ETSI TS 132 111-3 V3.3.0 (2000-12)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
Telecommunication Management;
Fault Management;
Part 3: Alarm Integration
Reference Point: CORBA solution set version 1:1
(3GPP TS 32.111-3 version 3.3.0 Release 1999)**

Reference

RTS/TSGS-0532111-3UR

Keywords

UMTS

Intellectual Property Rights

Foreword

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

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

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under www.etsi.org/key .

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 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 specifies the CORBA Solution Set (SS) for the IRP whose semantics is specified in Alarm IRP: Information Service (IS) (3GPP TS 32.111-2 [13]).

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 defines the usage of OMG CORBA Structured Event to carry information defined in notifications carrying alarm information. Clause 7 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] ITU-T Recommendation X.721: "Information technology - Open Systems Interconnection - Structure of management information: Definition of management information".

[2] ITU-T Recommendation X.736: "Information technology – Open Systems Interconnection – Security Alarm Reporting Function".

[3] ITU-T Recommendation X.732: "Information technology – Open Systems Interconnection – Relationship Management Function".

[4] ITU-T Recommendation X.732: "Information technology – Open Systems Interconnection – State Management Function".

[5] ITU-T Recommendation X.732: "Information technology – Open Systems Interconnection – Object Management Function".

[6] OMG TC Document telecom/98-11-01: "OMG Notification Service".

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

[8] 3GPP TS 32.106-8: "Name Convention for Managed Objects".

[9] 3GPP TS 32.106-1: "3G Configuration Management: Concept and Requirements".

[10] 3GPP TS 32.106-2: "Notification IRP: Information Service".

[11] 3GPP TS 32.106-3: "Notification IRP: CORBA Solution Set".

[11] ITU-T Recommendation X.735: "Information technology - Open Systems Interconnection - Systems Management: Log control function".

[12] 3GPP TS 32.111-1: "3G Fault Management".

[13] 3GPP TS 32.111-2: "Alarm Integration Reference Point: Information Service".

[14] 3GPP TS 32.111-4: "Alarm Integration Reference Point: CMIP Solution Set".

3 Definitions and abbreviations

3.1 Definitions

In addition to the terms and definitions defined in 3GPP TS 32.111-2 [13], 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 Solution Set version

The version of this CORBA SS is 1:1, where the first "1" indicates the version number of the Alarm IRP: IS (3GPP TS 32.111-2 [13]) and the second "1" indicates the version number of the present document.

4 Architectural features

The overall architectural feature of Alarm IRP is specified in 3GPP TS 32.111-2 [13]. 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 TC Document telecom [6]).

OMG Event Service 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 3GPP TS 32.111-2 [13].

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 3GPP TS 32.106-2 [10]. 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 3GPP TS 32.106-2 [6]. The name of the grammar is called "`EXTENDED_TCL`". See clause 2.4, Default Filter Constraint Language in 3GPP TS 32.106-2 [6]. This SS shall use this grammar only.

5 Mapping

5.1 Operation and Notification mapping

Alarm IRP: IS 3GPP TS 32.111-2 [13] 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 3GPP 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
`notifyNewAlarm`	`push_structured_events` Note that OMG Notification Service 3GPP TS 32.106-2 [6] defines this method. See clause 8.1	M
`notifyClearedAlarm`	`push_structured_events` See clause 8.1	M
`notifyChangedAlarm`	`push_structured_events` See clause 8.1	M
`notifyAckStateChanged`	`push_structured_events` See clause 8.1	M
`notifyAlarmListRebuilt`	`push_structured_events` See clause 8.1	M

5.2 Operation parameter mapping

Reference 3GPP TS 32.111-2 [13] defines semantics of parameters carried in operations across the Alarm IRP. Table 2 and table 3 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

5.3 Notification parameter mapping

Reference 3GPP TS 32.111-2 [13] defines semantics of parameters carried in notifications across the Alarm IRP. Table 7 and table 8 indicate the mapping of these parameters, as per notification, to their equivalents defined in this SS.

Table 7 and table 8 are relevant for `notifyNewAlarm, notifyChangedAlarm, notifyClearedAlarm, notifyAckStateChanged`.

