

# ETSI TS 132 106-3 V3.2.0 (2000-12)

---

*Technical Specification*

**Universal Mobile Telecommunications System (UMTS);  
Telecommunication Management;  
Configuration Management;  
Part 3: Notification Integration  
Reference Point: CORBA solution set version 1:1  
(3GPP TS 32.106-3 version 3.2.0 Release 1999)**

---



---

**Reference**

RTS/TSGS-0532106-3UR

---

**Keywords**

UMTS

***ETSI***

---

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

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

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

---

***Important notice***

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at <http://www.etsi.org/tb/status/>

If you find errors in the present document, send your comment to:  
editor@etsi.fr

---

***Copyright Notification***

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2000.

All rights reserved.

---

## Intellectual Property Rights

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

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

---

## Foreword

This Technical Specification (TS) has been produced by the ETSI 3<sup>rd</sup> 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](http://www.etsi.org/key) .

---

# Contents

Foreword.....	4
Introduction .....	4
1 Scope .....	6
2 References .....	6
3 Definitions and abbreviations.....	6
3.1 Definitions.....	6
3.2 Abbreviations .....	6
4 Architectural features .....	7
4.1 Notification services.....	7
4.1.1 Support of Push and Pull Interface .....	7
4.1.2 Support of multiple notifications in one push operation.....	7
5 Mapping.....	8
5.1 Operation mapping.....	8
5.2 Operation parameter mapping.....	9
5.3 Notification parameter mapping.....	12
5.4 Attribute mapping .....	12
6 Use of OMG Notification StructuredEvent .....	12
7 IRPAgent's Behaviour .....	15
7.1 Subscription .....	15
7.2 IRPAgent supports multiple categories of Notifications.....	15
7.3 IRPAgent's integrity risk of attach_push_b Method.....	15
8 Example of Notification related to alarm .....	16
<b>Annex A (normative): Notification IRP CORBA IDL.....</b>	<b>17</b>
<b>Annex B (informative): Change history .....</b>	<b>22</b>

---

## Foreword

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

The present document is part 3 of a multi-part TS covering the 3<sup>rd</sup> Generation Partnership Project: Technical Specification Group Services and System Aspects; Telecommunication Management; Configuration Management, as identified below:

Part 1: “3G Configuration Management: Concept and Requirements”;

Part 2: “Notification Integration Reference Point: Information Service Version 1”;

**Part 3: “Notification Integration Reference Point: CORBA Solution Set Version 1:1”;**

Part 4: “Notification Integration Reference Point: CMIP Solution Set Version 1:1”;

Part 5: “Basic Configuration Management IRP Information Model (including NRM) Version 1”;

Part 6: “Basic Configuration Management IRP CORBA Solution Set Version 1:1”;

Part 7: “Basic Configuration Management IRP CMIP Solution Set Version 1:1”;

Part 8: “Name Convention for Managed Objects”.

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

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

---

## Introduction

Configuration Management (CM), in general, provides the operator with the ability to assure correct and effective operation of the 3G network as it evolves. CM actions have the objective to control and monitor the actual configuration on the Network Elements (NEs) and Network Resources (NRs), and they may be initiated by the operator or by functions in the Operations Systems (OSs) or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality Of Service (QOS). The CM actions are initiated either as a single action on a NE of the 3G network or as part of a complex procedure involving actions on many NEs.

The Itf-N interface for CM is built up by a number of Integration Reference Points (IRPs) and a related Name Convention, which realise the functional capabilities over this interface. The basic structure of the IRPs is defined in ITU-T Recommendation X.736 [1] and OMG TC Document telecom [2]. For CM, a number of IRPs (and the Name

Convention) are defined herein, used by this as well as other specifications for Telecom Management (TM) produced by 3GPP. All these are included in 3GPP TS 32.106 from Part 2 and onwards.

The present document is Part 3 of 3GPP TS 32.106 (3GPP TS 32.106-3) - Notification IRP CORBA Solution Set.

**IRP Solution Set version:** The version of this CORBA Solution Set is 1:1, where the first “1” means that it corresponds to the Information Service (3GPP TS 32.106-2 [5]) version 1, and the second “1” means that it is the first CORBA Solution Set corresponding to this Information Service version.

