Technical Specification Group Services and System Aspects
Meeting #18, New Orleans, USA, 9-12 December 2002

TSGS#18(02)0752

Source: **SA5 (Telecom Management)**

Title: **Rel-5 CR 32.111-3 (Alarm IRP: CORBA SS) : Add notifyPotentialFaultyAlarmList (align with 32.111-2 Alarm IRP: Information Service)**

Document for: **Approval**

Agenda Item: **7.5.3**

Doc-1st-	Spec	CR	R	Phase	Subject	Cat	Version	Doc-2nd-	Workitem
SP-020752	32.111-3	024	-	Rel-5	Add notifyPotentialFaultyAlarmList in Alarm IRP: CORBA SS (Alignment with Information Service in Rel-5 32111-2)	F	5.1.0	S5-026932	OAM-NIM

3GPP TSG-SA5 (Telecom Management) **S5-026932**
Meeting #32, Vienna, Austria, 18-22 November 2002

CR-Form-v7

CHANGE REQUEST

⌘ **32.111-3** CR **024** ⌘rev **-** ⌘ Current version: **5.1.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☒ Core Network ☒

Title:	⌘	Add notifyPotentialFaultyAlarmList in Alarm IRP: CORBA SS (Alignment with Information Service in Rel-5 32111-2)
Source:	⌘	SA5
Work item code:	⌘	OAM-NIM
Date:	⌘	07/11/2002
Category:	⌘	**F**
Release:	⌘	REL-5

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

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Correction to align the missing feature in IS.
Summary of change:	⌘	Add a row in mapping table to map the IS notification construct to SS construct. Add a new row in Table 13 for a new parameter. Add a new table specifying the new notification. Add a new const definition in IDL.
Consequences if not approved:	⌘	SS misaligned with IS.

Clauses affected:	⌘	SubClause 5.1 Table 1. SubClause 5.3 Table 13. SubClause 5.3 Table 15. SubClause 5.3, add a New Table 16. Annex A.1.

		Y	N		
Other specs affected:	⌘		X	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	

Other comments:	⌘	

Change in SubClause 5.1 Table 1.

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

IS Operation/ notification 3G TS 32.111-2 [6]	SS Method	Qualifier
acknowledgeAlarms	acknowledge_alarms	M
unacknowledgeAlarms	unacknowledge_alarms	O
getAlarmList	get_alarm_list	M
getIRPVersion	get_alarm_IRP_versions	M
getAlarmCount	get_alarm_count	O
setComment	comment_alarms	O
clearAlarms	clear_alarms	O
getOperationProfile (note)	get_alarm_IRP_operations_profile	O
getNotificationProfile (note)	get_alarm_IRP_notification_profile	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
notifyPotentialFaultyAlarmList	push_structured_event See clause 6.1	O
NOTE: This operation is of ManagedGenericIRP IOC specified in [7]. The AlarmIRP IOC of [6] inherits from it.		

End of Change in SubClause 5.1 Table 1.

Change in SubClause 5.3 Table 13.

Table 13: Mapping for `notifyAlarmListRebuilt`

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_ALARM_LIST_REBUILT of interface NotificationType of module AlarmIRPConstDefs.
There is no corresponding IS attribute.	event_name	M	Carry an empty string.
There is no corresponding IS attribute.	variable Header		
objectClass, objectInstance	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	Name of NV pair is the REASON of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
alarmListAlignmentRequirement	One NV pair of filterable_ body_fields	O	Name of NV pair is the ALARM_LIST_ALIGNMENT_REQUIREMENT of interface AttributeNameValue of modue AlarmIRPConstDefs. Value of NV pair is an enum AlarmListAlignmentRequirementType of module AlarmIRPConstDefs.
There is no corresponding IS attribute.	remaining_-body		

End of Change in SubClause 5.3 Table 13.

Change in SubClause 5.3 Table 15

Table 15: 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_COMMENT_ADDED of interface NotificationType of module AlarmIRPConstDefs.
alarmType	event_name	M	See that of notifyNewAlarm.
There is no corresponding IS attribute.	variable Header		
objectClass, objectInstance	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.
comments	One NV pair of filterable_ body_fields	M	Name of NV pair is the COMMENTS of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a CommentSet of module AlarmIRPConstDefs.
There is no corresponding IS attribute.	remaining_-body		

End of Change in SubClause 5.3 Table 15.

Change in SubClause 5.3, add Table 16.

Table 16: Mapping for `notifyPotentialFaultyAlarmList`

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_POTENTIAL_FAULTY_ALARM_LIST of interface NotificationType of module AlarmIRPConstDefs.
There is no corresponding IS attribute.	event_name	M	It contains a NULL string.
There is no corresponding IS attribute.	variable Header		
objectClass, objectInstance	One NV pair of filterable_body_fields	M	See notifyNewAlarm. See sub-clause “Definition” of this notification in [6] for the description of the ussage of this field to indicate if part or all AlarmList is potentially faulty.
notification Id	One NV pair of filterable_body_fields	M	
eventTime	One NV pair of filterable_body_fields	M	See notifyNewAlarm.
systemDN	One NV pair of filterable_body_fields	M	See notifyNewAlarm.
reason	One NV pair of filterable_body_fields	M	Name of NV pair is the REASON of interface AttributeNameValue of module AlarmIRPConstDefs. Value of NV pair is a string.
There is no corresponding IS attribute.	remaining_body		