Table 7: Mapping from IS `notify[New,Changed,Cleared]Alarm` and `notifyAckStateChanged` parameters to SS equivalents

IS Notification parameter	SS Notification parameter	Comment
`notification Header`	`structuredEvent` Note that OMG Notification Service [6] defines this `structuredEvent`. See Clause 4 as well.	Attributes of `notificationHeader` are mapped to attributes of `structuredEvent`. See clause 5.4 for attributes related to `notificationHeader`. See Table 9 for qualifiers for the parameter-attributes. For `notifyNewAlarm`, `notifyChangedAlarm, notifyClearedAlarm` and `notifyAckStateChanged,` the `extendedEventType` shall contain a string of `extendedEventTypeValue.NOTIFY_FM_NEW_ALARM`, `extendedEventTypeValue.NOTIFY_FM_CHANGED_ALARM`, `extendedEventTypeValue.NOTIFY_FM_CLEARED_ALARM`, `extendedEventTypeValue.NOTIFY_FM_ACK_STATE_CHANGED` respectively.
`alarm Information Body`	`structuredEvent`	Attributes of `alarmInformationBody` are mapped to attributes of `structuredEvent`. See clause 5.4 for attributes related to `alarmInformationBody`. See table 10 for qualifiers for the parameter-attributes.

Table 8 is relevant for `notifyAlarmListRebuilt`.

Table 8: Mapping from IS `notifyAlarmListRebuilt` parameters to SS equivalents

IS Notification parameter	SS equivalent	Comment
`notification Header`	`structuredEvent`	Attributes of `notificationHeader` are mapped to attributes of `structuredEvent`. See clause 5.4 for attributes related to `notificationHeader`. See Table 9 for qualifiers for the parameter-attributes. The `eventType` shall contain a zero-length string. The `extendedEventType` shall contain a string of `extendedEventTypeValue.NOTIFY_FM_ALARM_LIST_REBUILT.` The `managedObjectInstance` shall carries the DN of the IRPAgent whose Alarm List has been rebuilt. Syntax and semantics of this string conform to the Managed Object string representation specified in [8].
`reason`	`reason`	It is a string indicating the Alarm List rebuilt reason.

5.4 Parameter Attribute mapping

Notification IRP: IS 3GPP TS 32.106-2 [10] defines the semantics of attributes for `notificationHeader` parameter. Alarm IRP: IS 3GPP TS 32.111-2 [13] identifies `notificationHeader` for use for its IRP. 3GPP TS 32.111-2 [13] also qualifies the attributes of the `notificationHeader` parameter. Table 9 shows the mapping of these IS attributes to SS equivalents.

Table 9: Mapping from IS `notificationHeader` attributes to SS equivalents

IS Attribute of `notificationHeader` in [10]	SS Attribute	Qualifier
managedObjectClass	managedObjectClass	M
managedObjectInstance	managedObjectInstance	M
notificationID	notificationID	M
eventTime	eventTime	M
systemDN	systemDN	M
eventType	eventType	M
extendedEventType	extendedEventType	M

6 Use of OMG Structured Event

Operation `notify` defined in 3GPP TS 32.111-2 [13] carries parameters, such as `notificationHeader` and `alarmInformationBody`. In CORBA SS, OMG defined `StructuredEvent` (see ITU-T Recommendation X.736 [2]) is used to carry notification. This clause identifies the OMG defined `StructuredEvent` attributes that carry the attributes of parameters defined in 3GPP TS 32.111-2 [13].

The composition of OMG Structured Event, as defined in the OMG TC Document telecom [6], is:

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

Table 11 lists all OMG Structured Event attributes in the second column. The first column identifies the SS attributes, if any, that shall be carried in the Structured Event attributes.

Attributes that are denoted as “optional” in subclause 5.4 of the present document may be absent from the OMG Structured Event. As an example, if the optional `monitoredAttributes` attribute is not used for a particular notification, then the `IRPAgent` may exclude `monitoredAttributes` from the filterable body fields for that particular notification. Individual notifications from the same `IRPAgent` may include or exclude the same optional attribute.