---

# 1 Scope

The present document specifies the Common Object Request Broker Architecture (CORBA) Solution Set (SS) for the IRP whose semantics is specified in Notification IRP: Information Service (3GPP TS 32.106-2 [5]).

---

## 2 References

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

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

- [1] ITU-T Recommendation X.736: "Security Alarm Reporting Function".
  - [2] OMG TC Document telecom (98-11-01): "OMG Notification Service".
  - [3] OMG CORBA services: Common Object Services Specification, Update: November 22, 1996. (Clause 4 contains the Event Service Specification.)
  - [4] 3GPP TS 32.106-8: "Name Convention for Managed Objects".
  - [5] 3GPP TS 32.106-2: "Notification IRP: Information Service".
  - [6] 3GPP TS 32.111-2: "Alarm IRP: Information Service".
  - [7] 3GPP TS 32.111-3: "Alarm IRP: CORBA Solution Set, version 1:1".
  - [8] ITU-T Recommendation X.733: "Alarm Reporting function".
  - [9] 3GPP TS 32.101: "3G Telecom Management principles and high level requirements".
  - [10] 3GPP TS 32.102: "3G Telecom Management architecture".
  - [11] 3GPP TS 32.106-1: "3G Configuration Management: Concept and Requirements".
- 

## 3 Definitions and abbreviations

### 3.1 Definitions

For the purposes of the present document, the following terms and definitions apply: Please refer to 3GPP TS 32.101 [9], 3GPP TS 32.102 [10] and 3GPP TS 32.106-1 [11].

### 3.2 Abbreviations

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

CM	Configuration Management
CORBA	Common Object Request Broker Architecture (OMG)
EC	Event channel (OMG)
IDL	Interface Definition Language (OMG)
IS	Information Service

NC	Notification Channel (OMG)
NE	Network Element
NV	Name and Value pair
EM	Element Manager
OMG	Object Management Group
SS	Solution Set
UML	Unified Modelling Language (OMG)

## 4 Architectural features

The overall architectural feature of Notification IRP is specified in 3GPP TS 32.106-2 [5]. This clause specifies features that are specific to the CORBA Solution Set (SS).

### 4.1 Notification services

In the CORBA Solution Set, notifications are emitted by IRPAgent using CORBA Notification service (OMG TC Document telecom [2]).

CORBA Event service (OMG CORBA services [3]) provides event routing and distribution capabilities. CORBA Notification service provides, in addition to Event service, event filtering and support for Quality Of Service (QOS) as well.

A subset of CORBA Notification services shall be used to support the implementation of notification. This CORBA Notification service subset, in terms of OMG Notification service (OMG TC Document telecom [2]) defined methods, is identified in the present.

#### 4.1.1 Support of Push and Pull Interface

The IRPAgent shall support the OMG Notification push interface model. Additionally, it may support the OMG Notification pull interface model as well.

#### 4.1.2 Support of multiple notifications in one `push` operation

For efficiency, IRPAgent uses the following OMG Notification Service (OMG TC Document telecom [2]) defined interface to pack multiple notifications and push them to IRPManager using one method `push_structured_events`. The method takes as input a parameter of type `EventBatch` as defined in the `OMG CosNotification` module (OMG TC Document telecom [2]). This data type is a sequence of Structured Events (see clause 4). Upon invocation, this parameter will contain a sequence of Structured Events being delivered to IRPManager by IRPAgent to which it is connected.

The maximum number of events that will be transmitted within a single invocation of this operation is controlled by IRPAgent wide configuration parameter. The amount of time IRPAgent will accumulate individual events into the sequence before invoking this operation is controlled by IRPAgent wide configuration parameter as well.

IRPAgent may push `EventBatch` with only one Structured Event.

The OMG Notification service (OMG TC Document telecom [2]) defined IDL module is shown below.

```
module CosNotifyComm {
    ...
    Interface SequencePushConsumer : NotifyPublish {
        void push_structured_events(
            in CosNotification::EventBatch notifications)
            raises( CosEventComm::Disconnected);
        ...
    }; // SequencePushConsumer
    ...
}; // CosNotifyComm
```



## 5 Mapping

### 5.1 Operation mapping

Notification IRP: IS (3GPP TS 32.106-2 [5]) defines semantics of operations visible across this IRP.

