TRANSMISSION_ERROR = 327;
const short MATERIAL_SUPPLY_EXHAUSTED = 330;
const short OUT_OF_MEMORY = 332;
const short OUTPUT_DEVICE_ERROR = 333;
const short PERFORMANCE_DEGRADED = 334;
const short PRESSURE_UNACCEPTABLE = 336;
const short QUEUE_SIZE_EXCEEDED = 339;
const short RECEIVE_FAILURE = 340;
const short REMOTE_NODE_TRANSMISSION_ERROR = 342;
const short RESOURCE_AT_OR_NEARING_CAPACITY = 343;
const short RESPONSE_TIME_EXCESSIVE = 344;
const short RETRANSMISSION_RATE_EXCESSIVE = 345;
const short SOFTWARE_ERROR = 346;
const short SOFTWARE_PROGRAM_ABNORMALLY_TERMINATED = 347;
const short SOFTWARE_PROGRAM_ERROR = 348;
const short TEMPERATURE_UNACCEPTABLE = 350;
const short THRESHOLD_CROSSED = 351;
const short TOXIC_LEAK_DETECTED = 353;
const short TRANSMIT_FAILURE = 354;
```

```
    const short UNDERLYING_RESOURCE_UNAVAILABLE = 356;
    const short VERSION_MISMATCH = 357;
    const short A_BIS_TO_BTS_INTERFACE_FAILURE = 501;
    const short A_BIS_TO_TRX_INTERFACE_FAILURE = 502;
    const short ANTENNA_PROBLEM = 503;
    const short BATTERY_BREAKDOWN = 504;
    const short BATTERY_CHARGING_FAULT = 505;
    const short CLOCK_SYNCHRONISATION_PROBLEM = 506;
    const short COMBINER_PROBLEM = 507;
    const short DISK_PROBLEM = 508;
    const short EXCESSIVE_RECEIVER_TEMPERATURE = 510;
    const short EXCESSIVE_TRANSMITTER_OUTPUT_POWER = 511;
    const short EXCESSIVE_TRANSMITTER_TEMPERATURE = 512;
    const short FREQUENCY_HOPPING_DEGRADED = 513;
    const short FREQUENCY_HOPPING_FAILURE = 514;
    const short FREQUENCY_REDEFINITION_FAILED = 515;
    const short LINE_INTERFACE_FAILURE = 516;
    const short LINK_FAILURE = 517;
    const short LOSS_OF_SYNCHRONISATION = 518;
    const short LOST_REDUNDANCY = 519;
    const short MAINS_BREAKDOWN_WITH_BATTERY_BACKUP = 520;
    const short MAINS_BREAKDOWN_WITHOUT_BATTERY_BACKUP = 521;
    const short POWER_SUPPLY_FAILURE = 522;
    const short RECEIVER_ANTENNA_FAULT = 523;
    const short RECEIVER_MULTICOUPLER_FAILURE = 525;
    const short REDUCED_TRANSMITTER_OUTPUT_POWER = 526;
    const short SIGNAL_QUALITY_EVALUATION_FAULT = 527;
    const short TIMESLOT_HARDWARE_FAILURE = 528;
    const short TRANSCEIVER_PROBLEM = 529;
    const short TRANSCODER_PROBLEM = 530;
    const short TRANSCODER_OR_RATE_ADAPTER_PROBLEM = 531;
    const short TRANSMITTER_ANTENNA_FAILURE = 532;
    const short TRANSMITTER_ANTENNA_NOT_ADJUSTED = 533;
    const short TRANSMITTER_LOW_VOLTAGE_OR_CURRENT = 535;
    const short TRANSMITTER_OFF_FREQUENCY = 536;
    const short DATABASE_INCONSISTENCY = 537;
    const short FILE_SYSTEM_CALL_UNSUCCESSFUL = 538;
    const short INPUT_PARAMETER_OUT_OF_RANGE = 539;
    const short INVALID_PARAMETER = 540;
    const short INVALID_POINTER = 541;
    const short MESSAGE_NOT_EXPECTED = 542;
    const short MESSAGE_NOT_INITIALISED = 543;
    const short MESSAGE_OUT_OF_SEQUENCE = 544;
    const short SYSTEM_CALL_UNSUCCESSFUL = 545;
    const short TIMEOUT_EXPIRED = 546;
    const short VARIABLE_OUT_OF_RANGE = 547;
    const short WATCH_DOG_TIMER_EXPIRED = 548;
    const short COOLING_SYSTEM_FAILURE = 549;
    const short EXTERNAL_EQUIPMENT_FAILURE = 550;
    const short EXTERNAL_POWER_SUPPLY_FAILURE = 551;
    const short EXTERNAL_TRANSMISSION_DEVICE_FAILURE = 552;
    const short REDUCED_ALARM_REPORTING = 561;
    const short REDUCED_EVENT_REPORTING = 562;
    const short RECUCED_LOGGING_CAPABILITY = 563;
    const short SYSTEM_RESOURCES_OVERLOAD = 564;
    const short BROADCAST_CHANNEL_FAILURE = 565;
    const short CALL_ESTABLISHMENT_ERROR = 566;
    const short INVALID_MESSAGE_RECEIVED = 567;
    const short INVALID_MSU_RECEIVED = 568;
    const short LAPD_LINK_PROTOCOL_FAILURE = 569;
    const short LOCAL_ALARM_INDICATION = 570;
    const short REMOTE_ALARM_INDICATION = 571;
    const short ROUTING_FAILURE = 572;
    const short SS7_PROTOCOL_FAILURE = 573;
    const short TRANSMISSION_FAILURE = 574;
};
```

```
/*
This block identifies the acknowledgement state of a reported alarm.
*/
interface AckState
{
   const short ACKNOWLEDGED = 1;
   const short UNACKNOWLEDGED = 2;
};

/*
This block identifies attributes which are included as part of the Alarm IRP
These attribute values should not clash with those defined for the attributes
of notification header (see IDL of Notification IRP).
*/
interface AttributeNameValue
{
   const string ALARM_ID = "f";
   const string PROBABLE_CAUSE = "g";
   const string PERCEIVED_SEVERITY = "h";
   const string SPECIFIC_PROBLEM = "i";
   const string ADDITIONAL_TEXT = "j";
   const string ACK_TIME = "k";
   const string ACK_USER_ID = "l";
   const string ACK_SYSTEM_ID = "m";
   const string ACK_STATE = "n";
   const string COMMENTS = "o";
   const string BACKED_UP_STATUS = "p";
   const string BACK_UP_OBJECT = "q";
   const string THRESHOLD_INFO = "r";
   const string TREND_INDICATION = "s";
   const string STATE_CHANGE_DEFINITION = "t";
   const string MONITORED_ATTRIBUTES = "u";
   const string PROPOSED_REPAIR_ACTIONS = "v";
   const string CORRELATED_NOTIFICATIONS = "w";
   const string REASON = "x";
};

/*
Defines the content of a Comment
*/
struct Comment
{
   ManagedGenericIRPConstDefs::IRPTime comment_time;
   string comment_text;
   string user_id;
   string system_id;
};

/*
Defines a set of comments which are placed in the COMMENTS attribute
of a structured event.
*/
typedef sequence <Comment> CommentSet;

/*
It indicates if an object has a back up.
True implies backed up.  False implies not backed up.
*/
typedef boolean BackedUpStatusType;

/*
It indicates if the threshold crossed was in the up or down direction.
*/
enum ThresholdIndicationType {Up, Down};
```

```
   /*
   It indicates if some observed condition is getting better, worse,
   or not changing.
   */
   enum TrendIndicationType {LessSevere, NoChange, MoreSevere};

   /*
   It is used to report a changed attribute value.
   */
   struct AttributeValueChangeType
   {
      string attribute_name;
      any    old_value;  // type depends on attribute
      any    new_value;  // type depends on attribute
   };

   typedef sequence <AttributeValueChangeType> AttributeChangeSetType;

   /*
   It is used to report an attribute and its value.
   */
   struct AttributeValueType
   {
      string attribute_name;
      any    value;  // type depends on the attribute
   };

   typedef sequence <AttributeValueType> AttributeSetType;

   typedef sequence <long> NotifIdSetType;

   /*
   This holds identifiers of notifications that are correlated.
   */
   struct CorelatedNotification
   {
      string source;  // Contains DN of MO that emitted the set of notifications
                      // DN string format in compliance with Name Convention for
                      // Managed Object.
                      // This may be a zero-length string.  In this case, the MO
                      // is identified by the value of the MOI attribute
                      // of the Structured Event, i.e., the notification.
      NotifIdSetType notif_id_set;  // Set of related notification ids
   };

   /*
   Correlated Notification sets are sets of Correlated Notification
   structures.
   */
   typedef sequence <CorelatedNotification> CorrelatedNotificationSetType;

   /*
   Define the structure returned when an operation fails for a set of alarm ids.
   A reason is provided in order to indicate why the operation failed.
   */
   struct BadAlarmInformationIdSeq
   {
      string alarm_information_reference;
      string reason;
   };

   typedef sequence <string> AlarmInformationIdSeq;
   typedef CosNotification::EventBatch AlarmInformationSeq;
};

/* ## Module: AlarmIRPSystem
```

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

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

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

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

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


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

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

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


/*
Request to acknowledge one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal acknowledge_alarms (
   in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
   in string ack_user_id,
   in string ack_system_id,
   out AlarmIRPConstDefs::BadAlarmInformationIdSeq
      bad_alarm_information_id_list
)
raises (AcknowledgeAlarms, ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);


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


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

```
            ManagedGenericIRPSystem::ParameterNotSupported,
            ManagedGenericIRPSystem::InvalidParameter);