Table 11: Use of OMG Structured Event

SS Attribute	OMG CORBA Structured Event attribute	Comment
There is no corresponding SS attribute.	domain_name	It contains a string defined by `interface IRPNotificationCategoryValue.alarmIRPVersion_1_1`. It indicates the syntax and semantics of this Structured Event defined by Alarm IRP: CORBA SS 1:1.
eventType	type_name	Attribute eventType is an attribute of `notificationHeader`. It shall indicate one of the following ITU-T defined semantics: communications alarm, processing error alarm, environmental alarm,

SS Attribute	OMG CORBA Structured Event attribute	Comment
		quality of service alarm and equipment alarm. It is a string. See block of const string definitions starting with "ET_" in the IDL.
extendedEvent Type	event_name	Attribute extendedEventType is an attribute of notificationHeader. It shall identify one of the following: - notify a new alarm - notify changes in alarm state - notify changes in alarm acknowledgement state - notify alarm cleared - notify Alarm List has been successfully rebuilt It is a string. See block of const string definitions starting with "NOTIFY_FM_" in the IDL.
There is no corresponding SS attribute.	variable Header	
managedObject Class, managedObjectI nstance	One NV pair of filterable_ body_fields	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. They are attributes of notificationHeader. Name of NV pair is a string, NV_MANAGED_OBJECT_INSTANCE defined in module NotificationIRPConstDefs. Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.106-3 [11]).
notification Id	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is a string, NV_NOTIFICATION_ID defined in module NotificationIRPConstDefs. Value of NV pair is a long. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.106-3 [11]).
eventTime	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is NV_EVENT_TIME defined in module NotificationIRPConstDefs. Value of NV pair is a IRPTime. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.106-3 [11]).
systemDN	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is a string, NV_SYSTEM_DN defined in module NotificationIRPConstDefs. Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS [11].
probableCause	One NV pair of filterable_ body_fields	It is an attribute of alarmInformationBody. Name of NV pair is a string, NV_PROBABLE_CAUSE defined in module NotificationIRPConstDefs. Value of NV pair is a short defined by PC_INDETERMINATE, PC_ALARM_INDICATION_SIGNAL, etc.
perceived Severity	One NV pair of filterable_ body_fields	It is an attribute of alarmInformationBody. Name of NV pair is a string, NV_PERCEIVED_SEVERITY defined in module NotificationIRPConstDefs. Value of NV pair is a short defined by PS_INDETERMINATE, PS_CRITICAL, etc.
specific Problem	One NV pair of filterable_ body_fields	It is an attribute of alarmInformationBody. Name of NV pair is a string, NV_SPECIFIC_PROBLEM defined in module NotificationIRPConstDefs. Value of NV pair is a string.
correlated Notifications	One NV pair of filterable_ body_fields	It is an attribute of alarmInformationBody. Name of NV pair is a string, NV_CORRELATED_NOTIFICATIONS defined in module NotificationIRPConstDefs. Value of NV pair is a CorrelatedNotificationSetType.
backed UpStatus	One NV pair of filterable_ body_fields	It is an attribute of alarmInformationBody. Name of NV pair is a string, NV_BACKED_UP_STATUS defined in module NotificationIRPConstDefs. Value of NV pair is a boolean BackedUpStatusType.
backUpObject	One NV pair of filterable_ body_fields	It is an attribute of alarmInformationBody. Name of NV pair is a string, NV_BACK_UP_OBJECT defined in module NotificationIRPConstDefs.