Table 1 maps the operations defined in Notification IRP: IS (3GPP TS 32.106-2 [5]) to their equivalents (methods) in this Solution Set (SS). It also qualifies if a method is Mandatory (M) or Optional (O)

**Table 1: Mapping from IS Operation to SS Equivalents**

IS Operations in 3GPP TS 32.106-2 [5]	SS Methods	Qualifier
subscribe	attach_push, attach_push_b, attach_pull	M, O, O
unsubscribe	detach	M
get Notification IRPVersion	get_notification_IRP_version	M
get Subscription Status	get_subscription_status	O
getSubscriptionIds	get_subscription_ids	O
change Subscription Filter	<p>If subscription is established using attach_push method, the SS equivalent shall be change_subscription_filter. The IDL specification of this method is included in Annex A. This method is Optional (O).</p> <p>If subscription is established using attach_push_b method, the SS equivalent shall be modify_constraints. The method is defined in OMG Notification Service Filter Interface (OMG TC Document telecom [2]). The IDL specification of this method is not included in Annex A. If IRPAgent supports the optional attach_push_b method, it shall support this method as mandatory.</p> <p>If subscription is established using attach_pull method, the SS equivalent shall be modify_constraints. The method is defined by OMG Notification Service Filter Interface (OMG TC Document telecom [2]). The IDL specification of this method is not included in Annex A. If IRPAgent supports the optional attach_pull method, it shall support this method as mandatory.</p>	See box on the left.
suspend Subscription	<p>If subscription is established using attach_push, there is no SS equivalent. In other words, IRPManager cannot suspend subscription.</p> <p>If subscription is established using attach_push_b, the SS equivalent shall be suspend_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]). The IDL specification of this method is not included in Annex A. If IRPAgent supports the optional attach_push_b method, it shall support this method as mandatory.</p> <p>If subscription is established using attach_pull, there is no SS equivalent.</p>	See box on the left

resume Subscription	<p>If subscription is established using <code>attach_push</code>, there is no SS equivalent. In other words, IRPManager cannot resume subscription.</p> <p>If subscription is established using <code>attach_push_b</code>, the SS equivalent shall be <code>resume_connection</code>. This method is defined by OMG Notification Service (OMG TC Document telecom [2]). The IDL specification of this method is not included in Annex A. If IRPAgent supports the optional <code>attach_push_b</code> method, it shall support this method as mandatory.</p> <p>If subscription is established using <code>attach_pull</code>, there is no SS equivalent.</p>	See box on the left
get Notification Categories	<code>get_notification_categories</code>	O

## 5.2 Operation parameter mapping

3GPP TS 32.106-2 [5] defines semantics of parameters carried in operations across the Notification IRP. Table 2 through table 12 indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

**Table 2: Mapping from IS subscribe parameters to SS attach\_push equivalents**

IS Operation parameter	SS Method parameter	Qualifier
<code>managerReference</code>	Object <code>manager_reference</code>	M
<code>timeTick</code>	long <code>time_tick</code>	O
<code>notificationCategories</code>	<code>NotificationIRPConstDefs::NotificationCategorySet</code> <code>notification_category_set</code>	O
<code>filter</code>	string <code>filter</code> (See NOTE)	O
<code>subscriptionId</code>	Return value of type <code>SubscriptionId</code>	M
<code>status</code>	<code>Attach</code> , <code>ParameterNotSupported</code> , <code>InvalidParameter</code> , <code>AlreadySubscribed</code> , <code>AtLeastOneNotificationCategoryNotSupported</code>	M
NOTE:	The grammar of the filter string is <code>extended_TCL</code> defined by OMG Notification Service (OMG TC Document telecom [2]). This grammar shall be the only one used for Alarm IRP: CORBA SS.	