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


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

#endif
```

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2000	S_07	SP-000012	-		Approved at TSG SA #7 and placed under Change Control	2.0.0	3.0.0
Mar 2000		-	-		cosmetic	3.0.0	3.0.1
Jun 2000	S_08	SP-000253	005		Split of TS - Part 3: Alarm Integration Reference Point (IRP): CORBA Solution Set (SS)	3.0.1	3.1.0
Sep 2000	S_09	SP-000439	003		Correct push_structured_event of push_structured_events	3.1.0	3.2.0
Sep 2000	S_09	SP-000439	004		Remove the use of interface to encapsulate const strings	3.1.0	3.2.0
Dec 2000	S_10	SP-000521	001	1	Allow “Structured Event Filterable Body Fields” to be absent if parameters are not used	3.2.0	3.3.0
Dec 2000	S_10	SP-000521	002	1	Specific behaviour of the Iterator	3.2.0	3.3.0
Dec 2000	S_10	SP-000521	005		Inconsistent qualifiers	3.2.0	3.3.0
Mar 2001	S_11	SP-010032	006		Missing how "Notify Alarm List Rebuilt" reason attribute is located in Structured Event	3.3.0	3.4.0
Mar 2001	S_11	SP-010032	007		Use alarmInformationBody in additionalInformation.ackTime	3.3.0	3.4.0
Jun 2001	S_12	SP-010239	008		Probable Cause “Intrusion Detection” is missing	3.4.0	3.5.0
Jun 2001	S_12		009		Alarm IRP: CORBA SS Rel4 - Addition of feature	3.5.1	4.0.0

3GPP TS 32.111-3 V3.5.1 (2001-06)

Technical Specification

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

Keywords
Fault Management, Alarms

3GPP

Postal address

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

Internet
http://www.3gpp.org

Contents

Foreword

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

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

Part 1: "3G Fault Management Requirements";

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

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

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

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

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

1 Scope

The present document specifies the CORBA Solution Set (SS) for the IRP whose semantics is specified in Alarm IRP: Information Service (IS) (3G TS 32.111-2 [6]).

Clause 1 to 3 provides background information. Clause 4 provides key architectural features supporting the SS. Clause 5 defines the mapping of operations, notification, parameters and attributes defined in IS to their SS equivalents. Clause 6 describes the notification interface containing the push method. Annex A contains the IDL specification.

2 References

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

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

[1] OMG TC Document telecom/98-11-01: “OMG Notification Service”.

[2] OMG CORBA Services: “Common Object Services Specification, Update: November 22, 1996” (Clause 4 contains the Event Service specification).

[3] 3G TS 32.106-8: “Name Convention for Managed Objects”.

[4] 3G TS 32.106-2: “Notification IRP: Information Service”.

[5] 3G TS 32.106-3: “Notification IRP: CORBA Solution Set”.

[6] 3G TS 32.111-2: “Alarm Integration Reference Point: Information Service”.

3 Definitions and abbreviations

3.1 Definitions

In addition to the terms and definitions defined in TS 32.111-2 [6], there are no additional definitions applicable to the present document.

3.2 Abbreviations

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

CORBA	Common Object Request Broker Architecture
IDL	Interface Definition Language
IRP	Integration Reference Point
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
OMG	Object Management Group
TMN	Telecommunications Management Network
UML	Unified Model Language

3.3 IRP document version number string

The IRP document version number (sometimes called “IRP version” or “version number”) string is used to identify this specification. The string is derived using the following rule.

Take the 3GPP document number on the front page of this specification, such as “3GPP TS 32.106-3 V3.2.0 (2000-12)”. Discard the leading “3GPP TS ”. Discard all characters after and including the last period. Eliminate leading and trailing spaces. Reduce multiple consecutive spaces with one space. Express the resultant in a string. Capitalised the string. For example, if the 3GPP document version number is “3GPP TS 32.106-3 V3.2.0 (2000-12)”, then the IRP document version number shall be “32.106 V3.2”.

This string is returned in `getAlarmIRPVersion` method and is carried in the first field of the notification header of all notifications related to alarm IRP.

4 Architectural Features

The overall architectural feature of Alarm IRP is specified in 3G TS 32.111-2 [6]. This clause specifies features that are specific to the CORBA SS.

4.1 Notification Services

In implementations of CORBA SS, IRPAgent conveys Alarm Information to IRPManager via OMG Notification Service (OMG Notification Service [1]).

OMG Event Service [2] provides event routing and distribution capabilities. OMG Notification Service provides, in addition to Event Service, event filtering and Quality Of Service (QOS) as well.

A necessary and sufficient sub set of OMG Notification Services shall be used to support `AlarmIRPNotifications` notifications as specified in 3G TS 32.111-2 [6].

4.2 Push and Pull Style

OMG Notification Service defines two styles of interaction. One is called push style. In this style, IRPAgent pushes notifications to IRPManager as soon as they are available. The other is called pull style. In this style, IRPAgent keeps the notifications till IRPManager requests for them.

This CORBA SS specifies that support of Push style is Mandatory (M) and that support of Pull style is Optional (O).

4.3 Support multiple notifications in one push operation

For efficiency reasons, IRPAgent may send multiple notifications using one single `push` operation. To pack multiple notifications into one `push` operation, IRPAgent may wait and not invoke the `push` operation as soon as notifications are available. To avoid IRPAgent to wait for an extended period of time that is objectionable to IRPManager, IRPAgent shall implement an IRPAgent wide timer configurable by administrator. On expiration of this timer, IRPAgent shall invoke `push` if there is at least one notification to be conveyed to IRPManager. This timer is re-started after each `push` invocation.

4.4 Filter

IRPAgent shall optionally support alarm filtering based on IRPManager's supplied alarm filter constraints (e.g., as parameter in `subscribe`() of 3G TS 32.106-2 [4]. Alarm filtering can be applied in the following cases:

- It is applicable to alarms emitted by IRPAgent via `AlarmIRPNotifications`. IRPManager supplies alarm filter constraint via the `subscribe` method. This filter is effective during the period of subscription.
- It is applicable to alarms returned by IRPAgent via the `out` parameter of `get_alarm_list` method. IRPManager supplies alarm filter constraint via the `get_alarm_list` method. This filter is effective only for this method invocation.
- It is applicable to the calculation of alarm counts returned by IRPAgent via the `out` parameters of `get_alarm_count` method. IRPManager supplies alarm filter constraint via the `get_alarm_count` method. This filter is effective only for this method invocation.

This SS shall use of filter constraint grammar specified by reference OMG Notification Service [1]. The name of the grammar is called "`EXTENDED_TCL`". See clause 2.4, Default Filter Constraint Language in OMG Notification Service [1]. This SS shall use this grammar only.

5 Mapping

5.1 Operation and Notification mapping

Alarm IRP: IS 3G TS 32.111-2 [6] defines semantics of operation and notification visible across the Alarm IRP. Table 1 indicates mapping of these operations and notifications to their equivalents defined in this SS.

Table 1: Mapping from IS Notification/Operation to SS equivalents

IS Operation/ notification 3G TS 32.111-2 [13]	SS Method	Qualifier
`acknowledgeAlarms`	`acknowledge_alarms`	M
`unacknowledgeAlarms`	`unacknowledge_alarms`	O
`getAlarmList`	`get_alarm_list`	M
`getAlarmIRPVersion`	`get_alarm_IRP_version`	M
`getAlarmCount`	`get_alarm_count`	O
`setComment`	`set_comment`	O
`notifyNewAlarm`	`push_structured_event` Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 6.1	M
`notifyClearedAlarm`	`push_structured_event`See clause 6.1	M
`notifyChangedAlarm`	`push_structured_event`See clause 6.1	M
`notifyAckStateChanged`	`push_structured_event`See clause 6.1	M
`notifyAlarmListRebuilt`	`push_structured_event` `See clause 6.1`	M
`notifyComments`	`push_structured_event`See clause 6.1	O

5.2 Operation parameter mapping

Reference 3G TS 32.111-2 [6] defines semantics of parameters carried in operations across the Alarm IRP. The following set of tables indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

Table 2: Mapping from IS `acknowledgeAlarms` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarmInformation ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list`	M
`ackUserId`	`string ack_user_id`	M
`ackSystemId`	`string ack_system_id`	O
`bad AlarmInformation ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq bad_alarm_information_id_list`	M

`status`	`CommonIRPConstDefs::Signal`
	Exceptions:
	`AcknowledgeAlarms, ParameterNotSupported, InvalidParameter`
		M

Table 3: Mapping from IS `unacknowledgeAlarms` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarm InformationReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list`	M
`ackUserId`	`string ack_user_id`	M
`ackSystemId`	`string ack_system_id`	O
`badAlarm Information ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq bad_alarm_information_id_list`	M

`status`	`CommonIRPConstDefs::Signal`
	Exceptions:
	`UnacknowledgeAlarms, OperationNotSupported, ParameterNotSupported, InvalidParameter`
		M

Table 4: Mapping from IS `getAlarmList` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarmAckState, filter`	`string filter`	O
`alarmInformation List`	Return value of type `AlarmIRPConstDefs::AlarmInformationSeq`	M
`status`	Exceptions: `GetAlarmList, ParameterNotSupported, InvalidParameter`	M

Table 5: Mapping from IS `getAlarmCount` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`alarmAckState, filter`	`string filter`	O
`criticalCount, majorCount, minorCount, warningCount, indeterminateCount,clearedCount`	`long critical_count, long major_count, long minor_count, long warning_count, long indeterminate_count, long cleared_count`	M
`status`	Exceptions: `GetAlarmCount, OperationNotSupported, ParameterNotSupported, InvalidParameter`	M

Table 6: Mapping from IS `getAlarmIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`versionNumberList`	Return value of type `CommonIRPConstDefs::VersionNumberSet`	M
`status`	Exceptions: `GetAlarmIRPVersion`	M

Table 7: Mapping from IS `setComment` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`AlarmInformation ReferenceList`	`AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list`	M
`commentUserId`	`string comment_user_id`	M
`commentSystemId`	`string comment_system_id`	M
`commentText`	`string comment_text`	M
`badAlarmInformationIdList`	`AlarmIRPConstDefs::BadAlarmInformationIdSe bad_alarm_information_id_list`	
`status`	Exceptions: `CommentAlarms,OperationNotSupported.`	

5.3 Notification parameter mapping

Reference 3G TS 32.111-2 [6] defines semantics of parameters carried in notifications. The following tables indicate the mapping of these parameters to their OMG CORBA Structured Event (defined in OMG Notification Service [1]) equivalents. The composition of OMG Structured Event, as defined in the OMG Notification Service [1], is:

```
Header
      Fixed Header
           domain_name
           type_name
           event_name
      Variable Header
Body
      filterable_body_fields
      remaining_body
```

The following tables list all OMG Structured Event attributes in the second column. The first column identifies the Alarm IRP: IS [6] defined notification parameters.

Table 8: Mapping for `notifyNewAlarm`

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding SS attribute.	`domain_name`		It carries the IRP document version number string. See sub-clause 3.3. It indicates the syntax and semantics of the Structured Event as defined by this specification.
`notification Type`	`type_name`	`M`	This is the `NOTIFY_FM_NEW_ALARM` of `interface NotificationType` of module `NotificationIRPConstDefs`.
`alarmType`	`event_name`	`M`	It identifies one of the following: • communications alarm,

			• processing error alarm, • environmental alarm, • quality of service alarm and • equipment alarm. It is a string. See block of const string definitions encapsulated by `interface AlarmTypes` in the IDL. The strings start with "ET_".
There is no corresponding SS attribute.	`variable Header`		
`managedObjectClass, managedObjectInstance`	One NV pair of `filterable_body_fields`	M	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. Name of NV pair is the `MANAGED_OBJECT_INSTANCE` of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS (3G TS 32.106-3 [5]).
`notificationId`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the NOTIFICATION_ID of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a long. See corresponding table in Notification IRP: CORBA SS (3G TS 32.106-3 [5]).
`eventTime`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the EVENT_TIME of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a `IRPTime`. See corresponding table in Notification IRP: CORBA SS (3G TS 32.106-3 [5]).
`systemDN`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the SYSTEM_DN of `interface AttributeNameValue` of `module` NotificationIRPConstDefs. Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS [5].
`probableCause`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the `PROBABLE_CAUSE` of `interface AttributeNameValue of module AlarmIRPConstDefs`. Value of NV pair is a short defined by `interface ProbableCause`.
`perceivedSeverity`	One NV pair of `filterable_body_fields`	M	Name of NV pair is the PERCEIVED_SEVERITY of `interface AttributeNameValue of module AlarmIRPConstDefs`. Value of NV pair is a short defined by `interface PerceivedSeverity`.
`specificProblem`	One NV pair of `filterable_body_fields`	O	Name of NV pair is the SPECIFIC_PROBLEM of `interface AttributeNameValue` of `module AlarmIRPConstDefs.` Value of NV pair is a string.
`correlatedNotifications`	One NV pair of `filterable_body_fields`	O	Name of NV pair is the CORRELATED_NOTIFICATIONS of `interface AttributeNameValue`. Value of NV pair is a `CorrelatedNotificationSetType.`
`backedUpStatus`	One NV pair of `filterable_body_fields`	O	Name of NV pair is the BACKED_UP_STATUS of `interface AttributeNameValue` of `module AlarmIRPConstDefs` Value of NV pair is a `boolean BackedUpStatusType`.

backUpObject	One NV pair of filterable_body_fields	O	Name of NV pair is the BACKED_UP_OBJECT of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string carrying of DN of the back-up object. See 3G TS 32.106-8 [3] for the DN string representation.
trendIndication	One NV pair of filterable_body_fields	O	Name of NV pair is the TREND_INDICATION of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is an enum TrendIndicationType.
thresholdInfo	One NV pair of filterable_body_fields	O	Name of NV pair is the THRESHOLD_INFO of interface ParameterNameValue of module AlarmIRPConstDefs. Value of NV pair is an enum ThresholdIndicationType.
stateChangeDefinition	One NV pair of filterable_body_fields	O	Name of NV pair is the STATE_CHANGE_DEFINITION of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is an AttributeChangeSetType.
monitoredAttributes	One NV pair of filterable_body_fields	O	Name of NV pair is the MONITORED_ATTRIBUTES of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is an AttributeSetType.
proposedRepairActions	One NV pair of filterable_body_fields	O	Name of NV pair is the PROPOSED_REPAIR_ACTIONS of inteface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
additionalText	One NV pair of filterable_body_fields	O	Name of NV pair is the ADDITIONAL_TEXT of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
alarmId	One NV pair of filterable_body_fields	M	Name of NV pair is the ALARM_ID of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string. If the string is a zero-length string or if this NV pair is absent, the default semantics is that alarmId is a concatenation ofmanagedObjectInstance, eventType, probableCause and specificProblem, if present, of this Structured Event. Since probableCuase is encoded as a short, it shall be converted into string before concatenation. The resultant string shall not contain spaces.
There is no corresponding IS attribute.	remaining_body		

Table 9: Mapping for notifyAckStateChanged

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_ACK_STATE_CHANGED of interface NotificationType of module NotificationIRPConstDefs.
alarmType	event_name	M	See that of notifyNewAlarm.

There is no corresponding IS attribute.	variable Header		
managedObject Class, managedObjectInstance	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
notificationId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
perceivedSeverity	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
ackTime	One NV pair of filterable_body_fields	M	Name of NV pair is the ACK_TIME of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a IRPTime of module ManagedGenericIRPConstDefs.
ackUserId	One NV pair of filterable_body_fields	M	Name of NV pair is the ACK_USER_ID of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
ackSystemId	One NV pair of filterable_body_fields	O	Name of NV pair is the ACK_SYSTEM_ID of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
ackState	One NV pair of filterable_body_fields	M	Name of NV pair is the ACK_STATE of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a short defined by interface AckState of module AlarmIRPConstDefs.
There is no corresponding IS attribute.	remaining_body		

Table 10: Mapping for notifyClearedAlarm

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_CLEARED_ALARM of interface NotificationType of module NotificationIRPConstDefs.

alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
managedObject Class, managedObjectInstance	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
notification Id	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
perceived Severity	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	remaining_ body		

Table 11: Mapping for notifyAlarmListRebuilt

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notification Type	type_name	M	This is the NOTIFY_FM_ALARM_LIST_REBUILT of interface NotificationType of module NotificationIRPConstDefs.
There is no corresponding IS attribute.	event_name	M	Carry an empty string.
There is no corresponding IS attribute.	variable Header		
managedObject Class, managedObjectInstance	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
notification Id	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_ body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_ body_fields	O	See that of notifyNewAlarm.

reason	One NV pair of filterable_body_fields	M	It is a string.
There is no corresponding IS attribute.	remaining_body		

Table 12: Mapping for notifyChangedAlarm

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_CHANGED_ALARM of interface NotificationType of module NotificationIRPConstDefs.
alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
notificationId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
perceivedSeverity	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	remaining_body		

Table 13: Mapping for notifyComments

IS Parameters	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name		See that of notifyNewAlarm.
notificationType	type_name	M	This is the NOTIFY_FM_CLEARED_ALARM of interface NotificationType of module NotificationIRPConstDefs.

alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
notificationId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
eventTime	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
probableCause	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
perceivedSeverity	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
alarmId	One NV pair of filterable_body_fields	M	See that of notifyNewAlarm.
comments		M	Name of NV pair is the COMMENTS of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a CommentSet.
There is no corresponding IS attribute.	remaining_body		

6 AlarmIRPNotifications Interface

OMG CORBA Notification push operation is used to realise the notification of AlarmIRPNotifications. All the notifications in this interface are implemented using this push_structured_event method.

6.1 Method push (M)

```
module CosNotifyComm {

      …

      Interface SequencePushConsumer : NotifyPublish {
            void push_structured_events(

                  in CosNotification::EventBatch notifications)

          raises( CosEventComm::Disconnected);

            …

      }; // SequencePushConsumer

      …
```

```
}; // CosNotifyComm
```

NOTE 1: The `push_structured_events` method takes an input parameter of type `EventBatch` as defined in the OMG `CosNotification` module (OMG Notification Service [1]). This data type is the same as a sequence of Structured Events. Upon invocation, this parameter will contain a sequence of Structured Events being delivered to IRPManager by IRPAgent to which it is connected.

NOTE 2: The maximum number of events that will be transmitted within a single invocation of this operation is controlled by IRPAgent wide configuration parameter.

NOTE 3: The amount of time the supplier (IRPAgent) of a sequence of Structured Events will accumulate individual events into the sequence before invoking this operation is controlled by IRPAgent wide configuration parameter as well.

NOTE 4: IRPAgent may push `EventBatch` with only one Structured Event.

Annex A (normative): IDL specification

```
#include "CosNotification.idl"
#include "generic.idl"

#ifndef AlarmIRP_idl
#define AlarmIRP_idl

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

/* ## Module: AlarmIRPConstDefs
This module contains commonly used definitions for Alarm IRP
================================================================
*/
module AlarmIRPConstDefs
{
   /*
   Define the this Alarm IRP version.
   This string is used for the return value of
       get_alarm_IRP_versions().
   It is used as return value of get_notification_categories()
       if the Notification IRP supports the emission of notifications
       defined by this Alarm IRP version.
   It is also used in the domain_name attribute of a structured event
       carrying alarm information defined by this Alarm IRP version.

