

**Source:** SA5 (Telecom Management)

**Title:** 3 Rel-4/5 CRs 32111-3, 32.303 (Fault Management; Part 3: Alarm IRP; CORBA SS & Configuration Management (CM); Notification IRP; CORBA SS) "Correction of CORBA ALARM\_IRP\_VERSION in line with adopted Rel-5 policy"

**Document for:** Approval

**Agenda Item:** 7.5.3

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Ca t	Ver- Curr ent	Doc-2nd- Level	Workite m	Remarks
SP-030064	32.111-3	025	-	Rel-4	<b>Correction of CORBA ALARM_IRP_VERSION in line with adopted Rel-5 policy</b>	F	4.5.0	S5-036121	OAM-FM	
SP-030064	32.111-3	026	-	Rel-5	<b>Correction of CORBA ALARM_IRP_VERSION in line with adopted Rel-5 policy</b>	A	5.2.0	S5-036122	OAM-NIM	<b>Parent CR.</b>
SP-030064	32.303	010	-	Rel-5	<b>Update the usage IRP_VERSION in line with adopted release 5 policy - alignment with 32.111-3</b>	F	5.1.2	S5-036133	OAM-NIM	<b>Child CR.</b>

## CHANGE REQUEST

# **32.111-3 CR 025** # rev **-** # Current version: **4.5.0** #

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

**Proposed change affects:** UICC apps#  ME  Radio Access Network  Core Network

<b>Title:</b>	# Correction of CORBA_ALARM_IRP_VERSION in line with adopted Rel-5 policy		
<b>Source:</b>	# S5		
<b>Work item code:</b>	# OAM-FM	<b>Date:</b>	# 28/02/2003
<b>Category:</b>	# <b>F</b>	<b>Release:</b>	# Rel-4
	Use <u>one</u> of the following categories:		Use <u>one</u> of the following releases:
	<b>F</b> (correction)		2 (GSM Phase 2)
	<b>A</b> (corresponds to a correction in an earlier release)		R96 (Release 1996)
	<b>B</b> (addition of feature),		R97 (Release 1997)
	<b>C</b> (functional modification of feature)		R98 (Release 1998)
	<b>D</b> (editorial modification)		R99 (Release 1999)
	Detailed explanations of the above categories can be found in 3GPP <a href="#">TR 21.900</a> .		Rel-4 (Release 4)
			Rel-5 (Release 5)
			Rel-6 (Release 6)

<b>Reason for change:</b>	# The interface version definition does not meet the format rule in 32.111-3.
<b>Summary of change:</b>	# The NOTIFICATION_IRP_VERSION constant definition is to be deleted, and behavioural comments added to clarify the behaviour of get_notification_IRP_versions. This aligns interface version handling with the release 5 approach used for 32.603 release 5.
<b>Consequences if not approved:</b>	# Systems using the original NOTIFICATION_IRP_VERSION constant will be incompatible with systems using the documented rules for defining the version string.

<b>Clauses affected:</b>	# Annex A.1 – delete the NOTIFICATION_IRP_VERSION constant and add comment. Annex A2 – add comments to define the desired behaviour of operation get_notification_IRP_versions						
<b>Other specs affected:</b>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> </table>	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Other core specifications	#
Y	N						
<input type="checkbox"/>	<input checked="" type="checkbox"/>						
	<input checked="" type="checkbox"/>	Test specifications					
	<input checked="" type="checkbox"/>	O&M Specifications	Rel-5 32.111-3, 32.303				
<b>Other comments:</b>	# Parent CR to 32303CR010 S5-036133. Mirror CR in 32111-3CR026 S5-036122.						