**Table 3: Mapping from IS subscribe parameters to SS attach\_push\_b equivalents**

IS Operation parameter	SS Method parameter	Qualifier
<code>managerReference</code>	Object <code>manager_reference</code>	M
<code>timeTick</code>	long <code>time_tick</code>	O
<code>notificationCategories</code>	<code>NotificationIRPConstDefs::NotificationCategorySet</code> <code>notification_category_set</code>	O
<code>filter</code>	string <code>filter</code>	O
<code>subscriptionId</code>	Return value of type <code>SubscriptionId</code>	M
Not specified in IS	<code>CosNotifyChannelAdmin::SequenceProxyPushSupplier</code> <code>system_reference</code> (See NOTE)	M
<code>status</code>	<code>Attach</code> , <code>OperationNotSupported</code> , <code>ParameterNotSupported</code> , <code>InvalidParameter</code> , <code>AlreadySubscribed</code> , <code>AtLeastOneNotificationCategoryNotSupported</code>	M
NOTE:	IRPAgent provides this reference to which IRPManager can invoke methods to manage the subscription. Valid methods are not defined in this IRP. OMG CORBA Notification Service defines these methods. Read interface <code>SequencePushSupplier:proxySupplier</code> , <code>CosNotifyComm::SequencePushSupplier</code> {}. IRPManager is expected to invoke <code>connect_sequence_push_consumer()</code> of this interface to connect its own <code>cosNotifyComm::sequencePushConsumer</code> with this reference. After successful connection, IRPAgent pushes sequence of Structured Events towards IRPManager.	

**Table 4: Mapping from IS subscribe parameters to SS attach\_pull equivalents**

IS Operation parameter	SS Method parameter	Qualifier
managerReference	Object manager_reference	M
timeTick	long time_tick	O
notificationCategories	NotificationIRPConstDefs::NotificationCategorySet notification_category_set	O
filter	string filter	O
subscriptionId	Return value of type SubscriptionId	M
Not specified in IS.	CosNotifyChannelAdmin::SequenceProxyPullSupplier system_reference	M
status	Attach, OperationNotSupported, ParameterNotSupported, InvalidParameter, AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported	M

**Table 5: Mapping from IS unsubscribe parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
managerReference	Object manager_reference	M
subscriptionId	string subscription_id	O
status	Detach, InvalidParameter	M

**Table 6: Mapping from IS getNotificationIRPVersion parameters to SS equivalents**

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

**Table 7: Mapping from IS getSubscriptionStatus parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscription_id	M
notificationCategoryList	Return value of type NotificationIRPConstDefs::NotificationCategorySet	M
filterInEffect	string filter_in_effect	O
subscriptionState	NotificationIRPConstDef::SubscriptionState subscription_state	O
timeTick	long time_tick	O
status	GetSubscriptionStatus, OperationNotSupported, InvalidParameter	M

**Table 8: Mapping from IS getSubscriptionIds parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
managerReference	Object manager_reference	M
subscriptionIdList	Return value of type NotificationIRPConstDefs::SubscriptionIdSet	M
status	GetSubscriptionIds, OperationNotSupported, InvalidParameter	M

**Table 9: Mapping from IS changeSubscriptionFilter parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscription_id	M
filter	string filter	M
status	ChangeSubscriptionFilter, OperationNotSupported, InvalidParameter	M

**Table 10: Mapping from IS suspendSubscription parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	If subscription is established using attach_push, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter. If subscription is established using attach_push_b, the SS equivalent method is suspend_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]) and requires no parameter. Therefore, there is no SS equivalent for this IS parameter. If subscription is established using attach_pull, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.	M
status	If subscription is established using attach_push, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter. If subscription is established using attach_push_b, the SS equivalent method is suspend_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]) and it returns a void. Therefore, there is no SS equivalent for this IS parameter. This suspend_connection method can raise OMG Notification Service (OMG TC Document telecom [2]) defined exception called ConnectionAlreadyInactive. If subscription is established using attach_pull, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.	M

**Table 11: Mapping from IS resumeSubscription parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	If subscription is established using attach_push, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter. If subscription is established using attach_push_b, the SS equivalent method is resume_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]) and requires no parameter. Therefore, there is no SS equivalent for this IS parameter. If subscription is established using attach_pull, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.	M
status	If subscription is established using attach_push, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter. If subscription is established using attach_push_b, the SS equivalent method is resume_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]) and returns a void. Therefore, there is no SS equivalent for this IS parameter. This resume_connection method can raise OMG Notification Service (OMG TC Document telecom [2]) defined exception called ConnectionAlreadyActive. If subscription is established using attach_pull, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.	M

**Table 12: Mapping from IS getNotificationCategories parameters to SS equivalents**

IS Operation parameter	SS Method parameter	Qualifier
notification CategoryList	Return value of type NotificationIRPConstDefs::NotificationCategorySet	M
eventTypeList	NotificationIRPConstDefs::EventTypesSet event_type_list	O
extendedEvent TypeList	NotificationIRPConstDefs::ExtendedEventTypesSet extended_event_type_list	O
status	GetNotificationCategories, OperationNotSupported	M

## 5.3 Notification parameter mapping

Notification IRP: IS (3GPP TS 32.106-2 [5]) defines a generic `notify` and its parameters. This SS does not provide the mapping of these parameters to their CORBA SS equivalents.