   See definition "IRP document version number string".
   */
   const string ALARM_IRP_VERSION = "<to be updated using the rule>";

   /*
   This block identifies the alarm types specified for this IRP version.
   These types carry the same semantics as the TMN ITU-T defined event
   types of the same name.
   Their encodings for this version of Alarm IRP are defined here.  Other IRP
   documents, or other versions of Alarm IRP, shall identify their own
   alarm types for their use.  They shall define their encodings
   as well.  Values defined here are unique among themselves.
   */
   interface AlarmType
   {
      const string COMMUNICATIONS_ALARM = "x1";
      const string PROCESSING_ERROR_ALARM = "x2";
      const string ENVIRONMENTAL_ALARM = "x3";
      const string QUALITY_OF_SERVICE_ALARM = "x4";
      const string EQUIPMENT_ALARM = "x5";
   };

   /*
   This block identifies the notification types defined by this
   Alarm IRP version.
   */
   interface NotificationType
   {
      const string NOTIFY_FM_NEW_ALARM = "x1";
      const string NOTIFY_FM_CHANGED_ALARM = "x2";
      const string NOTIFY_FM_ACK_STATE_CHANGED = "x3";
      const string NOTIFY_FM_COMMENT_ADDED = "x4";
      const string NOTIFY_FM_CLEARED_ALARM = "x5";
      const string NOTIFY_FM_ALARM_LIST_REBUILT = "x6";
```

```
};

/*
This block identifies the levels of severity.
*/
interface PerceivedSeverity
{
   const short INDETERMINATE = 1;
   const short CRITICAL = 2;
   const short MAJOR = 3;
   const short MINOR = 4;
   const short WARNING = 5;
   const short CLEARED = 6;
};

/*
This block identifies the probable cause of a reported alarm.
*/
interface ProbableCause
{
   const short ALARM_INDICATION_SIGNAL = 1;
   const short CALL_SETUP_FAILURE = 2;
   const short DEGRADED_SIGNAL_M3100 = 3;
   const short FAR_END_RECEIVER_FAILURE = 4;
   const short FRAMING_ERROR_M3100 = 5;
   const short LOSS_OF_FRAME = 6;
   const short LOSS_OF_POINTER = 7;
   const short LOSS_OF_SIGNAL = 8;
   const short PAYLOAD_TYPE_MISMATCH = 9;
   const short TRANSMISSION_ERROR = 10;
   const short REMOTE_ALARM_INTERFACE = 11;
   const short EXCESSIVE_BIT_ERROR_RATE = 12;
   const short PATH_TRACE_MISMATCH = 13;
   const short UNAVAILABLE = 14;
   const short SIGNAL_LABEL_MISMATCH = 15;
   const short LOSS_OF_MULTI_FRAME = 16;
   const short BACK_PLANE_FAILURE = 51;
   const short DATA_SET_PROBLEM = 52;
   const short EQUIPMENT_IDENTIFIER_DUPLICATION = 53;
   const short EXTERNAL_DEVICE_PROBLEM = 54;
   const short LINE_CARD_PROBLEM = 55;
   const short MULTIPLEXER_PROBLEM_M3100 = 56;
   const short NE_IDENTIFIER_DUPLICATION = 57;
   const short POWER_PROBLEM_M3100 = 58;
   const short PROCESSOR_PROBLEM_M3100 = 59;
   const short PROTECTION_PATH_FAILURE = 60;
   const short RECEIVER_FAILURE_M3100 = 61;
   const short REPLACEABLE_UNIT_MISSING = 62;
   const short REPLACEABLE_UNIT_TYPE_MISMATCH = 63;
   const short SYNCHRONISATION_SOURCE_MISMATCH = 64;
   const short TERMINAL_PROBLEM = 65;
   const short TIMING_PROBLEM_M3100 = 66;
   const short TRANSMITTER_FAILURE_M3100 = 67;
   const short TRUNK_CARD_PROBLEM = 68;
   const short REPLACEABLE_UNIT_PROBLEM = 69;
   const short AIR_COMPRESSOR_FAILURE = 101;
   const short AIR_CONDITIONING_FAILURE = 102;
   const short AIR_DRYER_FAILURE = 103;
   const short BATTERY_DISCHARGING = 104;
   const short BATTERY_FAILURE = 105;
   const short COMMERICAL_POWER_FAILURE = 106;
   const short COOLING_FAN_FAILURE = 107;
   const short ENGINE_FAILURE = 108;
   const short FIRE_DETECTOR_FAILURE = 109;
   const short FUSE_FAILURE = 110;
   const short GENERATOR_FAILURE = 111;
```

```
const short LOW_BATTERY_THRESHOLD = 112;
const short PUMP_FAILURE_M3100 = 113;
const short RECTIFIER_FAILURE = 114;
const short RECTIFIER_HIGH_VOLTAGE = 115;
const short RECTIFIER_LOW_F_VOLTAGE = 116;
const short VENTILATION_SYSTEM_FAILURE = 117;
const short ENCLOSURE_DOOR_OPEN_M3100 = 118;
const short EXPLOSIVE_GAS = 119;
const short FIRE = 120;
const short FLOOD = 121;
const short HIGH_HUMIDITY = 122;
const short HIGH_TEMPERATURE = 123;
const short HIGH_WIND = 124;
const short ICE_BUILD_UP = 125;
const short INTRUSION_DETECTION = 126;
const short LOW_FUEL = 127;
const short LOW_HUMIDITY = 128;
const short LOW_CABLE_PRESSURE = 129;
const short LOW_TEMPERATURE = 130;
const short LOW_WATER = 131;
const short SMOKE = 132;
const short TOXIC_GAS = 133;
const short STORAGE_CAPACITY_PROBLEM_M3100 = 151;
const short MEMORY_MISMATCH = 152;
const short CORRUPT_DATA_M3100 = 153;
const short OUT_OF_CPU_CYCLES = 154;
const short SOFTWARE_ENVIRONMENT_PROBLEM = 155;
const short SOFTWARE_DOWNLOAD_FAILURE = 156;
const short ADAPTER_ERROR = 301;
const short APPLICATION_SUBSYSTEM_FAILURE = 302;
const short BANDWIDTH_REDUCTION = 303;
const short COMMUNICATION_PROTOCOL_ERROR = 305;
const short COMMUNICATION_SUBSYSTEM_FAILURE = 306;
const short CONFIGURATION_OR_CUSTOMIZING_ERROR = 307;
const short CONGESTION = 308;
const short CPU_CYCLES_LIMIT_EXCEEDED = 310;
const short DATA_SET_OR_MODEM_ERROR = 311;
const short DTE_DCE_INTERFACE_ERROR = 313;
const short EQUIPMENT_MALFUNCTION = 315;
const short EXCESSIVE_VIBRATION = 316;
const short FILE_ERROR = 317;
const short HEATING_OR_VENTILATION_OR_COOLING_SYSTEM_PROBLEM = 321;
const short HUMIDITY_UNACCEPTABLE = 322;
const short INPUT_OUTPUT_DEVICE_ERROR = 323;
const short INPUT_DEVICE_ERROR = 324;
const short LAN_ERROR = 325;
const short LEAK_DETECTION = 326;
const short LOCAL_NODE_TRANSMISSION_ERROR = 327;
const short MATERIAL_SUPPLY_EXHAUSTED = 330;
const short OUT_OF_MEMORY = 332;
const short OUTPUT_DEVICE_ERROR = 333;
const short PERFORMANCE_DEGRADED = 334;
const short PRESSURE_UNACCEPTABLE = 336;
const short QUEUE_SIZE_EXCEEDED = 339;
const short RECEIVE_FAILURE = 340;
const short REMOTE_NODE_TRANSMISSION_ERROR = 342;
const short RESOURCE_AT_OR_NEARING_CAPACITY = 343;
const short RESPONSE_TIME_EXCESSIVE = 344;
const short RETRANSMISSION_RATE_EXCESSIVE = 345;
const short SOFTWARE_ERROR = 346;
const short SOFTWARE_PROGRAM_ABNORMALLY_TERMINATED = 347;
const short SOFTWARE_PROGRAM_ERROR = 348;
const short TEMPERATURE_UNACCEPTABLE = 350;
const short THRESHOLD_CROSSED = 351;
const short TOXIC_LEAK_DETECTED = 353;
const short TRANSMIT_FAILURE = 354;
```

```
    const short UNDERLYING_RESOURCE_UNAVAILABLE = 356;
    const short VERSION_MISMATCH = 357;
    const short A_BIS_TO_BTS_INTERFACE_FAILURE = 501;
    const short A_BIS_TO_TRX_INTERFACE_FAILURE = 502;
    const short ANTENNA_PROBLEM = 503;
    const short BATTERY_BREAKDOWN = 504;
    const short BATTERY_CHARGING_FAULT = 505;
    const short CLOCK_SYNCHRONISATION_PROBLEM = 506;
    const short COMBINER_PROBLEM = 507;
    const short DISK_PROBLEM = 508;
    const short EXCESSIVE_RECEIVER_TEMPERATURE = 510;
    const short EXCESSIVE_TRANSMITTER_OUTPUT_POWER = 511;
    const short EXCESSIVE_TRANSMITTER_TEMPERATURE = 512;
    const short FREQUENCY_HOPPING_DEGRADED = 513;
    const short FREQUENCY_HOPPING_FAILURE = 514;
    const short FREQUENCY_REDEFINITION_FAILED = 515;
    const short LINE_INTERFACE_FAILURE = 516;
    const short LINK_FAILURE = 517;
    const short LOSS_OF_SYNCHRONISATION = 518;
    const short LOST_REDUNDANCY = 519;
    const short MAINS_BREAKDOWN_WITH_BATTERY_BACKUP = 520;
    const short MAINS_BREAKDOWN_WITHOUT_BATTERY_BACKUP = 521;
    const short POWER_SUPPLY_FAILURE = 522;
    const short RECEIVER_ANTENNA_FAULT = 523;
    const short RECEIVER_MULTICOUPLER_FAILURE = 525;
    const short REDUCED_TRANSMITTER_OUTPUT_POWER = 526;
    const short SIGNAL_QUALITY_EVALUATION_FAULT = 527;
    const short TIMESLOT_HARDWARE_FAILURE = 528;
    const short TRANSCEIVER_PROBLEM = 529;
    const short TRANSCODER_PROBLEM = 530;
    const short TRANSCODER_OR_RATE_ADAPTER_PROBLEM = 531;
    const short TRANSMITTER_ANTENNA_FAILURE = 532;
    const short TRANSMITTER_ANTENNA_NOT_ADJUSTED = 533;
    const short TRANSMITTER_LOW_VOLTAGE_OR_CURRENT = 535;
    const short TRANSMITTER_OFF_FREQUENCY = 536;
    const short DATABASE_INCONSISTENCY = 537;
    const short FILE_SYSTEM_CALL_UNSUCCESSFUL = 538;
    const short INPUT_PARAMETER_OUT_OF_RANGE = 539;
    const short INVALID_PARAMETER = 540;
    const short INVALID_POINTER = 541;
    const short MESSAGE_NOT_EXPECTED = 542;
    const short MESSAGE_NOT_INITIALISED = 543;
    const short MESSAGE_OUT_OF_SEQUENCE = 544;
    const short SYSTEM_CALL_UNSUCCESSFUL = 545;
    const short TIMEOUT_EXPIRED = 546;
    const short VARIABLE_OUT_OF_RANGE = 547;
    const short WATCH_DOG_TIMER_EXPIRED = 548;
    const short COOLING_SYSTEM_FAILURE = 549;
    const short EXTERNAL_EQUIPMENT_FAILURE = 550;
    const short EXTERNAL_POWER_SUPPLY_FAILURE = 551;
    const short EXTERNAL_TRANSMISSION_DEVICE_FAILURE = 552;
    const short REDUCED_ALARM_REPORTING = 561;
    const short REDUCED_EVENT_REPORTING = 562;
    const short RECUCED_LOGGING_CAPABILITY = 563;
    const short SYSTEM_RESOURCES_OVERLOAD = 564;
    const short BROADCAST_CHANNEL_FAILURE = 565;
    const short CALL_ESTABLISHMENT_ERROR = 566;
    const short INVALID_MESSAGE_RECEIVED = 567;
    const short INVALID_MSU_RECEIVED = 568;
    const short LAPD_LINK_PROTOCOL_FAILURE = 569;
    const short LOCAL_ALARM_INDICATION = 570;
    const short REMOTE_ALARM_INDICATION = 571;
    const short ROUTING_FAILURE = 572;
    const short SS7_PROTOCOL_FAILURE = 573;
    const short TRANSMISSION_FAILURE = 574;
};
```

```
/*
This block identifies the acknowledgement state of a reported alarm.
*/
interface AckState
{
   const short ACKNOWLEDGED = 1;
   const short UNACKNOWLEDGED = 2;
};

/*
This block identifies attributes which are included as part of the Alarm IRP
These attribute values should not clash with those defined for the attributes
of notification header (see IDL of Notification IRP).
*/
interface AttributeNameValue
{
   const string ALARM_ID = "f";
   const string PROBABLE_CAUSE = "g";
   const string PERCEIVED_SEVERITY = "h";
   const string SPECIFIC_PROBLEM = "i";
   const string ADDITIONAL_TEXT = "j";
   const string ACK_TIME = "k";
   const string ACK_USER_ID = "l";
   const string ACK_SYSTEM_ID = "m";
   const string ACK_STATE = "n";
   const string COMMENTS = "o";
   const string BACKED_UP_STATUS = "p";
   const string BACK_UP_OBJECT = "q";
   const string THRESHOLD_INFO = "r";
   const string TREND_INDICATION = "s";
   const string STATE_CHANGE_DEFINITION = "t";
   const string MONITORED_ATTRIBUTES = "u";
   const string PROPOSED_REPAIR_ACTIONS = "v";
   const string CORRELATED_NOTIFICATIONS = "w";
   const string REASON = "x";
};