**How to create CRs using this form:**



```

    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 = 17;
    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 COMMERCIAL_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 REDUCED_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};

/* 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;

/*
ShortTypeOpt is a type carrying an optional parameter.
If the boolean is TRUE, then the value is present.
Otherwise the value is absent.
*/
union ShortTypeOpt switch (boolean)
{
    case TRUE: short value;
};

/*
Define the structure of Alarm ID and Perceived Severity used within the
alarm acknowledgment operation. Note: perceived_severity is an optional
parameter. If this value is present, it must have one of the defined values
of Interface PerceivedSeverity.
*/
struct AlarmInformationIdAndSev
{
    string alarm_information_reference;
    ShortTypeOpt 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

```

---

## A.2 IDL specification (file name "AlarmIRPSystem.idl")

```

#ifndef AlarmIRPSystem_idl
#define AlarmIRPSystem_idl

#include "AlarmIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"

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

/* ## 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.
    Implementations are to provide a return value consisting of one or more IRPVersions.
    Each IRPVersion is defined by the rule in the clause titled
    "IRP document version number string"

    */
    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::AlarmInformationIdAndSevSeq alarm_information_id_and_sev_list,
    in string ack_user_id,
    in string ack_system_id,
    out AlarmIRPConstDefs::BadAcknowledgeAlarmInfoSeq
    bad_ack_alarm_info_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

## CHANGE REQUEST

# **32.303 CR 010** # rev **-** # Current version: **5.1.2** #

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

<b>Title:</b>	# Update the usage IRP_VERSION in line with adopted release 5 policy - alignment with 32.111-3		
<b>Source:</b>	# S5		
<b>Work item code:</b>	# OAM-NIM	<b>Date:</b>	# 28/02/2003
<b>Category:</b>	# <b>F</b>	<b>Release:</b>	# Rel-5
	Use <u>one</u> of the following categories:		Use <u>one</u> of the following releases:
	<b>F</b> (correction)		2 (GSM Phase 2)
	<b>A</b> (corresponds to a correction in an earlier release)		R96 (Release 1996)
	<b>B</b> (addition of feature),		R97 (Release 1997)
	<b>C</b> (functional modification of feature)		R98 (Release 1998)
	<b>D</b> (editorial modification)		R99 (Release 1999)
	Detailed explanations of the above categories can be found in 3GPP <a href="#">TR 21.900</a> .		Rel-4 (Release 4)
			Rel-5 (Release 5)
			Rel-6 (Release 6)

<b>Reason for change:</b>	# Current usage of the constant definition does not meet the format rule defined in 32.111-3.
<b>Summary of change:</b>	# The constant definition is to be deleted, and behavioural comments added to get_Notification_IRP_versions to define the behaviour. This aligns interface version handling with the release 5 approach used for 32.603 release 5.
<b>Consequences if not approved:</b>	# Systems will encounter version incompatibility problems when using the original NOTIFICATION_IRP_VERSION constant will be incompatible with systems using the documented rules for defining the version string.

<b>Clauses affected:</b>	# Annex A.3 – delete the NOTIFICATION_IRP_VERSION constant and add comment. Annex A4 – add comments to define the desired behaviour of operation get_notification_IRP_versions						
<b>Other specs affected:</b>	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> </table>	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Other core specifications	#
Y	N						
<input type="checkbox"/>	<input checked="" type="checkbox"/>						
	<input checked="" type="checkbox"/>	Test specifications					
	<input checked="" type="checkbox"/>	O&M Specifications					
<b>Other comments:</b>	# Parent CR Rel-4 in 32111-3CR025 S5-036121.						

**How to create CRs using this form:**

## A.3 IDL specification (file name "NotificationIRPConstDefs.idl")

```

#ifndef NotificationIRPConstDefs_idl
#define NotificationIRPConstDefs_idl

#include "ManagedGenericIRPConstDefs.idl"

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

/* ## Module: NotificationIRPConstDefs
This module contains definitions specific for Notification IRP.
=====
*/
module NotificationIRPConstDefs
{

    /*
    Define the current Notification IRP version.
    This string is used for the return value of
    get_Notification_IRP_versions().