End of Change in SubClause 5.3 to add a new Table 16

Change in Annex A.1

Annex A (normative):
IDL specifications

A.1 IDL specification (file name "AlarmIRPConstDefs.idl")

```
#ifndef AlarmIRPConstDefs_idl
#define AlarmIRPConstDefs_idl

#include "CosNotification.idl"
#include "ManagedGenericIRPConstDefs.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
{
    */
   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";
      const string INTEGRITY_VIOLATION = "x6";
      const string OPERATIONAL_ VIOLATION = "x7";
      const string PHYSICAL_VIOLATION = "x8";
      const string SECURITY_SERVICE_OR_MECHANISM_VIOLATION = "x9";
      const string TIME_DOMAIN_VIOLATION = "x10";
   };

   /*
   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";
```

```
      const string NOTIFY_FM_POTENTIAL_FAULTY_ALARM_LIST = "x7";
};

/*
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 INDETERMINATE = 0;
   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;
   const short AUTHENTICATION_FAILURE= 575;
   const short BREACH_OF_CONFIDENTIALITY = 576;
   const short CABLE_TAMPER = 577;
   const short DELAYED_INFORMATION = 578;
   const short DENIAL_OF_SERVICE = 579;
   const short DUPLICATE_INFORMATION = 580;
   const short INFORMATION_MISSING = 581;
   const short INFORMATION_MODIFICATION_DETECTED = 582;
   const short INFORMATION_OUT_OF_SEQUENCE = 583;
   const short INTRUSION_DETECTION = 584;
   const short KEY_EXPIRED = 585;
   const short NON_REPUDIATION_FAILURE = 586;
   const short OUT_OF_HOURS_ACTIVITY = 587;
   const short OUT_OF_SERVICE = 588;
   const short PROCEDURAL_ERROR = 589;
   const short UNAUTHORISED_ACCESS_ATTEMPT = 590;
   const short UNEXPECTED_INFORMATION= 591;
   const short UNSPECIFIED_REASON = 592;
};

/*
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";
      const string CLEAR_USER_ID = "y";"Y";
      const string CLEAR_SYSTEM_ID = "z"; "z";
      const string ALARM_LIST_ALIGNMENT_REQUIREMENT = "ff";
   };

   /*
   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 the AlarmList alignment is required.
   */
   enum AlarmListAlignmentRequirementType {Required, NotRequired};


/* FloatTypeOpt is an optional type.
   If the discriminator is true the value is present.
   Otherwise the value is null.
*/
union FloatTypeOpt switch (boolean)
{
   case TRUE: float value;
};


/* ThresholdLevelIndType describes multi-level
   threshold crossings.
   Up is the only permitted choice for a counter.
   If indication is "up", low value is optional.

   @member indication: indicates up or down direction
     of crossing.
   @member low: the low observed value.
   @member high: the high observed value.
*/
```

```
struct ThresholdLevelIndType
{
    ThresholdIndicationType indication;
    FloatTypeOpt low;
    float high;
};

/* ThresholdLevelIndTypeOpt is an optional type.
   If the discriminator is true the value is present.
   Otherwise, the value is null.
*/

union ThresholdLevelIndTypeOpt switch (boolean)
{
  case TRUE: ThresholdLevelIndType value;
};


/* ThresholdInfoType indicates some guage or counter
   attribute passed a set threshold.

   @member attributeID: identifies the attribute that
     crossed the threshold.
   @member observedValue: attributes that are of type
     integer will be converted to floats.
   @member thresholdlevel: This parameter is for
     multi-level threhsolds. Optional.
   @member armTime: May contain empty string.
*/

struct ThresholdInfoType
{
  string attributeID;
  float observedValue;
  ThresholdLevelIndTypeOpt thresholdLevel;
  string armTime;
};


   /*
   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 of Alarm ID and Perceived Severity used within the
alarm acknowledgment operation. Note: perceivedSeverity is an optional
parameter.
*/
struct AlarmInformationIdAndSev
{
   string alarm_information_reference;
   PerceivedSeverity perceived_severity;
};

/*
Define set of the above structure of Alarm ID and Perceived Severity.
*/
typedef sequence <AlarmInformationIdAndSev> AlarmInformationIdAndSevSeq;

/*
It indicates the reason for an alarm acknowledgement to have failed:
  - The specified Alarm Information is absent from the Alarm List
  - The Perceived Severity to be acknowledged has changed and/or is different
    within the Alarm List
  - The acknowledgement failed for some other reason
*/
enum AcknowledgeFailureCategories
{
   UnknownAlarmId,
   WrongPerceivedSeverity,
   AcknowledgmentFailed
};

/*
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 BadAlarmInformationId
```

```
    {
       string alarm_information_reference;
       string reason;
    };

    /*
    Define the structure returned when the acknowledge operation fails for a set
    of alarm ids.
    A failure category and a reason are provided in order to indicate why the
    operation failed.
    */
    struct BadAcknowledgeAlarmInfo
    {
       string alarm_information_reference;
       AcknowledgeFailureCategories failure_category;
       string reason;
    };

    typedef sequence <BadAlarmInformationId> BadAlarmInformationIdSeq;
    typedef sequence <BadAcknowledgeAlarmInfo> BadAcknowledgeAlarmInfoSeq;
    typedef sequence <string> AlarmInformationIdSeq;
    typedef CosNotification::EventBatch AlarmInformationSeq;
};
#endif
```

End of Change in Annex A.1