/*
Defines the content of a Comment
*/
struct Comment
{
   ManagedGenericIRPConstDefs::IRPTime comment_time;
   string comment_text;
   string user_id;
   string system_id;
};

/*
Defines a set of comments which are placed in the COMMENTS attribute
of a structured event.
*/
typedef sequence <Comment> CommentSet;

/*
It indicates if an object has a back up.
True implies backed up.  False implies not backed up.
*/
typedef boolean BackedUpStatusType;

/*
It indicates if the threshold crossed was in the up or down direction.
*/
enum ThresholdIndicationType {Up, Down};
```

```
   /*
   It indicates if some observed condition is getting better, worse,
   or not changing.
   */
   enum TrendIndicationType {LessSevere, NoChange, MoreSevere};

   /*
   It is used to report a changed attribute value.
   */
   struct AttributeValueChangeType
   {
      string attribute_name;
      any    old_value;  // type depends on attribute
      any    new_value;  // type depends on attribute
   };

   typedef sequence <AttributeValueChangeType> AttributeChangeSetType;

   /*
   It is used to report an attribute and its value.
   */
   struct AttributeValueType
   {
      string attribute_name;
      any    value;  // type depends on the attribute
   };

   typedef sequence <AttributeValueType> AttributeSetType;

   typedef sequence <long> NotifIdSetType;

   /*
   This holds identifiers of notifications that are correlated.
   */
   struct CorelatedNotification
   {
      string source;  // Contains DN of MO that emitted the set of notifications
                      // DN string format in compliance with Name Convention for
                      // Managed Object.
                      // This may be a zero-length string.  In this case, the MO
                      // is identified by the value of the MOI attribute
                      // of the Structured Event, i.e., the notification.
      NotifIdSetType notif_id_set;  // Set of related notification ids
   };

   /*
   Correlated Notification sets are sets of Correlated Notification
   structures.
   */
   typedef sequence <CorelatedNotification> CorrelatedNotificationSetType;

   /*
   Define the structure returned when an operation fails for a set of alarm ids.
   A reason is provided in order to indicate why the operation failed.
   */
   struct BadAlarmInformationIdSeq
   {
      string alarm_information_reference;
      string reason;
   };

   typedef sequence <string> AlarmInformationIdSeq;
   typedef CosNotification::EventBatch AlarmInformationSeq;
};

/* ## Module: AlarmIRPSystem
```

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

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

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

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

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


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

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

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


/*
Request to acknowledge one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal acknowledge_alarms (
   in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
   in string ack_user_id,
   in string ack_system_id,
   out AlarmIRPConstDefs::BadAlarmInformationIdSeq
      bad_alarm_information_id_list
)
raises (AcknowledgeAlarms, ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);


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


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

```
            ManagedGenericIRPSystem::ParameterNotSupported,
            ManagedGenericIRPSystem::InvalidParameter);

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


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

#endif
```

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2000	S_07	SP-000012	-		Approved at TSG SA #7 and placed under Change Control	2.0.0	3.0.0
Mar 2000		-	-		cosmetic	3.0.0	3.0.1
Jun 2000	S_08	SP-000253	005		Split of TS - Part 3: Alarm Integration Reference Point (IRP): CORBA Solution Set (SS)	3.0.1	3.1.0
Sep 2000	S_09	SP-000439	003		Correct push_structured_event of push_structured_events	3.1.0	3.2.0
Sep 2000	S_09	SP-000439	004		Remove the use of interface to encapsulate const strings	3.1.0	3.2.0
Dec 2000	S_10	SP-000521	001	1	Allow “Structured Event Filterable Body Fields” to be absent if parameters are not used	3.2.0	3.3.0
Dec 2000	S_10	SP-000521	002	1	Specific behaviour of the Iterator	3.2.0	3.3.0
Dec 2000	S_10	SP-000521	005		Inconsistent qualifiers	3.2.0	3.3.0
Mar 2001	S_11	SP-010032	006		Missing how "Notify Alarm List Rebuilt" reason attribute is located in Structured Event	3.3.0	3.4.0
Mar 2001	S_11	SP-010032	007		Use alarmInformationBody in additionalInformation.ackTime	3.3.0	3.4.0
Jun 2001	S_12	SP-010239	008		Probable Cause “Intrusion Detection” is missing	3.4.0	3.5.0
Jun 2001	S_12		009		Alarm IRP: CORBA SS Rel4 - Addition of feature	3.5.1	4.0.0

3GPP TSG-SA5 (Telecom Management) **S5-010339**
Meeting #20, Brighton, UK, 28 May - 1 June 2001

CR-Form-v3

CHANGE REQUEST

⌘ **32.111-4** **CR** **001** ⌘ rev **-** ⌘ Current version: **3.1.1** ⌘

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

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

Title:	⌘	Alarm IRP: CMIP SS Rel4 - Addition of feature
Source:	⌘	SA5
Work item code:	⌘	OAM-FM
Date:	⌘	08/062001
Category:	⌘	**B**
Release:	⌘	Rel4

Use one of the following categories:
F *(essential correction)*
A *(corresponds to a correction in an earlier release)*
B *(Addition of feature),*
C *(Functional modification of feature)*
D *(Editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	Reflect changes to Alarm IRP: IS Rel4 (32.111-2).
Summary of change:	⌘	
Consequences if not approved:	⌘	There will be no CMIP SS that corresponds to the Alarm IRP: IS Rel4.

Clauses affected:	⌘	All clauses.
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	This CR is dependent on the approval of CR32.111-1-003_S5-010343 and CR32.111-2-008_S5-010344. As SA5 had not reviewed the implementation of this CR, the attached revised TS 32.111-4 is submitted to SA#12 for Information only.

3GPP TS 32.111-4 V3.1.2 (2001-06)

Technical Specification

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

Keywords

Fault Management, Alarms

3GPP

Postal address

3GPP support office address

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

Internet

http://www.3gpp.org

Contents

Foreword

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

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

Part 1: "3G Fault Management Requirements";

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

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

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

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

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

1 Scope

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

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

2 References

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

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

[1] 3G TS 32.301-2: "Notification Integration Reference Point: Information Service".

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

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

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

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

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

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

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

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

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

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

3 Definitions and abbreviations

3.1 Definitions

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

3.2 Abbreviations

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

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

4 Basic aspects

The present document provides all the GDMO and ASN.1 definitions necessary to implement the Alarm IRP Information Service [?] for the CMIP interface. The Alarm IRP Information Service description is based on Information Object Classes (IOC), Relationships among IOC and Interfaces (used or implemented by IOC) which include Operations and/or Notifications.

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

Regarding the Notifications, the present document is based on the Notification IRP CMIP Solution Set (3G TS 32.301-4 [10]).

4.3 Reporting cleared alarms

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

4.4 Acknowledgment of alarms

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

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

– One Action to acknowledge alarms;

– One Action to unacknowledge alarms;

– ITU-T X.721 [4] compliant Alarm Notification to inform Managers about changes of acknowledgment state.

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

4.5 Management of comments associated to alarms

This feature provides the Operators with the capability to add comments to an alarm and to share such information among all the OS (EM and NM) that are involved in the network management. An OS shall have the capability to record more than one comment for each alarm.
The management of the comments associated to alarms is similar to the management of the acknowledgment of alarms and is achieved by means of the same MOC `alarmControl`. For the management of the comments, the MOC `alarmControl` includes one Action to set the comment and Notification to distribute the comments to other OS.

4.6 Alignment of alarm conditions over the Itf-N

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

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

– The Manager sends to the Agent a *getAlarmList* request containing the following information:

- *alarmAckState*, used to select the alarms from the Agent's alarm list for the current alignment (e.g. all active alarms).
- *destination*, identifying the destination to which event reports that have passed the filter conditions are sent.
- *filter*, this optional parameter defines the conditions an alarm notification shall fulfil in order to be forwarded to the Manager. It applies only for the current alignment request.

– After evaluation of the request, the Agent first generates an *alignmentId* value, which unambiguously identifies this alignment process. This value is used by the Manager to correlate alarm reports to the corresponding alignment requests, in case this Manager issues several alarm alignments in parallel.

– The Agent creates a temporary Event Forwarding Discriminator (EFD) instance for the purpose of this alarm alignment, using the parameters *destination* and *filter* received in the request. If the *filter* parameter is absent or NULL, all alarm notifications are forwarded to the Manager through this EFD, according to the value of the parameter *alarmAckState*.
The filter is set by the Agent automatically in order to forward to only those alarm notifications containing, at the beginning of the field *additionalText*, either the string "(ALIGNMENT-<alignmentId>)“ or the string „(ALIGNMENTEND-<alignmentId>)“.

– The Agent sends back a *getAlarmList* response, which contains the *alignmentId* described above and the *status* information, indicating the result of the request. (see the message flow in Figure 1).

– The Agent scans now its alarm list. For every alarm, which matches the criteria defined by the *alarmAckState* parameter, the Agent inserts, at the beginning of the field *additionalText*, the string „(ALIGNMENT-<alignmentId>)“. According to ITU-T Recommendation X.734 [6], the Agent forwards these alarm notifications towards all EFDs.
In the last alarm of the list the Agent inserts the string „(ALIGNMENTEND-<alignmentId>)“ to indicate the end of the alarm alignment.

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

a) By all the EFD instances used for „real-time“ alarm reporting, due to the presence of the sub-string „ALIGNMENT“ in the field *additionalText* (see 3G TS 32.301-4 [10]).

b) By all temporary EFD instances possibly created for parallel alignments, due to the presence of the unambiguous sub-string „<alignmentId>“ in the *additionalText* field.

– After sending the last alarm report (identified by the sub-string „ALIGNMENTEND“ in the *additionalText),* the Agent automatically deletes the temporary EFD instance (see Figure 1).

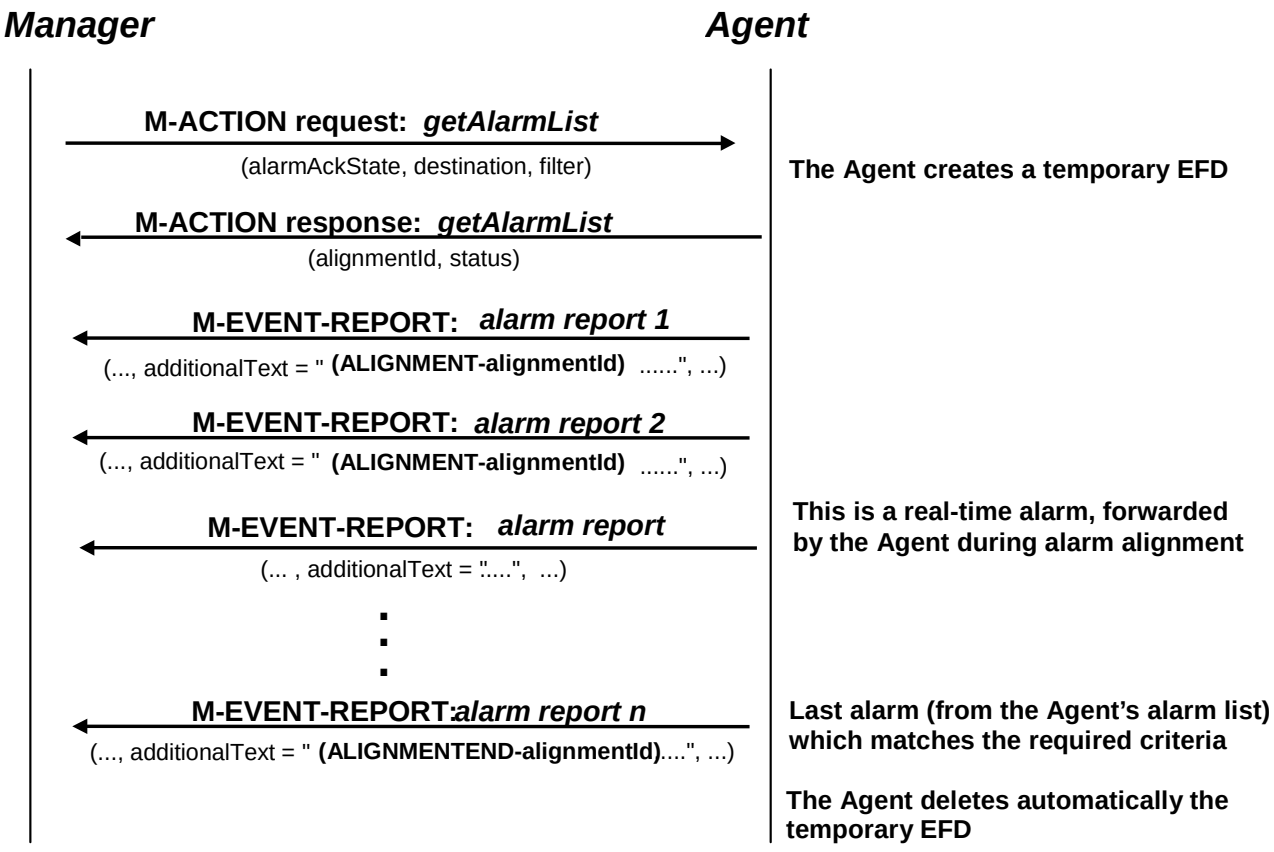


Figure 1: Alignment arrow diagram

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

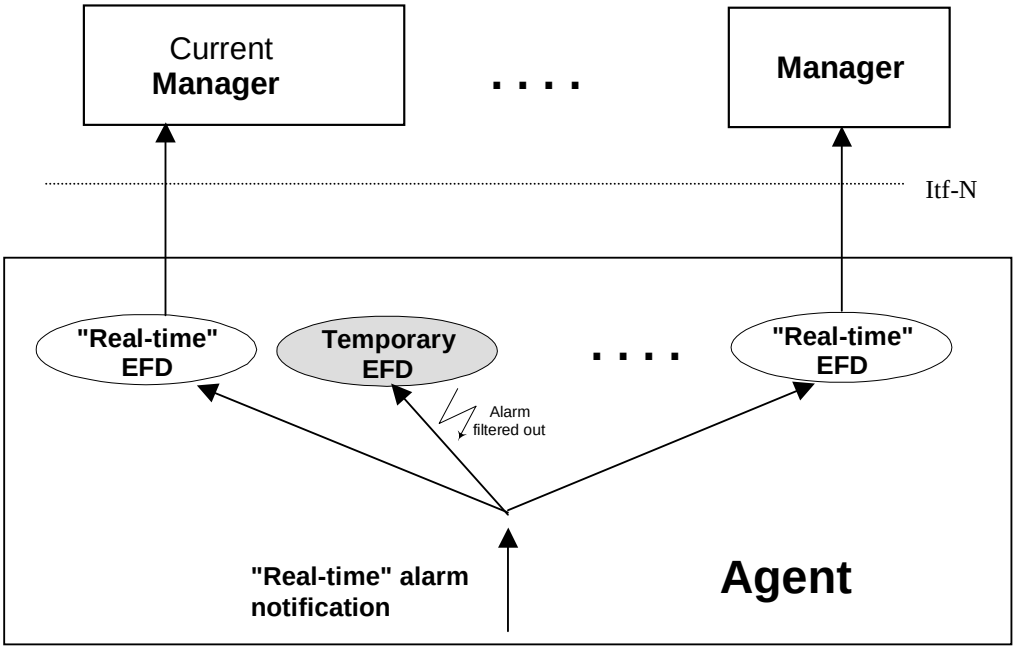


Figure 2: Treatment of “real time” alarms

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

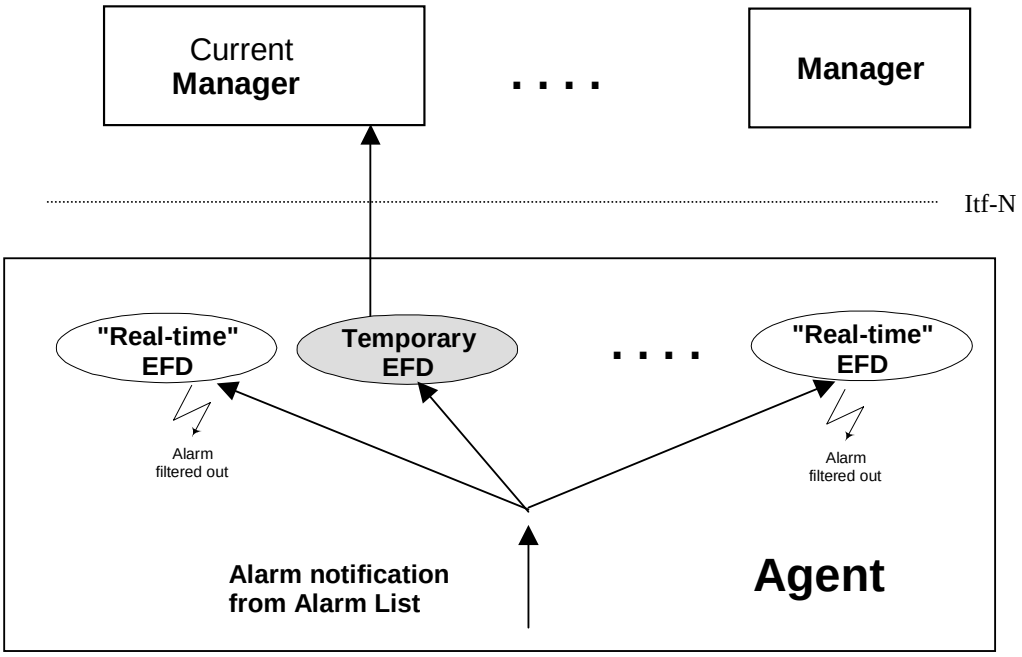


Figure 3: Treatment of “alignent” alarms

4.7 Mapping

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

4.7.1 Mapping of IOC and Interfaces

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

4.7.2 Mapping of Interface/Operations

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

Table 1: Mapping of Operations

Interface/Operations of the Alarm IRP Information Services	GDMO Actions of CMIP Solution Set	Qualifier
AlarmIRPOperations_1/acknowledgeAlarms	acknowledgeAlarms	M
AlarmIRPOperations_1/getAlarmList	getAlarmList	M

AlarmIRPOperations_2/getAlarmCount	getAlarmCount	O
AlarmIRPOperations_3/unacknowledgeAlarms	unacknowledgeAlarms	O
AlarmIRPOperations_4/setComment	setComment	O
GenericIRPVersionOperation/getIRPVersion	getAlarmIRPVersion	M
GenericIRPProfileOperation/getNotificationProfile	getNotificationProfile	O
GenericIRPProfileOperation/getOperationProfile	getOperationProfile	O

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

4.7.3 Mapping of Parameters of each operation

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

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

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

Table 2: Mapping of parameters of ‘acknowledgementAlarms’

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

Table 3: Mapping of Parameters of ‘getAlarmCount’

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
criticalCount	OUT	criticalCount	M
majorCount	OUT	majorCount	M
minorCount	OUT	minorCount	M
warningCount	OUT	warningCount	M
indeterminateCount	OUT	indeterminateCount	M
clearedCount	OUT	clearedCount	M
status	OUT	status	M

Table 4: Mapping of Parameters of ‘getAlarmList’

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O

alarmAckState	IN	alarmAckState	O
--		destination (input) - see NOTE 1	M
alarmInformationList	OUT	(sequence of alarm notifications) (see Clause 4.5)	M
status	OUT	status	M
--		alignmentId (output) - see NOTE 2	M

NOTE 1: destination is a CMIP specific parameter and is determined by the Manager.
NOTE 2: alignmentId is a CMIP specific parameter and is determined by the Agent

Table 5: Mapping of Parameters of 'getAlarmIRPVersion'

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

Table 4: Mapping of Parameters of 'getOperationProfile'

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

Table 4: Mapping of Parameters of 'getNotificatioProfile'

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

Table 4: Mapping of Parameters of 'setComment'

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

Table 6: Mapping of Parameters of 'unacknowledgeAlarms'

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

ackUserId	IN	ackUserId	M
ackSystemId	IN	ackSystemId	O
badAlarmInformationReferenceList	OUT	errorAlarmReferenceList	M
status	OUT	status	M

4.7.4 Mapping of Notifications

Table 7 maps the Notifications defined in the Information Service of the Alarm IRP to the equivalent Notifications of the CMIP solution set for the Alarm IRP. The CMIP Notifications are qualified as Mandatory (M) or Optional (O).

Table 7: Mapping of Notifications

Notifications of Information Services of the Alarm IRP	Equivalent Notifications of the CMIP solution set for the Alarm IRP	Qualifier
notifyNewAlarm	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationAlarm ITU-T X.721 [4]	M
notifyChangedAlarm	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationAlarm ITU-T X.721 [4]	O
notifyClearedAlarm	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationAlarm ITU-T X.721 [4]	M
notifyAckStateChanged	environmentalAlarm ITU-T X.721 [4] equipmentAlarm ITU-T X.721 [4] qualityofServiceAlarm ITU-T X.721 [4] processingErrorAlarm ITU-T X.721 [4] communicationAlarm ITU-T X.721 [4]	M
notifyAlarmListRebuilt	alarmListRebuilt	M
notifyComments	notifyComments	O

4.7.5 Mapping of Parameters of each notification

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

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

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

Notification parameters of Information Services	CMIP equivalences	Qualifier
--	notificationIdentifier (Note 1)	M
probableCause	probableCause	M
specificProblems	specificProblems	O
perceivedSeverity	perceivedSeverity	M
backedUpStatus	backedUpStatus	O
backUpObject	backUpObject	O
trendIndication	trendIndication	O
thresholdInfo	thresholdInfo	O
correlatedNotifications	correlatedNotifications	O
stateChangeDefinition	stateChangeDefinition	O
monitoredAttributes	monitoredAttributes	O
proposedRepairActions	proposedRepairActions	O
additionalText	additionalText	O
additionalInformation	additionalInformation	(Note 2)
NOTE 1: notificationIdentifier is a parameter of the Notification Header defined in 3G TS 32.301-2 [1]. NOTE 2: See qualification information in 3G TS 32.111-2 [9], Table 13: Parameter-Attributes of `alarmInformationBody`.		

Table 9: Mapping of Parameters of 'notifyAlarmListRebuilt'

Notification parameters of Information Services	CMIP equivalents	Qualifier
	notificationIdentifier (see Note)	
reason	reason	M
NOTE: notificationIdentifier is a parameter of the Notification Header defined in 3G TS 32.301-2 [1].		

Table 9: Mapping of Parameters of 'notifyComments'

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

5 GDMO definitions

5.1 Managed Object Classes

5.1.1 alarmControl

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