SS Attribute	OMG CORBA Structured Event attribute	Comment
		Value of NV pair is a string carrying of DN of the back-up object. See 3GPP TS 32.106-8 [8] for the DN string representation.
`trend Indication`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_TREND_INDICATION defined in module NotificationIRPConstDefs. Value of NV pair is an `enum TrendIndicationType`.
`thresholdInfo`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_THRESHOLD_INFO defined in module NotificationIRPConstDefs. Value of NV pair is an `enum ThresholdIndicationType`.
`stateChange Definition`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_STATE_CHANGE_DEFINITION defined in module NotificationIRPConstDefs. Value of NV pair is an `AttributeChangeSetType.`
`monitored Attributes`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_MONITORED_ATTRIBUTES defined in module NotificationIRPConstDefs. Value of NV pair is an `AttributeSetType`.
`proposed RepairActions`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_PROPOSED_REPAIR_ACTIONS defined in module NotificationIRPConstDefs. Value of NV pair is a string.
`additional Text`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_ADDITIONAL_TEXT defined in module NotificationIRPConstDefs. Value of NV pair is a string.
`additional Information.al armId`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_ALARM_ID defined in module NotificationIRPConstDefs. 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 of `managedObjectInstance, 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.
`additional Information. ackTime`	One NV pair of `filterable_ body_fields`	It is an attribute of `notificationHeader`. Name of NV pair is a string, NV_ACK_TIME defined in module NotificationIRPConstDefs. Value of NV pair is a `IRPTime`.
`additional Information. ackUserId`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_ACK_USER_ID defined in module NotificationIRPConstDefs. Value of NV pair is a string.
`additional Information. ackSystemId`	One NV pair of `filterable_ body_fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_ACK_SYSTEM_ID defined in module NotificationIRPConstDefs. Value of NV pair is a string.
`additional Information. ackState`	One NV pair of `filterable_body_ fields`	It is an attribute of `alarmInformationBody`. Name of NV pair is a string, NV_ACK_STATE defined in module NotificationIRPConstDefs. Value of NV pair is a short defined by ACK_STATE_ACKNOWLEDGED and ACK_STATE_UNACKNOWLEDGED.
There is no corresponding SS attribute.	`remaining_ body`	

7 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_events` method.