Other IRPs using the Notification IRP such as Alarm IRP: IS (3GPP TS 32.111-2 [6]) extend the generic `notify` for their specific use. Their corresponding SS documents shall define the mapping from their specific notification parameters (defined in their IS document) to their SS equivalents. These SS documents shall qualify their SS equivalents as well.

## 5.4 Attribute mapping

Notification IRP: IS (3GPP TS 32.106-2 [5]) defines the semantics of common attributes carried in notifications. This SS does not provide the mapping of these attributes to their CORBA SS equivalents. Other IRPs such as Alarm IRP: IS (3GPP TS 32.111-2 [6]) identify and qualify these common attributes for use in their environment. Their corresponding SS documents define the mapping of these attributes to their SS equivalents.

---

# 6 Use of OMG Notification StructuredEvent

Notification IRP: IS (3GPP TS 32.106-2 [5]) defines attributes that are commonly present in notifications of all notification categories such as notifications emitted from Alarm IRP IRPAgent.

In CORBA SS, OMG defined `StructuredEvent` (OMG TC Document telecom [2]) is used to carry notification. This clause identifies the OMG defined `StructuredEvent` attributes that carry the common attributes defined in 3GPP TS 32.106-2 [5].

The composition of OMG `StructuredEvent` is:

```
Header
  Fixed Header
    Domain_name
    Type_name
    Event_name
  Variable Header
Body
  Filterable_body_fields
  Remaining_body
```

Table 13 shows the OMG Structured Event attributes (middle column) that are used to carry the common notification attributes defined in Notification IRP: IS (3GPP TS 32.106-2 [5]).

**Table 13: Attributes of StructuredEvent**

Common attributes defined in Notification IRP: IS (3GPP TS 32.106-2 [5])	Attribute defined by OMG Structured Event	Comment
There is no corresponding SS attribute.	domain_name	<p>It indicates that the StructuredEvent, carried in the Notification, is defined by a specific 3GPP IRP such as Alarm IRP, as opposed to OMG specified Telecommunication, healthcare, utility, finance, etc. It indicates the CORBA SS version number as well.</p> <p>It is a string. Legal values are defined in module.</p> <p>For Alarm IRP version 1:1, the value is ALARM_IRP_VERSION_1_1.</p>
eventType	type_name	<p>It indicates event types of this notification. The semantics of the event type is defined by ITU-T TMN Recommendations. Each IRP, such as Alarm IRP IS version 1, shall identify the ITU-T defined event types for their use. Each such IRP document shall define the values of the identified event Types as well.</p> <p>Dependent on the notification category, possible legal values are:</p> <p>COMMUNICATIONS_ALARM (clause 8.1.1 of ITU-T Recommendation X.736 [8]),            QUALITY_OF_SERVICE_ALARM (clause 8.1.1 of ITU-T Recommendation X.736 [8]),            PROCESSING_ERROR_ALARM (clause 8.1.1 of ITU-T Recommendation X.736 [8]),            EQUIPMENT_ALARM (clause 8.1.1 of ITU-T Recommendation X.736 [8]),            ENVIRONMENTAL_ALARM (clause 8.1.1 of ITU-T Recommendation X.736 [8]),            PHYSICAL_VIOLATION (ITU-T Recommendation X.736 [1]),            INTEGRITY_VIOLATION (ITU-T Recommendation X.736 [1]),            SECURITY_VIOLATION (ITU-T Recommendation X.736 [1]),            TIME_DOMAIN_VIOLATION (ITU-T Recommendation X.736 [1]),            OPERATIONAL_VIOLATION (ITU-T Recommendation X.736 [1]).</p> <p>The bracketed number of each type indicates the reference where the semantics of the type is specified.</p> <p>It is a string. See each individual CORBA SS IDL module for each IRP using the Notification IRP, for legal values used by that IRP version.</p> <p>Since each IRP except Notification IRP specifies its own set of event type, the values specified by each IRP are only unique within one IRP. For uniqueness among all IRPs' specifications, the values of event type shall be coupled with the notification category, the value carried in domain_name of the same notification.</p>
extended EventType	event_name	<p>The legal values carried in this attribute are specified by the IRP using the notification. For example, Alarm IRP: CORBA SS (3GPP TS 32.111-3 [7]) defines and uses the following values:</p> <p>NOTIFY_FM_NEW_ALARM,            NOTIFY_FM_CHANGED_ALARM,            NOTIFY_FM_ACK_STATE_CHANGED,            NOTIFY_FM_CLEARED_ALARM and            NOTIFY_FM_ALARM_LIST_REBUILT.</p> <p>It is a string. See each individual CORBA SS IDL module for each IRP using the Notification IRP, for legal values used by that IRP version.</p> <p>Since each IRP except Notification IRP specifies its own set of extended event type, the values specified by each IRP are only unique within one IRP. For uniqueness among all IRPs' specification, the values of extended event type shall</p>