```
alarmControl MANAGED OBJECT CLASS
    DERIVED FROM
        ”Rec. X.721 | ISO/IEC 10165-2 : 1992”:top;
    CHARACTERIZED BY
        alarmControlBasicPackage,
        alarmAcknowledgementPackage,
        alarmCommentPackage,
        alarmIRPVersionPackage,
        alarmProfilePackage;
REGISTERED AS { ts32-111AlarmObjectClass 1};
```

5.2 Packages

5.2.1 alarmControlBasicPackage

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

alarmControlBasicPackageBehaviour BEHAVIOUR
DEFINED AS
```

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

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

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

The action 'getAlarmCount' is the means, for the Manager, to ask the number of currently available alarms in the Agent according to the specification in the action request.

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

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

5.2.2 alarmAcknowledgementPackage

alarmAcknowledgementPackage **PACKAGE**
BEHAVIOUR
alarmAcknowledgementPackageBehaviour;
ACTIONS
acknowledgeAlarms,
unacknowledgeAlarms;
NOTIFICATIONS
"Rec. X.721 | ISO/IEC 10165-2 : 1992":communicationsAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":environmentalAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":equipmentAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":processingErrorAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":qualityofServiceAlarm;
REGISTERED AS { ts32-111AlarmPackage 2};

alarmAcknowledgementPackageBehaviour **BEHAVIOUR**
DEFINED AS

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

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

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

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

5.2.3 alarmCommentPackage

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

alarmCommentPackageBehaviour **BEHAVIOUR**
DEFINED AS

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

5.2.4 alarmIRPVersionPackage

alarmIRPVersionPackage **PACKAGE**
BEHAVIOUR
alarmIRPVersionPackageBehaviour;
ATTRIBUTES
supportedAlarmIRPVersions GET;
ACTIONS
getAlarmIRPVersion;
REGISTERED AS { ts32-111AlarmPackage 4};

alarmIRPVersionPackageBehaviour **BEHAVIOUR**
DEFINED AS

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

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

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

5.2.5 alarmProfilePackage

alarmProfilePackage PACKAGE
BEHAVIOUR
alarmProfilePackageBehaviour;
ACTIONS
getOperationProfile,
getNotificationProfile;

REGISTERED AS { ts32-111AlarmPackage 5};

alarmProfilePackageBehaviour BEHAVIOUR

DEFINED AS

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

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

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

5.3 Actions

5.3.1 acknowledgeAlarms (M)

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

acknowledgeAlarmsBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains the following data:

- *alarmReferenceList*

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

- *ackUserId*

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

- *ackSystemId*

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

The 'Action response' contains the following data:

- *status*

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

- *errorAlarmReferenceList*

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

5.3.2 getAlarmCount (O)

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

getAlarmCountBehaviour **BEHAVIOUR**
DEFINED AS

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

The ‘Action information’ field contains the following data:

- *alarmAckState*

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

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

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

- *filter*

  The handling of this optional parameter is as follows:

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

  The ‘Action response’ is composed of:

- The numbers of alarms for each *perceivedSeverity* value (if applicable).

- The parameter *status* containing the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).";

5.3.3 getAlarmList (M)

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

getAlarmListBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains the following data:

- *alarmAckState*

  Depending on this optional parameter value, the NM gets the alarm reports according to the following possible choices:

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

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

- *destination*

  This parameter identifies the destination to which the alarm reports that have passed the test conditions specified in the parameter 'filter' are sent. According to ITU-T Recommendation X.721 [4], if no destination is specified in the request, then the discriminator is created with the destination defaulted to the AE-Title of the invoker.

- *filter*

  The handling of this optional parameter (valid only for the current alignment request) is as follows:

- if present and not NULL, it indicates a filter constraint which shall apply in the forwarding of the alignment-related alarm reports
- if its value is NULL, no real filter shall be considered and the Manager receives the alarms according to the value of the parameter *alarmAckState* (see above).

The ‘Action response’ contains the following data:

- *alignmentId*

  The parameter is defined by the Agent and identifies unambiguously the current alarm alignment procedure. It allows the Manager to distinguish between alarm reports sent as consequence of several own alignment requests triggered in parallel.

- *status*

  The parameter contains the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).

  After the action response is forwarded to the NM, the Agent sends the alarm list as a sequence of single alarm notifications in accordance with the values of the request parameters. Every alarm notification contains all fields of the alarm stored in the alarm list. In particular:

- The field *additionalText* contains at the beginning a string to allow a Manager to recognise that this alarm report is sent due to a previous *getAlarmList* request. The structure of this string is:

- '(ALIGNMENT-alignmentId)' for every alarm report except the last one **or**

- '(ALIGNMENTEND-alignmentId)' for the last alarm report sent by the Agent due to the current *getAlarmList* request.

- If available, the data related to the acknowledgment history (i.e. ackState, ackTime, ackUserId, ackSystemId) are provided in the field *additionalInformation*.

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

5.3.4 setComment (M)

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

setCommentBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains:

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

- commentUserId
  Contains the identity of the User that invokes this operation.

- commentSystemId
  Contains the identity of the NM that invokes this operation.

- commentText
  Contains the text of the comment.

The ‘Action response’ is composed of the following data:

- errorAlarmReferenceList
  List of pair of `alarmId` and failure reason.