7.1 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 TC Document telecom [6]). 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

```
/* ## Module: AlarmConstDefs

This module contains commonly used definitions.
===============================================================================
 */

#ifndef  AlarmIRPConstDefs_idl
#define  AlarmIRPConstDefs_idl

#include "CosNotification.idl"

#pragma prefix "3gppsa5.org"
module AlarmIRPConstDefs {

  /*
  This block identifies all TMN ITU-T defined event types used by Alarm
  IRP of this version.  Their semantics are defined by ITU-T.  Their
  encodings for this version of Alarm IRP are defined here.  Other IRP
  documents, or other versions of Alarm IRP, shall identify their own
  ITU-T defined event types for their use.  They shall define their encodings
  as well.  Note all values are unique among themselves.  Other IRP documents
  can use the same values.
  */

  const string ET_COMMUNICATIONS_ALARM = "x1";
  const string ET_PROCESSING_ERROR_ALARM = "x2";
  const string ET_ENVIRONMENTAL_ALARM = "x3";
  const string ET_QUALITY_OF_SERVICE_ALARM = "x4";
  const string ET_EQUIPMENT_ALARM = "x5";


  /*
  This block identifies IRP defined, and not ITU-T defined, event types
  used by this Alarm IRP version.
  */

  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_CLEARED_ALARM = "x4";
  const string NOTIFY_FM_ALARM_LIST_REBUILT = "x5";

  /*
  It indicates if an object has a back up.
  True implies backep 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 in a notification to report a changed attribute value.
*/

 struct AttributeValueChangeType {
   string   attributeName;
   any      oldValue; // type depends on attribute
   any      newValue; // type depends on attribute
 };

typedef sequence <AttributeValueChangeType> AttributeChangeSetType;

/*
It is used in a notification to report a changed attribute value.
*/

struct AttributeValueType {
   string attributeName;
   any  value;    // type will depend on the attribute
 };

 typedef sequence <AttributeValueType> AttributeSetType;


/*
This block identifies the levels of severity.
*/

const short PS_INDETERMINATE= 1;
const short PS_CRITICAL= 2;
const short PS_MAJOR= 3;
const short PS_MINOR= 4;
const short PS_WARNING= 5;
const short PS_CLEARED= 6;

/*
This block identifies the acknowledgement state reported alarm.
*/
  const short ACK_STATE_ACKNOWLEDGED= 1;
  const short ACK_STATE_UNACKNOWLEDGED= 2;

/*
This block identifies the probable cause of a reported alarm.
*/

   const short PC_INDETERMINATE = 0;
   const short PC_ALARM_INDICATION_SIGNAL = 1;
   const short PC_CALL_SETUP_FAILURE = 2;
   const short PC_DEGRADED_SIGNAL_M3100 = 3;
   const short PC_FAR_END_RECEIVER_FAILURE = 4;
   const short PC_FRAMING_ERROR_M3100 = 5;
   const short PC_LOSS_OF_FRAME = 6;
```

```
    const short PC_LOSS_OF_POINTER = 7;
    const short PC_LOSS_OF_SIGNAL = 8;
    const short PC_PAYLOAD_TYPE_MISMATCH = 9;
    const short PC_TRANSMISSION_ERROR = 10;
    const short PC_REMOTE_ALARM_INTERFACE = 11;
    const short PC_EXCESSIVE_BIT_ERROR_RATE = 12;
    const short PC_PATH_TRACE_MISMATCH = 13;
    const short PC_UNAVAILABLE = 14;
    const short PC_SIGNAL_LABEL_MISMATCH = 15;
    const short PC_LOSS_OF_MULTI_FRAME = 16;
    const short PC_BACK_PLANE_FAILURE = 51;
    const short PC_DATA_SET_PROBLEM = 52;
    const short PC_EQUIPMENT_IDENTIFIER_DUPLICATION = 53;
    const short PC_EXTERNAL_DEVICE_PROBLEM = 54;
    const short PC_LINE_CARD_PROBLEM = 55;
    const short PC_MULTIPLEXER_PROBLEM_M3100 = 56;
    const short PC_NE_IDENTIFIER_DUPLICATION = 57;
    const short PC_POWER_PROBLEM_M3100 = 58;
    const short PC_PROCESSOR_PROBLEM_M3100 = 59;
    const short PC_PROTECTION_PATH_FAILURE = 60;