		be coupled with the notification category, the value carried in domain_name of the same notification.
There is no corresponding SS attribute.	variable Header	
managed Object Class, managed Object Instance	One NV pair of filterable_body_fields (name-value)	Name of NV pair is a string, NV_MANAGED_OBJECT_INSTANCE. Value of NV pair is a string. Syntax and semantics of this string conform to the Managed Object string representation specified in (3GPP TS 32.106-8 [4]). Note that two SS attributes are carried in this one NV pair since the string representation specified in 3GPP TS 32.106-8 [4] can convey the semantics of managedObjectClass and managedObjectInstance in one string.
notification Id	One NV pair of filterable_body_fields	Name of NV pair is a string, NV_NOTIFICATION_ID. Value of NV pair is an unsigned long.
eventTime	One NV pair of filterable_body_fields	Name of NV pair is a string, NV_EVENT_TIME. Value of NV pair is an IRPTime.
systemDN	One NV pair of filterable_body_fields	Name of NV pair is a string, NV_SYSTEM_DN. Value of NV pair is a string. Syntax and semantics of this string conforms to the Managed Object string representation specified in 3GPP TS 32.106-8 [4].
There is no corresponding SS attribute.	remaining_Body	

---

## 7 IRPAgent's Behaviour

This clause describes some IRPAgent's behaviour not captured by IDL.

### 7.1 Subscription

IRPManager can invoke multiple `attach_push`, multiple `attach_push_b` or multiple `attach_pull` using different `manager_reference(s)`. As far as IRPAgent is concerned, the IRPAgent will emit notifications to multiple "places" with their independent filter requirements. IRPAgent will not know if the notifications are going to the same IRPManager.

If IRPManager invokes multiple `attach_push`, `attach_push_b` or `attach_pull` using the same `manager_reference` and with an already subscribed `notification_category`, IRPAgent shall raise `AlreadySubscribed` exception to all invocations except one.

IRPManager can invoke multiple `attach_push` using the same `manager_reference` and with one or more not-yet-subscribed `notification_categories`. In this case, if IRPAgent supports all the notification categories requested, IRPAgent shall accept the invocation; otherwise, it raises `AtLeastOneNotificationCategoryNotSupported` exception. IRPAgent shall have similar behaviour for `attach_push_b` and `attach_pull`.

When IRPManager is in subscription by invoking `attach_push`, IRPManager can change the filter constraint, using `change_subscription_filter`, applicable to the notification categories specified in the `attach_push`.

When IRPManager is in subscription by invoking `attach_push_b`, IRPManager can change the filter constraint during subscription using the OMG defined Notification Service Filter Interface. IRPManager shall not use `change_subscription_filter`; otherwise it shall get an exception.

### 7.2 IRPAgent supports multiple categories of Notifications

IRPAgent may emit multiple categories of Notifications. IRPAgent may have mechanism for IRPManager to pull for notifications of multiple categories.

IRPManager can query IRPAgent about the categories of notifications supported by using `get_notification_categories`.

IRPManager uses a parameter, `notification_categories`, in `attach_push`, `attach_push_b` and `attach_pull` to specify one or more categories of notifications wanted.