    It should be updated based on the rule of subclause
    titled "IRP document version number string".
    */
    const string NOTIFICATION_IRP_VERSION = "<to be updated using the rule>";
    */
    /*
    Define the parameters (in the notification header) specified in
    the Notification IRP: IS.
    */
    interface AttributeNameValue
    {
        const string NOTIFICATION_ID = "a";
        const string EVENT_TIME = "b";
        const string SYSTEM_DN = "c";
        const string MANAGED_OBJECT_CLASS = "d";
        const string MANAGED_OBJECT_INSTANCE = "e";
    };

    /*
    It defines the notification categories.
    A notification category is identified by the IRP name and its version number.
    */
    typedef ManagedGenericIRPConstDefs::VersionNumberSet NotificationCategorySet;

    /*
    It defines the notification types of a particular notification category.
    */
    typedef sequence <string> NotificationTypePerNotificationCategory;

    /*
    This sequence identifies all notification types of all notification
    categories identified by NotificationCategorySet. The number of elements
    in this sequence shall be identical to that of NotificationCategorySet.
    */
    typedef sequence <NotificationTypePerNotificationCategory>
        NotificationTypesSet;

    /*
    It defines a sequence of SubscriptionIds.
    */
    typedef string SubscriptionId;
    typedef sequence <SubscriptionId> SubscriptionIdSet;

    /*
    This indicates if the subscription is Active (not suspended), Suspended,
    or Invalid.
    */
    enum SubscriptionState {Active, Suspended, Invalid};
};
#endif

```

## A.4 IDL specification (file name "NotificationIRPSystem.idl")

```

#ifndef NotificationIRPSystem_idl
#define NotificationIRPSystem_idl

#include "CosNotifyChannelAdmin.idl"
#include "ManagedGenericIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"
#include "NotificationIRPConstDefs.idl"

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

/* ## Module: NotificationIRPSystem
This module implements capabilities of Notification IRP.
=====
*/
module NotificationIRPSystem
{
    /*
    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 GetNotificationIRPVersions { string reason; };
    exception GetNotificationIRPOperationsProfile { string reason; };
    exception GetNotificationIRPNotificationProfile { string reason; };
    exception Attach { string reason; };
    exception DetachException { string reason; };
    exception GetSubscriptionStatus { string reason; };
    exception ChangeSubscriptionFilter { string reason; };
    exception GetNotificationCategories { string reason; };
    exception SuspendSubscription { string reason; };
    exception ResumeSubscription { string reason; };
    exception GetSubscriptionIds { string reason; };

    exception AlreadySubscribed {};
    exception AtLeastOneNotificationCategoryNotSupported {};

    interface NotificationIRP
    {
        /*
        Return the list of all supported Notification IRP versions
        Each IRPVersion is defined by the rule in TS 32.311 clause titled
        "IRP document version number string"–
        */
        ManagedGenericIRPConstDefs::VersionNumberSet get_notification_IRP_versions
        (
        )
        raises (GetNotificationIRPVersions);

        /*
        Return the list of all supported operations and their supported
        parameters for a specific Notification IRP version.
        */
        ManagedGenericIRPConstDefs::MethodList
        get_notification_IRP_operations_profile (
            in ManagedGenericIRPConstDefs::VersionNumber
            notification_irp_version
        )
        raises (GetNotificationIRPOperationsProfile,
            ManagedGenericIRPSystem::OperationNotSupported,
            ManagedGenericIRPSystem::InvalidParameter);

        /*
        Return the list of all supported notifications.
        Agent should always throw a ManagedGenericIRPSystem::OperationNotSupported
        exception.
        Similar method, such as get_alarm_IRP_notification_profile,
        is supported in other IRP versions such as Alarm IRP.
        */
        ManagedGenericIRPConstDefs::MethodList
        get_notification_IRP_notification_profile (
            in ManagedGenericIRPConstDefs::VersionNumber
            notification_irp_version
        )
    }
}

```