    const short PC_RECEIVER_FAILURE_M3100 = 61;
    const short PC_REPLACEABLE_UNIT_MISSING = 62;
    const short PC_REPLACEABLE_UNIT_TYPE_MISMATCH = 63;
    const short PC_SYNCHRONISATION_SOURCE_MISMATCH = 64;
    const short PC_TERMINAL_PROBLEM = 65;
    const short PC_TIMING_PROBLEM_M3100 = 66;
    const short PC_TRANSMITTER_FAILURE_M3100 = 67;
    const short PC_TRUNK_CARD_PROBLEM = 68;
    const short PC_REPLACEABLE_UNIT_PROBLEM = 69;
    const short PC_AIR_COMPRESSOR_FAILURE = 101;
    const short PC_AIR_CONDITIONING_FAILURE = 102;
    const short PC_AIR_DRYER_FAILURE = 103;
    const short PC_BATTERY_DISCHARGING = 104;
    const short PC_BATTERY_FAILURE = 105;
    const short PC_COMMERICAL_POWER_FAILURE = 106;
    const short PC_COOLING_FAN_FAILURE = 107;
    const short PC_ENGINE_FAILURE = 108;
    const short PC_FIRE_DETECTOR_FAILURE = 109;
    const short PC_FUSE_FAILURE = 110;
    const short PC_GENERATOR_FAILURE = 111;
    const short PC_LOW_BATTERY_THRESHOLD = 112;
    const short PC_PUMP_FAILURE_M3100 = 113;
    const short PC_RECTIFIER_FAILURE = 114;
    const short PC_RECTIFIER_HIGH_VOLTAGE = 115;
    const short PC_RECTIFIER_LOW_F_VOLTAGE = 116;
    const short PC_VENTILATION_SYSTEM_FAILURE = 117;
    const short PC_ENCLOSURE_DOOR_OPEN_M3100 = 118;
    const short PC_EXPLOSIVE_GAS = 119;
    const short PC_FIRE = 120;
    const short PC_FLOOD = 121;
    const short PC_HIGH_HUMIDITY = 122;
    const short PC_HIGH_TEMPERATURE = 123;
    const short PC_HIGH_WIND = 124;
    const short PC_ICE_BUILD_UP = 125;
    const short PC_LOW_FUEL = 127;
    const short PC_LOW_HUMIDITY = 128;
    const short PC_LOW_CABLE_PRESSURE = 129;
    const short PC_LOW_TEMPERATURE = 130;
    const short PC_LOW_WATER = 131;
    const short PC_SMOKE = 132;
    const short PC_TOXIC_GAS = 133;
    const short PC_STORAGE_CAPACITY_PROBLEM_M3100 = 151;
    const short PC_MEMORY_MISMATCH = 152;
    const short PC_CORRUPT_DATA_M3100 = 153;
    const short PC_OUT_OF_CPU_CYCLES = 154;
    const short PC_SOFTWARE_ENVIRONMENT_PROBLEM = 155;
    const short PC_SOFTWARE_DOWNLOAD_FAILURE = 156;
    const short PC_ADAPTER_ERROR = 301;
    const short PC_APPLICATION_SUBSYSTEM_FAILURE = 302;
    const short PC_BANDWIDTH_REDUCTION = 303;
    const short PC_COMMUNICATION_PROTOCOL_ERROR = 305;
    const short PC_COMMUNICATION_SUBSYSTEM_FAILURE = 306;
    const short PC_CONFIGURATION_OR_CUSTOMIZING_ERROR = 307;
    const short PC_CONGESTION = 308;
    const short PC_CPU_CYCLES_LIMIT_EXCEEDED = 310;
    const short PC_DATA_SET_OR_MODEM_ERROR = 311;
    const short PC_DTE_DCE_INTERFACE_ERROR = 313;
    const short PC_EQUIPMENT_MALFUNCTION = 315;
    const short PC_EXCESSIVE_VIBRATION = 316;
    const short PC_FILE_ERROR = 317;
    const short PC_HEATING_OR_VENTILATION_OR_COOLING_SYSTEM_PROBLEM = 321;
    const short PC_HUMIDITY_UNACCEPTABLE = 322;
    const short PC_INPUT_OUTPUT_DEVICE_ERROR = 323;
    const short PC_INPUT_DEVICE_ERROR = 324;
    const short PC_LAN_ERROR = 325;
    const short PC_LEAK_DETECTION = 326;
    const short PC_LOCAL_NODE_TRANSMISSION_ERROR = 327;
    const short PC_MATERIAL_SUPPLY_EXHAUSTED = 330;
    const short PC_OUT_OF_MEMORY = 332;
    const short PC_OUTPUT_DEVICE_ERROR = 333;
    const short PC_PERFORMANCE_DEGRADED = 334;
    const short PC_PRESSURE_UNACCEPTABLE = 336;
    const short PC_QUEUE_SIZE_EXCEEDED = 339;
    const short PC_RECEIVE_FAILURE = 340;
    const short PC_REMOTE_NODE_TRANSMISSION_ERROR = 342;
    const short PC_RESOURCE_AT_OR_NEARING_CAPACITY = 343;
    const short PC_RESPONSE_TIME_EXCESSIVE = 344;
    const short PC_RETRANSMISSION_RATE_EXCESSIVE = 345;
    const short PC_SOFTWARE_ERROR = 346;
    const short PC_SOFTWARE_PROGRAM_ABNORMALLY_TERMINATED = 347;
    const short PC_SOFTWARE_PROGRAM_ERROR = 348;