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

5.3.5 getAlarmIRPVersion(M)

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

getAlarmIRPVersionBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains no data.

The ‘Action response’ is composed of the following data:

- *versionNumbersList*

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

- *status*

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

5.3.6 getNotificationProfile (O)

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

getNotificationProfileBehaviour **BEHAVIOUR**

DEFINED AS

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

The 'Action information' contains the following data:

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

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

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

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

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

5.3.7 getOperationProfile (O)

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

getOperationProfileBehaviour **BEHAVIOUR**

DEFINED AS

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

The 'Action information' contains the following data:

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

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

- *operationNameProfile*
  It contains a list of operation names.

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

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

5.3.8 unacknowledgeAlarms (O)

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

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

The 'Action information' field contains the following data:

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

- *ackUserId*

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

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

  The 'Action response' contains the following data:

- *status*

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

- *errorAlarmReferenceList*

  This parameter (significant only if *status* = unackPartlySuccessful) contains the list of MOI (Managed Object Instance) and notificationIdentifier pairs of the alarms which could not be unacknowledged and, for each

alarm, also the reason of the error. MOI can be absent if scope of uniqueness of notificationIdentifier is across the IRPAgent. “;

5.4 Notifications

5.4.1 alarmListRebuilt (M)

alarmListRebuilt **NOTIFICATION**
BEHAVIOUR
alarmListRebuiltBehaviour;
WITH INFORMATION SYNTAX
TS32-111-4TypeModule .AlarmListRebuiltInfo;
REGISTERED AS { ts32-111AlarmNotification 1};

alarmListRebuiltBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Event Information' field contains the following data:

- *notificationIdentifier*

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

- *reason*

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

5.4.2 notifyComments (M)

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

notifyCommentsBehaviour **BEHAVIOUR**
DEFINED AS

”This notification is used by the Agent to inform the NM that a comment has been associated to one or more alarms.

The 'Event Information' field contains the following data:

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

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

";

5.5 Attributes

5.5.1 alarmControlId

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

alarmControlIdBehaviour **BEHAVIOUR**
DEFINED AS
"This attribute names an instance of a 'alarmControl' object class.";

5.5.2 alarmsCountSummary

alarmsCountSummary **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX
TS32-111-4TypeModule .AlarmsCountSummary;
MATCHES FOR
EQUALITY;
BEHAVIOUR
alarmsCountSummaryBehaviour;
REGISTERED AS { ts32-111AlarmAttribute 2};

alarmsCountSummaryBehaviour **BEHAVIOUR**
DEFINED AS

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

5.5.3 supportedAlarmIRPVersions

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

supportedAlarmIRPVersionsBehaviour **BEHAVIOUR**
DEFINED AS

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

5.6 Parameters

5.6.1 ackStateParameter

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

ackStateParameterBehaviour **BEHAVIOUR**
DEFINED AS
”This parameter models the optional *additionalInformation* field of the alarm notification. If present, it informs the NM about the current acknowledgement state of the present alarm.”;

5.6.2 ackSystemIdParameter

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

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

5.6.3 ackTimeParameter

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

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

5.6.4 ackUserIdParameter

ackUserIdParameter **PARAMETER**
CONTEXT
TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
TS32-111-4TypeModule.UserId;
BEHAVIOUR
ackUserIdParameterBehaviour;
REGISTERED AS { ts32-111AlarmParameter 4};

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

6 ASN.1 definitions for Alarm IRP

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

```
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
--EXPORTS everything
IMPORTS

NotificationIdentifier, Destination, EventTime, ProbableCause, PerceivedSeverity
FROM Attribute-ASN1Module {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1}

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

CMISFilter, ObjectInstance, ObjectClass, EventTypeId
FROM CMIP-1 {joint-iso-ccitt ms(9) cmip(1) modules(0) protocol(3)};


baseNodeUMTS  OBJECT IDENTIFIER ::= { itu-t (0) identified-organization (4) etsi (0) mobileDomain (0)
                                      umts-Operation-Maintenance (3) }

ts32-111Prefix          OBJECT IDENTIFIER ::= { baseNodeUMTS ts-32-111(111)}
ts32-111Part4           OBJECT IDENTIFIER ::= { ts32-111Prefix part4(4)}
ts32-111-4InfoModel     OBJECT IDENTIFIER ::= { ts32-111Part4 informationModel(0)}

ts32-111AlarmObjectClass      OBJECT IDENTIFIER ::= { ts32-111-4InfoModel managedObjectClass(3)}
ts32-111AlarmPackage          OBJECT IDENTIFIER ::= { ts32-111-4InfoModel package(4)}
ts32-111AlarmParameter        OBJECT IDENTIFIER ::= { ts32-111-4InfoModel parameter(5)}
ts32-111AlarmAttribute        OBJECT IDENTIFIER ::= { ts32-111-4InfoModel attribute(7)}
ts32-111AlarmAction           OBJECT IDENTIFIER ::= { ts32-111-4InfoModel action(9)}
ts32-111AlarmNotification     OBJECT IDENTIFIER ::= { ts32-111-4InfoModel action(10)}



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

```
    errorAlarmReferenceList         AckErrorList
    }
AckState ::= ENUMERATED
    {
    acknowledged        (0),
    unacknowledged (1)
    }
AckTime ::= GeneralizedTime
AlarmChoice ::= ENUMERATED
    {
    allAlarms                   (0),
    allActiveAlarms             (1),
    allActiveAndAckAlarms       (2),
    allActiveAndUnackAlarms     (3),
    allClearedAndUnackAlarms    (4)
    }
AlarmsCountSummary ::= SEQUENCE
    {
    activeAlarmsCount       INTEGER,    -- this is the sum of criticalCount, majorCount, minorCount, warningCount
                                        -- and indeterminateCount
    criticalCount           INTEGER,
    majorCount              INTEGER,
    minorCount              INTEGER,
    warningCount            INTEGER,
    indeterminateCount      INTEGER,
    clearedCount            INTEGER
    }
AlarmListRebuiltInfo ::= SEQUENCE
    {   notificationIdentifier NotificationIdentifier,    -- ITU-T X.721
    reason              ErrorCauses
    }
Comment ::= GraphicString
ErrorCauses ::= ENUMERATED
    {
    noError (0),                    -- operation / notification successfully performed
    wrongFilter (1),                -- the value of the filter parameter is not valid
    wrongAlarmAckState (2),         -- the value of the alarmAckState parameter (e.g. getAlarmCount) is not valid
    ackPartlySuccessful (3),        -- acknowledgment request partly successful
    unackPartlySuccessful (4),      -- unacknowledgment request partly successful
    wrongAlarmReference (5),        -- alarm identifier used in the alarm reference list not found (e.g. in case of
                                       acknowledgement request)
    wrongAlarmReferenceList (6),    -- the alarm reference list (e.g. in case of acknowledgement request) is empty or
                                       completely wrong
    alarmAlreadyAck (7),            -- alarm to be acknowledged is already in this state
    alarmAlreadyUnack (8),          -- alarm to be acknowledged is already in this state
    wrongUserId (9),                -- the user identifier in the unacknowledgement operation is not the same as in
                                       the previous acknowledgementAlarms request
    wrongSystemId (10),             -- the system identifier in the unacknowledgement operation is not the same as in
                                       the previous acknowledgementAlarms request
    alarmAckNotAllowed (11),        -- current management system not allowed to acknowledge the alarm (e.g. due to
                                       acknowledgement competence rules)
    setCommentPartlySuccessful (12), -- the setComment action partly successful (e.g. some alarmId are not in the
                                       alarmList)
    unspecifiedErrorReason (255)    -- operation failed, specific error unknown
    }
ErrorInfo ::= SEQUENCE
```

```
    {
    moi ObjectInstance OPTIONAL, -- absent if uniqueness of notificationIdentifier is across IRPAgent
    notificationIdentifier          NotificationIdentifier, -- ITU-T X.721
    reason          ErrorCauses
    }
GeneralObjectId ::= INTEGER
GetAlarmCount ::= SEQUENCE
    {
    alarmAckState       AlarmChoice OPTIONAL,
    filter              CMISFilter OPTIONAL-- ITU-T X.711
    }
GetAlarmCountReply ::= SEQUENCE
    {
    criticalCount           INTEGER,
    majorCount              INTEGER,
    minorCount              INTEGER,
    warningCount            INTEGER,
    indeterminateCount      INTEGER,
    clearedCount            INTEGER,
    status                  ErrorCauses
    }
GetAlarmIRPVersionReply ::= SEQUENCE
    {
    versionNumberList   SupportedAlarmIRPVersions,
    status              ErrorCauses
    }
GetAlarmList ::= SEQUENCE
    {
    alarmAckState       AlarmChoice OPTIONAL,
    destination         Destination,                -- ITU-T X.721
    filter              CMISFilter OPTIONAL-- ITU-T X.711
    }
GetAlarmListReply ::= SEQUENCE
    {
    alignmentId     INTEGER,
    status          ErrorCauses
    }
GetNotificationProfileReply ::= SEQUENCE
    {
    notificationNameProfile         NotificationList,
    notificationParameterProfile    ParameterListOfList,
    status                          ErrorCauses
    }