```

raises (GetNotificationIRPNotificationProfile,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Obtain the list of all supported notification categories.
*/
NotificationIRPConstDefs::NotificationCategorySet
    get_notification_categories (
        out NotificationIRPConstDefs::NotificationTypesSet
        notification_type_list
    )
raises (GetNotificationCategories,
        ManagedGenericIRPSystem::OperationNotSupported);

NotificationIRPConstDefs::SubscriptionId attach_push (
    in string manager_reference,
    in unsigned long time_tick,
    in NotificationIRPConstDefs::NotificationCategorySet
    notification_categories,
    in ManagedGenericIRPConstDefs::StringTypeOpt filter
)
raises (Attach, ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter, AlreadySubscribed,
        AtLeastOneNotificationCategoryNotSupported);

NotificationIRPConstDefs::SubscriptionId attach_push_b (
    in string manager_reference,
    in unsigned long time_tick,
    in NotificationIRPConstDefs::NotificationCategorySet
    notification_categories,
    in ManagedGenericIRPConstDefs::StringTypeOpt filter,
    out CosNotifyChannelAdmin::SequenceProxyPushSupplier system_reference
)
raises (Attach, ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter,
        AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported);

NotificationIRPConstDefs::SubscriptionId attach_pull (
    in string manager_reference,
    in unsigned long time_tick,
    in NotificationIRPConstDefs::NotificationCategorySet
    notification_categories,
    in ManagedGenericIRPConstDefs::StringTypeOpt filter,
    out CosNotifyChannelAdmin::SequenceProxyPullSupplier system_reference
)
raises (Attach, ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter,
        AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported);

/*
Replace the present filter constraint with the one provided.
*/
void change_subscription_filter (
    in NotificationIRPConstDefs::SubscriptionId subscription_id,
    in string filter
)
raises (ChangeSubscriptionFilter,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Check the current state of the subscription.
*/
NotificationIRPConstDefs::NotificationCategorySet get_subscription_status
(
    in NotificationIRPConstDefs::SubscriptionId subscription_id,
    out ManagedGenericIRPConstDefs::StringTypeOpt filter_in_effect,
    out NotificationIRPConstDefs::SubscriptionState subscription_state,
    out unsigned long time_tick
)
raises (GetSubscriptionStatus,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

NotificationIRPConstDefs::SubscriptionIdSet get_subscription_ids (
    in string manager_reference
)
raises (GetSubscriptionIds,
        ManagedGenericIRPSystem::OperationNotSupported,

```



```
        ManagedGenericIRPSystem::InvalidParameter);

    /*
    Suspends the event flow until a resume is issued.
    */
    void suspend_subscription (
        in NotificationIRPConstDefs::SubscriptionId subscription_id
    )
    raises (SuspendSubscription,
        ManagedGenericIRPSystem::OperationNotSupported);
        ManagedGenericIRPSystem::InvalidParameter);

    /*
    Resumes the event flow if it was suspended.
    */
    void resume_subscription (
        in NotificationIRPConstDefs::SubscriptionId subscription_id
    )
    raises (ResumeSubscription,
        ManagedGenericIRPSystem::OperationNotSupported);
        ManagedGenericIRPSystem::InvalidParameter);

    /*
    Terminates the subscription with the agent.
    */
    void detach (
        in string manager_reference,
        in NotificationIRPConstDefs::SubscriptionId subscription_id
    )
    raises (DetachException);
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);
};