    const short PC_TEMPERATURE_UNACCEPTABLE = 350;
    const short PC_THRESHOLD_CROSSED = 351;
    const short PC_TOXIC_LEAK_DETECTED = 353;
    const short PC_TRANSMIT_FAILURE = 354;
    const short PC_UNDERLYING_RESOURCE_UNAVAILABLE = 356;
    const short PC_VERSION_MISMATCH = 357;
    const short PC_A_BIS_TO_BTS_INTERFACE_FAILURE = 501;
    const short PC_A_BIS_TO_TRX_INTERFACE_FAILURE = 502;
    const short PC_ANTENNA_PROBLEM = 503;
    const short PC_BATTERY_BREAKDOWN = 504;
    const short PC_BATTERY_CHARGING_FAULT = 505;
    const short PC_CLOCK_SYNCHRONISATION_PROBLEM = 506;
    const short PC_COMBINER_PROBLEM = 507;
    const short PC_DISK_PROBLEM = 508;
    const short PC_EXCESSIVE_RECEIVER_TEMPERATURE = 510;
    const short PC_EXCESSIVE_TRANSMITTER_OUTPUT_POWER = 511;
    const short PC_EXCESSIVE_TRANSMITTER_TEMPERATURE = 512;
    const short PC_FREQUENCY_HOPPING_DEGRADED = 513;
    const short PC_FREQUENCY_HOPPING_FAILURE = 514;
    const short PC_FREQUENCY_REDEFINITION_FAILED = 515;
    const short PC_LINE_INTERFACE_FAILURE = 516;
```

```
     const short PC_LINK_FAILURE = 517;
     const short PC_LOSS_OF_SYNCHRONISATION = 518;
     const short PC_LOST_REDUNDANCY = 519;
     const short PC_MAINS_BREAKDOWN_WITH_BATTERY_BACKUP = 520;
     const short PC_MAINS_BREAKDOWN_WITHOUT_BATTERY_BACKUP = 521;
     const short PC_POWER_SUPPLY_FAILURE = 522;
     const short PC_RECEIVER_ANTENNA_FAULT = 523;
     const short PC_RECEIVER_MULTICOUPLER_FAILURE = 525;
     const short PC_REDUCED_TRANSMITTER_OUTPUT_POWER = 526;
     const short PC_SIGNAL_QUALITY_EVALUATION_FAULT = 527;
     const short PC_TIMESLOT_HARDWARE_FAILURE = 528;
     const short PC_TRANSCEIVER_PROBLEM = 529;
     const short PC_TRANSCODER_PROBLEM = 530;
     const short PC_TRANSCODER_OR_RATE_ADAPTER_PROBLEM = 531;
     const short PC_TRANSMITTER_ANTENNA_FAILURE = 532;
     const short PC_TRANSMITTER_ANTENNA_NOT_ADJUSTED = 533;
     const short PC_TRANSMITTER_LOW_VOLTAGE_OR_CURRENT = 535;
     const short PC_TRANSMITTER_OFF_FREQUENCY = 536;
     const short PC_DATABASE_INCONSISTENCY = 537;
     const short PC_FILE_SYSTEM_CALL_UNSUCCESSFUL = 538;
     const short PC_INPUT_PARAMETER_OUT_OF_RANGE = 539;
     const short PC_INVALID_PARAMETER = 540;
     const short PC_INVALID_POINTER = 541;
     const short PC_MESSAGE_NOT_EXPECTED = 542;
     const short PC_MESSAGE_NOT_INITIALISED = 543;
     const short PC_MESSAGE_OUT_OF_SEQUENCE = 544;
     const short PC_SYSTEM_CALL_UNSUCCESSFUL = 545;
     const short PC_TIMEOUT_EXPIRED = 546;
     const short PC_VARIABLE_OUT_OF_RANGE = 547;
     const short PC_WATCH_DOG_TIMER_EXPIRED = 548;
     const short PC_COOLING_SYSTEM_FAILURE = 549;
     const short PC_EXTERNAL_EQUIPMENT_FAILURE = 550;
     const short PC_EXTERNAL_POWER_SUPPLY_FAILURE = 551;
     const short PC_EXTERNAL_TRANSMISSION_DEVICE_FAILURE = 552;
     const short PC_REDUCED_ALARM_REPORTING = 561;
     const short PC_REDUCED_EVENT_REPORTING = 562;
     const short PC_RECUCED_LOGGING_CAPABILITY = 563;
     const short PC_SYSTEM_RESOURCES_OVERLOAD = 564;
     const short PC_BROADCAST_CHANNEL_FAILURE = 565;
     const short PC_CALL_ESTABLISHMENT_ERROR = 566;
     const short PC_INVALID_MESSAGE_RECEIVED = 567;
     const short PC_INVALID_MSU_RECEIVED = 568;
     const short PC_LAPD_LINK_PROTOCOL_FAILURE = 569;
     const short PC_LOCAL_ALARM_INDICATION = 570;
     const short PC_REMOTE_ALARM_INDICATION = 571;
     const short PC_ROUTING_FAILURE = 572;
     const short PC_SS7_PROTOCOL_FAILURE = 573;
     const short PC_TRANSMISSION_FAILURE = 574;


   typedef sequence <string> AlarmInformationIdSeq;

   typedef CosNotification::EventBatch AlarmInformationSeq;


};

#endif


/* ## Module: AlarmIRPSystem
```

```
This module contains the specification of all operations of Alarm IRP Agent