IRPManager uses a zero-length sequence in `notification_categories` of `attach_push`, `attach_push_b` and `attach_pull` to specify that all IRPAgent supported categories of notifications are wanted. If IRPManager uses `attach_push` with zero-length sequence in `notification_categories` and if the operation is successful, IRPAgent shall reject subsequent `attach_push` operation, regardless if the `notification_categories` contains a zero-length sequence or one or more specific notification categories. IRPAgent shall have similar behaviour for `attach_push_b` and `attach_pull`.

### 7.3 IRPAgent's integrity risk of `attach_push_b` Method

In the case that IRPAgent implements this method by extending or using OMG compliant Notification Service, the following IRPManager behaviour illustrates a risk to IRPAgent's integrity.

Given the object reference (IOR) of the `SequenceProxyPushSupplier` (as the mandatory output parameter of the subject method), IRPManager can invoke `sequenceProxyPushSupplier.MyAdmin` method.

IRPManager can then obtain the consumer admin object of the proxy. Then IRPManager can invoke `consumerAdmin.MyChannel` to get the IOR of the Notification Channel. IRPManager then can call



`eventChannel.MyFactory` which will provide `IRPManager` the IOR of the `EventChannelFactory` itself. `IRPManager` can then be able to invoke methods directly on the `EventChannelFactory`, like `get_all_channels` which lists all channel numbers and `create_channel` which allows `IRPManager` to create any number of additional channels.

A malicious `IRPManager` can, given access to the `EventChannelFactory`, get a list of existing channels and start connecting them together at random thus compromising the `IRPAgent`'s integrity. Deployment of this `attach_push_b` needs strong authentication and authorisation mechanism in place.

`attach_push` is mandatory. `IRPAgent` compliant to this IRP shall implement it.

`attach_push_b` is optional. It is recommended that `IRPAgent` concerned with integrity risk should not implement the `attach_push_b` option.

---

## 8 Example of Notification related to alarm

The following is an example of Notification related to alarm.

If `type_name == NOTIFY_FM_NEW_ALARM`, then the `filterable_body_field` attributes can contain:

```
{
    systemDN, "...";
    alarmId, "abce232",
        notificationId, 4467,
    managedObjectInstance, "...",
        eventTime, ...,
    probableCause, 3,
    perceivedSeverity, 2,
    specificProblems, "xxx",
    additionalText, "...",
    ...
}
```

---

## Annex A (normative): Notification IRP CORBA IDL

```
/* ## Module: CommonIRPConstDefs
This module contains definitions commonly used among all IRPs such as Alarm IRP.
=====
*/

#ifndef CommonIRPConstDefs_idl
#define CommonIRPConstDefs_idl
#include <TimeBase.idl>
#pragma prefix "3gpssa5.org"

module CommonIRPConstDefs {

    /*
    Definition imported from CosTime. The time refers to time in Greenwich
    Time Zone. It also consists of a time displacement factor in the form
    of minutes of displacement from the Greenwich Meridian.
    */
    typedef TimeBase::UtcT IRPTime;

    enum Signal {OK, Failure, PartialFailure};

    typedef sequence <string> VersionNumberSet;

};

#endif

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

#ifndef NotificationIRPConstDefs_idl
#define NotificationIRPConstDefs_idl
#pragma prefix "3gpssa5.org"

module NotificationIRPConstDefs {

    /*
    This is a string sequence identifying notification categories.
    A notification category is identified by the IRP name and its version.
    */
    typedef sequence <string> NotificationCategorySet;

    /*
    This is a sequence of strings identifying event types of a particular
    notification category.
    */
    typedef sequence <string> EventTypesPerNotificationCategory;

    /*
    This sequence identifies all event types of all notification categories
    identified by NotificationCategorySet. The number of elements in this
    sequence shall be identical to that of NotificationCategorySet.
    */
};

}
```

```

*/
typedef sequence <EventTypesPerNotificationCategory> EventTypesSet;

/*
This is a sequence of strings identifying extended event types of
a particular notification category.
*/
typedef sequence <string> ExtendedEventTypePerNotificationCategory;

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

typedef sequence <long> NotifIDSetType;

/*
This holds identifiers of notifications that are correlated.
*/

struct CorrelatedNotification {
    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 parameter-attribute
                  // of the Structured Event, i.e., the notification.
    NotifIDSetType notifIDSet;
};

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

/*
This is a sequence of strings identifying Subscription Ids.
*/
typedef string SubscriptionId;
typedef sequence <SubscriptionId> SubscriptionIdSet;

/*
This block encapsulates valid strings carried in domain_name of
structured event header. It carries the name of IRP and its
corresponding CORBA SS version number. They are the returned
values for get_XXX_IRP_version() as well.
*/
const string ALARM_IRP_VERSION_1_1 = "1f1"; //alarm IRP 1:1
const string CONFIGURATION_IRP_VERSION_1_1 = "1c1"; //CM IRP 1:1

/*
This string is used as return value for get_notification_irp_version()
*/
const string NOTIFICATION_IRP_VERSION_1_1 = "1n1"; //Notification IRP 1:1