#endif
```

## CHANGE REQUEST

# **32.111-3 CR 026** # rev **-** # Current version: **5.2.0** #

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

**Proposed change affects:** UICC apps  ME  Radio Access Network  Core Network

<b>Title:</b>	# Correction of CORBA_ALARM_IRP_VERSION in line with adopted Rel-5 policy		
<b>Source:</b>	# S5		
<b>Work item code:</b>	# OAM-NIM	<b>Date:</b>	# 28/02/2003
<b>Category:</b>	# <b>A</b>	<b>Release:</b>	# Rel-5
	Use <u>one</u> of the following categories:		Use <u>one</u> of the following releases:
	<b>F</b> (correction)		2 (GSM Phase 2)
	<b>A</b> (corresponds to a correction in an earlier release)		R96 (Release 1996)
	<b>B</b> (addition of feature),		R97 (Release 1997)
	<b>C</b> (functional modification of feature)		R98 (Release 1998)
	<b>D</b> (editorial modification)		R99 (Release 1999)
	Detailed explanations of the above categories can be found in 3GPP <a href="#">TR 21.900</a> .		Rel-4 (Release 4)
			Rel-5 (Release 5)
			Rel-6 (Release 6)

<b>Reason for change:</b>	# The interface version definition does not meet the format rule in 32.111-3.
<b>Summary of change:</b>	# The NOTIFICATION_IRP_VERSION constant definition is to be deleted, and behavioural comments added to clarify the behaviour of get_notification_IRP_versions. This aligns interface version handling with the release 5 approach used for 32.603 release 5.
<b>Consequences if not approved:</b>	# Systems using the original NOTIFICATION_IRP_VERSION constant will be incompatible with systems using the documented rules for defining the version string.

<b>Clauses affected:</b>	# Annex A.1 – delete the NOTIFICATION_IRP_VERSION constant. Annex A.2 – add comments to define the desired behaviour of operation get_notification_IRP_versions								
<b>Other specs affected:</b>	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> <tr> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> </table> Other core specifications # <input type="checkbox"/> Test specifications # <input type="checkbox"/> O&M Specifications # <input type="checkbox"/>	Y	N	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Y	N								
<input type="checkbox"/>	<input checked="" type="checkbox"/>								
<input type="checkbox"/>	<input checked="" type="checkbox"/>								
<input type="checkbox"/>	<input checked="" type="checkbox"/>								
<b>Other comments:</b>	# Rel-5 Mirror CR of 32111-3CR025 S5-036121.								

**How to create CRs using this form:**

## 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 COMMERCIAL_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";
    const string CLEAR_SYSTEM_ID = "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
```

## A.2 IDL specification (file name "AlarmIRPSystem.idl")

```

#ifndef AlarmIRPSystem_idl
#define AlarmIRPSystem_idl

#include "AlarmIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"

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

/* ## 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 ClearAlarms { 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.

        -
        Implementations are to provide a return value consisting of one or more IRPVersions.
        Each IRPVersion is defined by the rule in the clause titled

```

"IRP document version number string"

```

*/
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::AlarmInformationIdAndSevSeq
    alarm_information_id_and_sev_list,
    in string ack_user_id,
    in ManagedGenericIRPConstDefs::StringTypeOpt ack_system_id,
    out AlarmIRPConstDefs::BadAcknowledgeAlarmInfoSeq
    bad_ack_alarm_info_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 ManagedGenericIRPConstDefs::StringTypeOpt 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 ManagedGenericIRPConstDefs::StringTypeOpt comment_system_id,
    in string comment_text,
    out AlarmIRPConstDefs::BadAlarmInformationIdSeq
    bad_alarm_information_id_list
)
raises (CommentAlarms, ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

```

```

/*
Request to clear one or more alarms.
*/
ManagedGenericIRPConstDefs::Signal clear_alarms (
    in AlarmIRPConstDefs::AlarmInformationIdSeq alarm_information_id_list,
    in string clear_user_id,
    in string clear_system_id,
    out AlarmIRPConstDefs::BadAlarmInformationIdSeq
        bad_alarm_information_id_list
)
raises (ClearAlarms, 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 ManagedGenericIRPConstDefs::StringTypeOpt 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 ManagedGenericIRPConstDefs::StringTypeOpt 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

```