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

IRPVersionNumber ::= GraphicString

NotificationList ::= SET OF NotificationName

NotificationName ::= GraphicString
```

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

OperationList ::= SET OF OperationName

OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

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

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

END -- of module TS32-111-4TypeModule

Annex A (informative): Change history

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

3GPP TS 32.111-4 V3.1.2 (2001-06)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Telecommunication Management;
Fault Management;
Part 4: Alarm Integration Reference Point:
CMIP Solution Set ~~Version 1:1~~
(Release ~~19994~~)

Keywords

Fault Management, Alarms

3GPP

Postal address

3GPP support office address

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

Internet

http://www.3gpp.org

Contents

Foreword

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

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

Part 1: "3G Fault Management Requirements";

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

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

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

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

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

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

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

1 Scope

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

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

2 References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Definitions and abbreviations

3.1 Definitions

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

3.2 Abbreviations

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

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

4 Basic aspects

The present document provides all the GDMO and ASN.1 definitions necessary to implement the Alarm IRP Information Service [?] for the CMIP interface. The Alarm IRP Information Service description is based on Information Object Classes (IOC), Relationships among IOC and Interfaces (used or implemented by IOC) which include Operations and/or Notifications.

In the present document, for the CMIP interfaces the IOC are modeled as GDMO “Managed Object Classes” (MOC) defined specifically for alarm management, the Operations are modeled as GDMO “Actions” of a MOC ~~defined specifically for alarm management~~ while the Notifications are modeled as GDMO “Notifications” included in MOCs that need to report events to the Manager. In more detail, the Notifications related to alarm management are included in a MOC defined in the

present document while the Notifications defined for alarm reporting are not included in any MOC defined in the present document. They will be included in other MOCs defined in other CMIP Solution Set or in other CMIP Information Models.

Regarding the Notifications, the present document is based on the Notification IRP CMIP Solution Set (3G TS 32.301~~106~~-4 ~~[14]~~[10]).

4.3 Reporting cleared alarms

On the CMIP interfaces the clearing of alarms is reported by the Agent to the Managers in accordance with the mechanisms defined in ITU-T Recommendation X.733 ~~[8]~~[5] and ITU-T Recommendation Q.821 ~~[10]~~[7].

4.4 Acknowledgment of alarms

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

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

– One Action to acknowledge alarms;

– One Action to unacknowledge alarms;

– ITU-T ~~Recommendation~~ X.721 ~~[6]~~[4] compliant Alarm Notification to inform Managers about changes of acknowledgment state.

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

4.5 Management of comments associated to alarms

This feature provides the Operators with the capability to add comments to an alarm and to share such information among all the OS (EM and NM) that are involved in the network management. An OS shall have the capability to record more than one comment for each alarm.
The management of the comments associated to alarms is similar to the management of the acknowledgment of alarms and is achieved by means of the same MOC `alarmControl`. For the management of the comments, the MOC `alarmControl` includes one Action to set the comment and Notification to distribute the comments to other OS.

4.5~~5~~6 Alignment of alarm conditions over the Itf-N

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

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

– The Manager sends to the Agent a *getAlarmList* request containing the following information:

- *alarmAckState*, used to select the alarms from the Agent's alarm list for the current alignment (e.g. all active alarms).

- *destination*, identifying the destination to which event reports that have passed the filter conditions are sent.

- *filter*, this optional parameter defines the conditions an alarm notification shall fulfil in order to be forwarded to the Manager. It applies only for the current alignment request.

– After evaluation of the request, the Agent first generates an *alignmentId* value, which unambiguously identifies this alignment process. This value is used by the Manager to correlate alarm reports to the corresponding alignment requests, in case this Manager issues several alarm alignments in parallel.

  – The Agent creates a temporary Event Forwarding Discriminator (EFD) instance for the purpose of this alarm alignment, using the parameters *destination* and *filter* received in the request. If the *filter* parameter is absent or NULL, all alarm notifications are forwarded to the Manager through this EFD, according to the value of the parameter *alarmAckState*.
  The filter is set by the Agent automatically in order to forward to only those alarm notifications containing, at the beginning of the field *additionalText*, either the string "(ALIGNMENT-<alignmentId>)“ or the string „(ALIGNMENTEND-<alignmentId>)“.

– The Agent sends back a *getAlarmList* response, which contains the *alignmentId* described above and the *status* information, indicating the result of the request. (see the message flow in Figure 1).

– The Agent scans now its alarm list. For every alarm, which matches the criteria defined by the *alarmAckState* parameter, the Agent inserts, at the beginning of the field *additionalText*, the string „(ALIGNMENT-<alignmentId>)“. According to ITU-T Recommendation X.734 ~~[9]~~[6], the Agent forwards these alarm notifications towards all EFDs. In the last alarm of the list the Agent inserts the string „(ALIGNMENTEND-<alignmentId>)“ to indicate the end of the alarm alignment.

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

a) By all the EFD instances used for „real-time“ alarm reporting, due to the presence of the sub-string „ALIGNMENT“ in the field *additionalText* (see 3G TS 32.301~~106~~-4 ~~[14]~~[10]).

b) By all temporary EFD instances possibly created for parallel alignments, due to the presence of the unambiguous sub-string „<alignmentId>“ in the *additionalText* field.

– After sending the last alarm report (identified by the sub-string „ALIGNMENTEND“ in the *additionalText)*, the Agent automatically deletes the temporary EFD instance (see Figure 1).

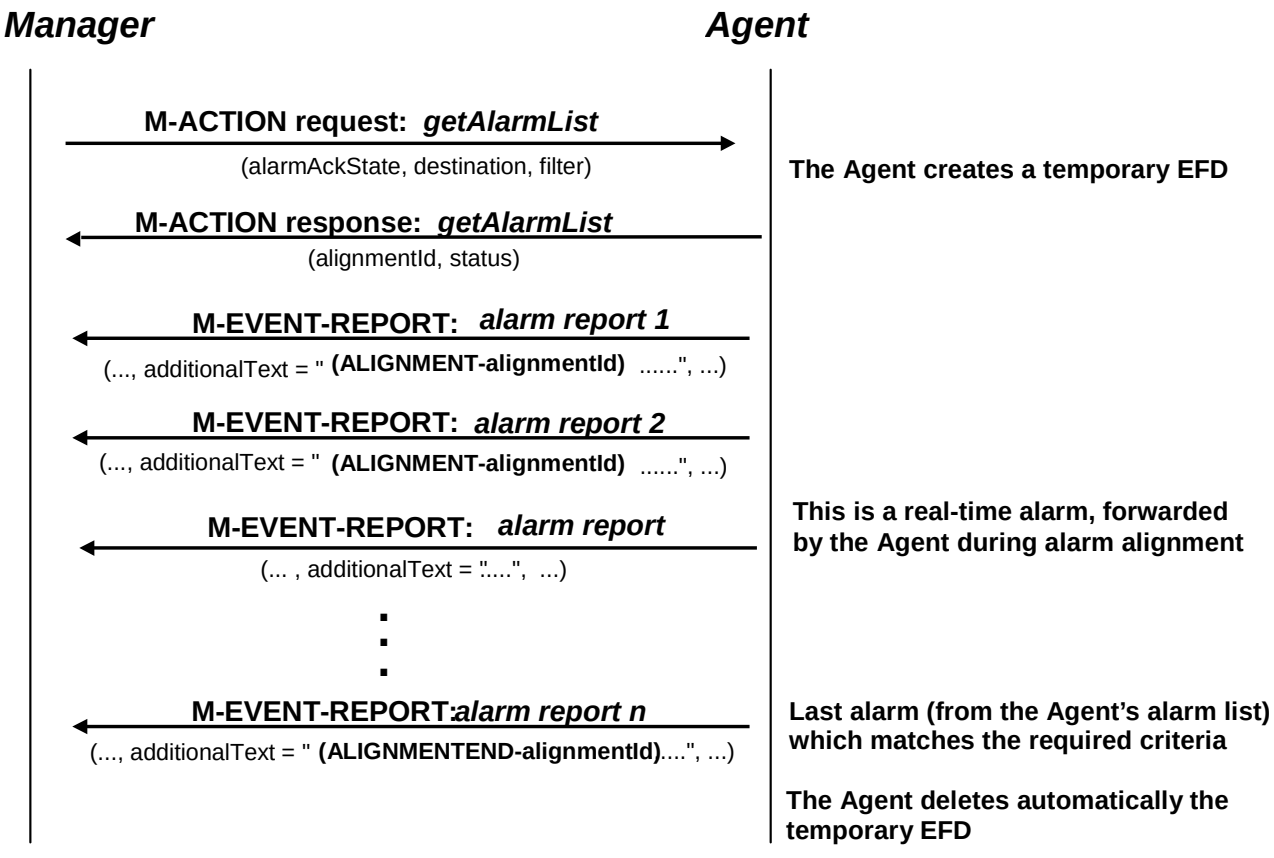


Figure 1: Alignment arrow diagram

Figure 2 shows the handling of a „real-time“ alarm notification (occured during the execution of the *getAlarmList* operation), which is forwarded by the Agent (according to ITU-T Recommendation X.734 ~~[9]~~[6]) to all currently available EFD instances. Dependent on the *discriminatorConstruct* setting of every EFD, such an alarm may or may not reach the related Manager. In any case, this alarm is filtered out by the temporary EFD assigned to the Manager, which triggered the *getAlarmList* request.

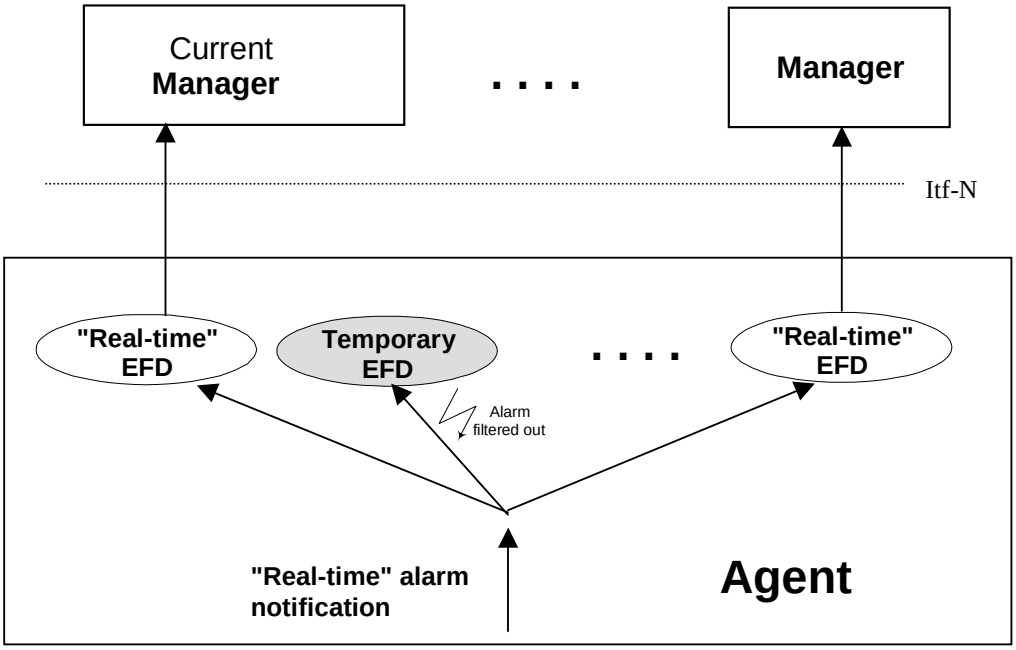


Figure 2: Treatment of “real time” alarms

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

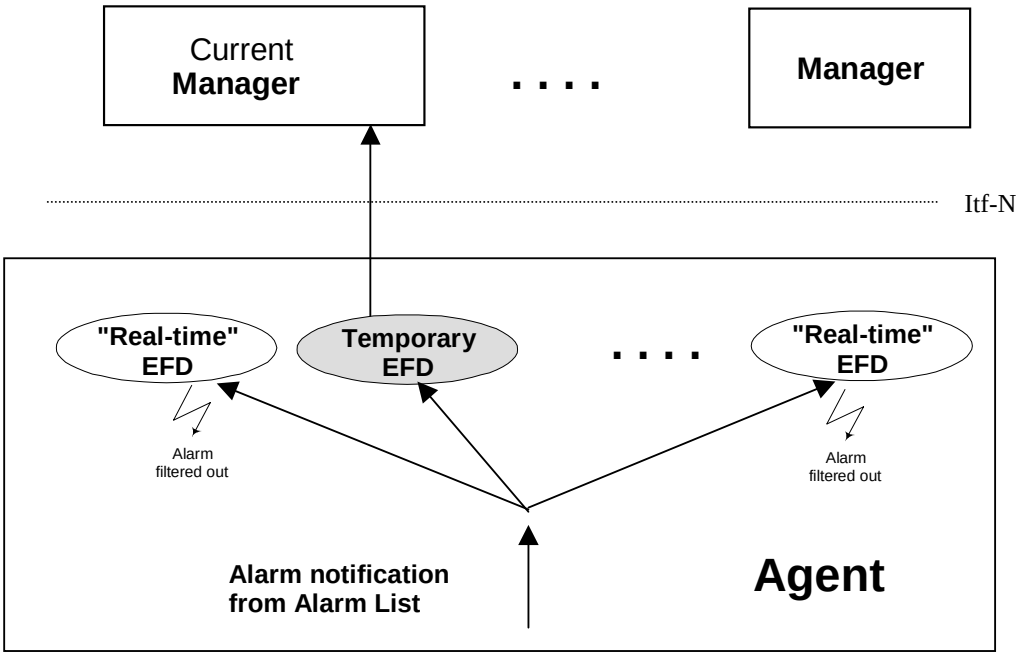


Figure 3: Treatment of “alignent” alarms

4.7~~6~~ Mapping

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

4.7.1 Mapping of IOC and Interfaces

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

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

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

Table 1: Mapping of Operations

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

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

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

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

4.76.32 Mapping of Parameters of each operation

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

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

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

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

Table 2: Mapping of parameters of 'acknowledgementAlarms'

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

Table 3: Mapping of Parameters of 'getAlarmCount'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
criticalCount	OUT	criticalCount	M
majorCount	OUT	majorCount	M
minorCount	OUT	minorCount	M

warningCount	OUT	warningCount	M
indeterminateCount	OUT	indeterminateCount	M
clearedCount	OUT	clearedCount	M
status	OUT	status	M

Table 4: Mapping of Parameters of 'getAlarmList'

Operation parameters of Information Services	IN/OUT	CMIP equivalents	Qualifier
filter	IN	filter	O
alarmAckState	IN	alarmAckState	O
--		destination (input) - see NOTE 1	M
alarmInformationList	OUT	(sequence of alarm notifications) (see Clause 4.5)	M
status	OUT	status	M
--		alignmentId (output) - see NOTE 2	M

NOTE 1: destination is a CMIP specific parameter and is determined by the Manager.
NOTE 2: alignmentId is a CMIP specific parameter and is determined by the Agent

Table 5: Mapping of Parameters of 'getAlarmIRPVersion'

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

Table 4: Mapping of Parameters of 'getOperationProfile'

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

Table 4: Mapping of Parameters of 'getNotificatioProfile'

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

Table 4: Mapping of Parameters of 'setComment'

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

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

Table 6: Mapping of Parameters of 'unacknowledgeAlarms'

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

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

Table 7 maps the Notifications defined in the Information Service of the Alarm IRP to the equivalent Notifications of the CMIP solution set for the Alarm IRP. The CMIP Notifications are qualified as Mandatory (M) or Optional (O).

Table 7: Mapping of Notifications

Notifications of Information Services of the Alarm IRP	Equivalent Notifications of the CMIP solution set for the Alarm IRP	Qualifier
notifyNewAlarm	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	M
notifyChangedAlarm	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	O
notifyClearedAlarm	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	M
notifyAckStateChanged	environmentalAlarm ITU-T X.721 ~~[6]~~[4] equipmentAlarm ITU-T X.721 ~~[6]~~[4] qualityofServiceAlarm ITU-T X.721 ~~[6]~~[4] processingErrorAlarm ITU-T X.721 ~~[6]~~[4] communicationAlarm ITU-T X.721 ~~[6]~~[4]	M
notifyAlarmListRebuilt	alarmListRebuilt	M
notifyComments	notifyComments	O

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

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

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

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

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

Table 9: Mapping of Parameters of 'notifyAalarmListRebuilt'

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

Table 9: Mapping of Parameters of 'notifyComments'

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

5 GDMO definitions

5.1 Managed Object Classes

5.1.1 alarmControl

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

```
alarmControl MANAGED OBJECT CLASS
    DERIVED FROM
        ”Rec. X.721 | ISO/IEC 10165-2 : 1992”:top;
    CHARACTERIZED BY
        alarmControlBasicPackage,
        alarmAcknowledgementPackage,
        alarmCommentPackage,
        alarmIRPVersionPackage,
        alarmProfilePackage;
    REGISTERED AS { ts32-111AlarmObjectClass 1};
```

5.2 Packages

5.2.1 alarmControlBasicPackage

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

alarmControlBasicPackageBehaviour BEHAVIOUR
DEFINED AS
```

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

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

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

The action 'getAlarmCount' is the means, for the Manager, to ask the number of currently available alarms in the Agent according to the specification in the action request.

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

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

5.2.2 alarmAcknowledgementPackage

alarmAcknowledgementPackage **PACKAGE**
BEHAVIOUR
alarmAcknowledgementPackageBehaviour;
ACTIONS
acknowledgeAlarms,
unacknowledgeAlarms;
NOTIFICATIONS
"Rec. X.721 | ISO/IEC 10165-2 : 1992":communicationsAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":environmentalAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":equipmentAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":processingErrorAlarm,
"Rec. X.721 | ISO/IEC 10165-2 : 1992":qualityofServiceAlarm;
REGISTERED AS { ts32-111AlarmPackage 2};

alarmAcknowledgementPackageBehaviour **BEHAVIOUR**
DEFINED AS

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

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

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

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

5.2.3 alarmCommentPackage

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

alarmCommentPackageBehaviour **BEHAVIOUR**
DEFINED AS

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

5.2.43 alarmIRPVersionPackage

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

alarmIRPVersionPackageBehaviour **BEHAVIOUR**
DEFINED AS

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

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

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

5.2.5 alarmProfilePackage

alarmProfilePackage PACKAGE
BEHAVIOUR
alarmProfilePackageBehaviour;
ACTIONS
getOperationProfile,
getNotificationProfile;

REGISTERED AS { ts32-111AlarmPackage 5};

alarmProfilePackageBehaviour BEHAVIOUR

DEFINED AS

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

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

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

5.3 Actions

5.3.1 acknowledgeAlarms (M)

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

acknowledgeAlarmsBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains the following data:

- *alarmReferenceList*

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

- *ackUserId*

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

- *ackSystemId*

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

The 'Action response' contains the following data:

- *status*

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

- *errorAlarmReferenceList*

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

5.3.2 getAlarmCount (O)

getAlarmCount **ACTION**
BEHAVIOUR
getAlarmCountBehaviour;
MODE
CONFIRMED;
WITH INFORMATION SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmCount;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmCountReply;
REGISTERED AS { ts32-111AlarmAction 2};

getAlarmCountBehaviour **BEHAVIOUR**
DEFINED AS

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

The ‘Action information’ field contains the following data:

- *alarmAckState*

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

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

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

- *filter*

  The handling of this optional parameter is as follows:

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

  The ‘Action response’ is composed of:

- The numbers of alarms for each *perceivedSeverity* value (if applicable).