specified in Alarm IRP: IS version 1 and Alarm IRP: CORBA SS version 1:1.
=============================================================================
*/

#ifndef  AlarmIRPSystem_idl
#define  AlarmIRPSystem_idl

#include "CosNotification.idl"
#include "AlarmIRPConstDefs.idl"
#include "CommonIRPConstDefs.idl"
#pragma prefix "3gppsa5.org"

module AlarmIRPSystem {

  /*
  System fails to complete the operation.  System provides
  reasons whose semantics is outside the scope of this IRP.
  */

  exception AcknowledgeAlarms { string reason; };
  exception UnacknowledgeAlarms { string reason; };
  exception GetAlarmList {string reason; };
  exception GetAlarmIRPVersion { string reason; };
  exception GetAlarmCount { string reason; };
  exception ParameterNotSupported { string parameter; };
    //name of the unsupported parameter as defined in IDL.
  exception InvalidParameter { string parameter; };
    //name of the parameter as defined in IDL
  exception OperationNotSupported {};
  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,InvalidParameter);

        /**
        This method destroys the iterator.
        */

        void destroy ();

    }; // end of AlarmInformationIterator


  /*
  This interface specifies all methods supported by System as
  specified in 3GPP AlarmIRP: CORBA Solution Set version 1:1.
  */

  interface AlarmIRPOperations {

     CommonIRPConstDefs::Signal acknowledge_alarms (
       in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
       in string ack_user_id,
       in string ack_system_id,
       out AlarmIRPConstDefs::AlarmInformationIdSeq
      bad_alarm_information_id_list
    )
    raises (AcknowledgeAlarms,ParameterNotSupported,InvalidParameter);


     CommonIRPConstDefs::Signal unacknowledge_alarms (
       in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
       in string ack_user_id,
       in string ack_system_id,
       out AlarmIRPConstDefs::AlarmInformationIdSeq
         bad_alarm_information_id_list
    )
    raises (UnacknowledgeAlarms, OperationNotSupported, ParameterNotSupported,
         InvalidParameter);



     /*
     This method returns Alarm Informations.
     If flag is TRUE, all returned Alarm Informations shall be
     in AlarmInformationSeq that contains 0,1 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,ParameterNotSupported,InvalidParameter);


     void get_alarm_count (
       in string filter,
       out long critical_count,
       out long major_count,
       out long minor_count,
       out long warning_count,
       out long indeterminate_count,
       out long cleared_count
    )
    raises (GetAlarmCount, OperationNotSupported, ParameterNotSupported,
         InvalidParameter);


     CommonIRPConstDefs::VersionNumberSet get_alarm_IRP_version ()
       raises (GetAlarmIRPVersion);
 };

};

#endif
```

Annex B (informative): Change history

Change history					
TSG SA#	**Version**	**CR**	**Tdoc SA**	**New Version**	**Subject/Comment**
S_07	2.0.0	-	SP-000012	3.0.0	Approved at TSG SA #7 and placed under Change Control
Mar 2000	3.0.0			3.0.1	cosmetic
S_08	3.0.1	005	SP-000253	3.1.0	Split of TS - Part 3: Alarm Integration Reference Point (IRP): CORBA Solution Set (SS)
S_09	3.1.0	003	SP-000439	3.2.0	Correct push_structured_event of push_structured_events
S_09	3.1.0	004	SP-000439	3.2.0	Remove the use of interface to encapsulate const strings
S_10	3.2.0	001R1	SP-000521	3.3.0	Allow “Structured Event Filterable Body Fields” to be absent if parameters are not used
S_10	3.2.0	002R1	SP-000521	3.3.0	Specific behaviour of the Iterator
S_10	3.2.0	005	SP-000521	3.3.0	Inconsistent qualifiers

History

Document history		
V3.1.0	July 2000	Publication
V3.2.0	September 2000	Publication
V3.3.0	December 2000	Publication