- The parameter *status* containing the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).";

5.3.3 getAlarmList (M)

getAlarmList **ACTION**
BEHAVIOUR
getAlarmListBehaviour;
MODE
CONFIRMED;
WITH INFORMATION SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmList;
WITH REPLY SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GetAlarmListReply;
REGISTERED AS { ts32-111AlarmAction 3};

getAlarmListBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains the following data:

- *alarmAckState*

  Depending on this optional parameter value, the NM gets the alarm reports according to the following possible choices:

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

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

- *destination*

  This parameter identifies the destination to which the alarm reports that have passed the test conditions specified in the parameter 'filter' are sent. According to ITU-T Recommendation X.721 ~~[6]~~[4], if no destination is specified in the request, then the discriminator is created with the destination defaulted to the AE-Title of the invoker.

- *filter*

  The handling of this optional parameter (valid only for the current alignment request) is as follows:

- if present and not NULL, it indicates a filter constraint which shall apply in the forwarding of the alignment-related alarm reports
- if its value is NULL, no real filter shall be considered and the Manager receives the alarms according to the value of the parameter *alarmAckState* (see above).

The ‘Action response’ contains the following data:

- *alignmentId*

  The parameter is defined by the Agent and identifies unambiguously the current alarm alignment procedure. It allows the Manager to distinguish between alarm reports sent as consequence of several own alignment requests triggered in parallel.

- *status*

  The parameter contains the results of the NM action. Possible values: noError (0), error (the value indicates the reason of the error).

  After the action response is forwarded to the NM, the Agent sends the alarm list as a sequence of single alarm notifications in accordance with the values of the request parameters. Every alarm notification contains all fields of the alarm stored in the alarm list. In particular:

- The field *additionalText* contains at the beginning a string to allow a Manager to recognise that this alarm report is sent due to a previous *getAlarmList* request. The structure of this string is:

- '(ALIGNMENT-alignmentId)' for every alarm report except the last one **or**

- '(ALIGNMENTEND-alignmentId)' for the last alarm report sent by the Agent due to the current *getAlarmList* request.

- If available, the data related to the acknowledgment history (i.e. ackState, ackTime, ackUserId, ackSystemId) are provided in the field *additionalInformation*.

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

5.3.4 setComment (M)

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

setCommentBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains:

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

- commentUserId
  Contains the identity of the User that invokes this operation.

- commentSystemId
  Contains the identity of the NM that invokes this operation.

- commentText
  Contains the text of the comment.

The ‘Action response’ is composed of the following data:

- errorAlarmReferenceList
  List of pair of `alarmId` and failure reason.

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

5.3.54 getAlarmIRPVersion (M)

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

getAlarmIRPVersionBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Action information' field contains no data.

The ‘Action response’ is composed of the following data:

- *versionNumbersList*

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

- *status*

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

5.3.6 getNotificationProfile (O)

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

getNotificationProfileBehaviour **BEHAVIOUR**

DEFINED AS

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

The 'Action information' contains the following data:

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

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

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

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

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

5.3.7 getOperationProfile (O)

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

getOperationProfileBehaviour **BEHAVIOUR**

DEFINED AS

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

The 'Action information' contains the following data:

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

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

- *operationNameProfile*
  It contains a list of operation names.

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

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

5.3.~~5~~8 unacknowledgeAlarms (O)

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

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

The 'Action information' field contains the following data:

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

- *ackUserId*

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

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

  The 'Action response' contains the following data:

- *status*

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

- *errorAlarmReferenceList*

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

5.4 Notifications

5.4.1 alarmListRebuilt (M)

alarmListRebuilt **NOTIFICATION**
BEHAVIOUR
alarmListRebuiltBehaviour;
WITH INFORMATION SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmListRebuiltInfo;
REGISTERED AS { ts32-111AlarmNotification 1};

alarmListRebuiltBehaviour **BEHAVIOUR**
DEFINED AS

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

The 'Event Information' field contains the following data:

- *notificationIdentifier*

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

- *reason*

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

5.4.2 notifyComments (M)

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

notifyCommentsBehaviour **BEHAVIOUR**
DEFINED AS

”This notification is used by the Agent to inform the NM that a comment has been associated to one or more alarms.

The 'Event Information' field contains the following data:

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

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

";

5.5 Attributes

5.5.1 alarmControlId

alarmControlId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .GeneralObjectId;
MATCHES FOR
EQUALITY;
BEHAVIOUR
alarmControlIdBehaviour;
REGISTERED AS { ts32-111AlarmAttribute 1};

alarmControlIdBehaviour **BEHAVIOUR**
DEFINED AS
"This attribute names an instance of a 'alarmControl' object class.";

5.5.2 alarmsCountSummary

alarmsCountSummary **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmsCountSummary;
MATCHES FOR
EQUALITY;
BEHAVIOUR
alarmsCountSummaryBehaviour;
REGISTERED AS { ts32-111AlarmAttribute 2};

alarmsCountSummaryBehaviour **BEHAVIOUR**
DEFINED AS

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

5.5.3 supportedAlarmIRPVersions

supportedAlarmIRPVersions **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule . SupportedAlarmIRPVersions;
MATCHES FOR
EQUALITY;

BEHAVIOUR
supportedAlarmIRPVersionsBehaviour;
REGISTERED AS { ts32-111AlarmAttribute 3};

supportedAlarmIRPVersionsBehaviour **BEHAVIOUR**
DEFINED AS
”This attribute provides the information concerning the Alarm IRP versions currently supported by the Agent.”;

5.6 Parameters

5.6.1 ackStateParameter

ackStateParameter **PARAMETER**
CONTEXT
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AckState;
BEHAVIOUR
ackStateParameterBehaviour;
REGISTERED AS { ts32-111AlarmParameter 1};

ackStateParameterBehaviour **BEHAVIOUR**
DEFINED AS
”This parameter models the optional *additionalInformation* field of the alarm notification. If present, it informs the NM about the current acknowledgement state of the present alarm.”;

5.6.2 ackSystemIdParameter

ackSystemIdParameter **PARAMETER**
CONTEXT
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .~~Ack~~SystemId;
BEHAVIOUR
ackSystemIdParameterBehaviour;
REGISTERED AS { ts32-111AlarmParameter 2};

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

5.6.3 ackTimeParameter

ackTimeParameter **PARAMETER**
CONTEXT
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AckTime;
BEHAVIOUR
ackTimeParameterBehaviour;
REGISTERED AS { ts32-111AlarmParameter 3};

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

5.6.4 ackUserIdParameter

ackUserIdParameter **PARAMETER**
CONTEXT
~~TS32-111-AlarmAsn1TypeModule~~TS32-111-4TypeModule .AlarmInfo.additionalInformation;
WITH SYNTAX
TS32-111-4~~AlarmAsn1~~TypeModule.~~Ack~~UserId;
BEHAVIOUR
ackUserIdParameterBehaviour;
REGISTERED AS { ts32-111AlarmParameter 4};

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

6 ASN.1 definitions for Alarm IRP

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

~~TS32-111-AlarmAsn1TypeModule--{ObjectIdentifierValue} to be defined~~
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
--EXPORTS everything
IMPORTS

NotificationIdentifier, Destination, EventTime, ProbableCause, PerceivedSeverity
FROM Attribute-ASN1Module {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1}

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

CMISFilter, ObjectInstance, ObjectClass, EventTypeId
FROM CMIP-1 {joint-iso-ccitt ms(9) cmip(1) modules(0) protocol(3)};

baseNodeUMTS OBJECT IDENTIFIER ::= { itu-t (0) identified-organization (4) etsi (0) mobileDomain (0) umts-Operation-Maintenance (3) }

ts32-111Prefix OBJECT IDENTIFIER ::= { baseNodeUMTS ts-32-111(111)}
ts32-111Part4 OBJECT IDENTIFIER ::= { ts32-111Prefix part4(4)}
ts32-111-4InfoModel OBJECT IDENTIFIER ::= { ts32-111Part4 informationModel(0)}

ts32-111AlarmObjectClass OBJECT IDENTIFIER ::= { ts32-111-4InfoModel managedObjectClass(3)}
ts32-111AlarmPackage OBJECT IDENTIFIER ::= { ts32-111-4InfoModel package(4)}
ts32-111AlarmParameter OBJECT IDENTIFIER ::= { ts32-111-4InfoModel parameter(5)}
ts32-111AlarmAttribute OBJECT IDENTIFIER ::= { ts32-111-4InfoModel attribute(7)}
ts32-111AlarmAction OBJECT IDENTIFIER ::= { ts32-111-4InfoModel action(9)}
ts32-111AlarmNotification OBJECT IDENTIFIER ::= { ts32-111-4InfoModel action(10)}

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

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

```
AckOrUnackAlarms ::= SEQUENCE
    {
    alarmReferenceList  SET OF AlarmReference,   -- ITU-T X.721
    ackUserId           AckUserId,
    ackSystemId         AckSystemId OPTIONAL
    }
AckOrUnackAlarmsReply ::= SEQUENCE
    {
    status              ErrorCauses,
    errorAlarmReferenceList     AckErrorList
    }
AckState ::= ENUMERATED
    {
    acknowledged        (0),
    unacknowledged (1)
    }
AckSystemId ::= GraphicString
AckTime ::= GeneralizedTime
AckUserId ::= GraphicString
AlarmChoice ::= ENUMERATED
    {
    allAlarms                   (0),
    allActiveAlarms             (1),
    allActiveAndAckAlarms       (2),
    allActiveAndUnackAlarms     (3),
    allClearedAndUnackAlarms    (4)
    }
AlarmsCountSummary ::= SEQUENCE
    {
    activeAlarmsCount       INTEGER,    -- this is the sum of criticalCount, majorCount, minorCount, warningCount
                                        -- and indeterminateCount
    criticalCount           INTEGER,
    majorCount              INTEGER,
    minorCount              INTEGER,
    warningCount            INTEGER,
    indeterminateCount      INTEGER,
    clearedCount            INTEGER
    }
AlarmListRebuiltInfo ::= SEQUENCE
    {   notificationIdentifier NotificationIdentifier,    -- ITU-T X.721
    reason              ErrorCauses
    }
Comment ::= GraphicString
ErrorCauses ::= ENUMERATED
    {
    noError (0),                    -- operation / notification successfully performed
    wrongFilter (1),                -- the value of the filter parameter is not valid
    wrongAlarmAckState (2),         -- the value of the alarmAckState parameter (e.g. getAlarmCount) is not valid
    ackPartlySuccessful (3),        -- acknowledgment request partly successful
    unackPartlySuccessful (4),      -- unacknowledgment request partly successful
    wrongAlarmReference (5),        -- alarm identifier used in the alarm reference list not found (e.g. in case of
                                       acknowledgement request)
    wrongAlarmReferenceList (6),    -- the alarm reference list (e.g. in case of acknowledgement request) is empty or
                                       completely wrong
    alarmAlreadyAck (7),            -- alarm to be acknowledged is already in this state
    alarmAlreadyUnack (8),          -- alarm to be acknowledged is already in this state
```

```
    wrongUserId (9),                    -- the user identifier in the unacknowledgement operation is not the same as in
                                           the previous acknowledgementAlarms request
    wrongSystemId (10),                 -- the system identifier in the unacknowledgement operation is not the same as in
                                           the previous acknowledgementAlarms request
    alarmAckNotAllowed (11),            -- current management system not allowed to acknowledge the alarm (e.g. due to
                                           acknowledgement competence rules)
    setCommentPartlySuccessful (12), -- the setComment action partly successful (e.g. some alarmId are not in the
                                           alarmList)
    unspecifiedErrorReason (255)        -- operation failed, specific error unknown
    }
ErrorInfo ::= SEQUENCE
    {
    moi ObjectInstance OPTIONAL, -- absent if uniqueness of notificationIdentifier is across IRPAgent
    notificationIdentifier          NotificationIdentifier, -- ITU-T X.721
    reason          ErrorCauses
    }
GeneralObjectId ::= INTEGER
GetAlarmCount ::= SEQUENCE
    {
    alarmAckState       AlarmChoice OPTIONAL,
    filter              CMISFilter OPTIONAL-- ITU-T X.711
    }
GetAlarmCountReply ::= SEQUENCE
    {
    criticalCount           INTEGER,
    majorCount              INTEGER,
    minorCount              INTEGER,
    warningCount            INTEGER,
    indeterminateCount      INTEGER,
    clearedCount            INTEGER,
    status                  ErrorCauses
    }
GetAlarmIRPVersionReply ::= SEQUENCE
    {
    versionNumberList   SupportedAlarmIRPVersions,
    status              ErrorCauses
    }
GetAlarmList ::= SEQUENCE
    {
    alarmAckState       AlarmChoice OPTIONAL,
    destination         Destination,                -- ITU-T X.721
    filter              CMISFilter OPTIONAL-- ITU-T X.711
    }
GetAlarmListReply ::= SEQUENCE
    {
    alignmentId     INTEGER,
    status          ErrorCauses
    }
GetNotificationProfileReply ::= SEQUENCE
    {
    notificationNameProfile         NotificationList,
    notificationParameterProfile    ParameterListOfList,
    status                          ErrorCauses
    }

GetOperationProfileReply ::= SEQUENCE
```

{
operationNameProfile OperationList,
operationParameterProfile ParameterListOfList,
status ErrorCauses
}

IRPVersionNumber ::= GraphicString

NotificationList ::= SET OF NotificationName

NotificationName ::= GraphicString

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

OperationList ::= SET OF OperationName

OperationName ::= GraphicString

ParameterList ::= SET OF ParameterName

ParameterListOfList ::= SET OF ParameterList

ParameterName ::= GraphicString

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

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

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

Annex A (informative): Change history

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