```

```

/*
This block encapsulates string used in the name of the Name Value
pair of the structured event.
*/

const string NV_NOTIFICATION_ID = "a";
const string NV_CORRELATED_NOTIFICATIONS = "b";
const string NV_EVENT_TIME = "c";
const string NV_SYSTEM_DN = "d";
const string NV_MANAGED_OBJECT_CLASS = "e";
const string NV_MANAGED_OBJECT_INSTANCE = "f";
const string NV_PROBABLE_CAUSE = "g";
const string NV_PERCEIVED_SEVERITY = "h";
const string NV_SPECIFIC_PROBLEM = "i";
const string NV_ADDITIONAL_TEXT = "j";
const string NV_ALARM_ID = "k";
const string NV_ACK_USER_ID = "l";
const string NV_ACK_TIME = "m";
const string NV_ACK_SYSTEM_ID = "n";
const string NV_ACK_STATE = "o";
const string NV_BACKED_UP_STATUS = "p";
const string NV_BACK_UP_OBJECT = "q";
const string NV_THRESHOLD_INFO = "r";
const string NV_TREND_INDICATION = "s";
const string NV_STATE_CHANGE_DEFINITION = "t";
const string NV_MONITORED_ATTRIBUTES = "u";
const string NV_PROPOSED_REPAIR_ACTIONS = "v";

/*
This indicates if the subscription is active (not suspended) or inactive.
*/
enum SubscriptionState {Inactive, Active, DontKnow};

};

#endif

/* ## Module: NotificationIRPSystem
This module implements capabilities of IRPAgent specified in Notification
IRP: Information Service version 1 and its equivalents in Notification
IRP: CORBA Solution Set version 1:1.
=====
*/

#ifndef NotificationIRPSystem_idl
#define NotificationIRPSystem_idl

#include "CosNotifyComm.idl"
#include "CosNotifyChannelAdmin.idl"
#include "NotificationIRPConstDefs.idl"
#include "CommonIRPConstDefs.idl"
#pragma prefix "3gppsa5.org"

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 Attach { string reason; };
exception DetachException { string reason; };

exception GetSubscriptionStatus { string reason; };
exception GetSubscriptionIds { string reason; };
exception ChangeSubscriptionFilter { string reason; };
exception GetNotificationCategories { string reason; };
exception GetNotificationIRPVersion { 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 AlreadySubscribed {};
exception AtLeastOneNotificationCategoryNotSupported {};

interface NotificationIRPOperations {

/* ## Operation: attach_push
*/
NotificationIRPConstDefs::SubscriptionId attach_push (
    in Object manager_reference,
    in long time_tick,
    in NotificationIRPConstDefs::NotificationCategorySet
        notification_category_set,
    in string filter
)
raises (Attach, ParameterNotSupported, InvalidParameter, AlreadySubscribed,
        AtLeastOneNotificationCategoryNotSupported);

/* ## Operation: attach_push_b
*/
NotificationIRPConstDefs::SubscriptionId attach_push_b (
    in Object manager_reference,
    in long time_tick,
    in NotificationIRPConstDefs::NotificationCategorySet
        notification_category_set,
    in string filter,
    out CosNotifyChannelAdmin::SequenceProxyPushSupplier system_reference
)
raises
(Attach, OperationNotSupported, ParameterNotSupported, InvalidParameter, AlreadySubscribed,
AtLeastOneNotificationCategoryNotSupported);

/* ## Operation: attach_pull
*/
NotificationIRPConstDefs::SubscriptionId attach_pull (
    in Object manager_reference,
    in long time_tick,
    in NotificationIRPConstDefs::NotificationCategorySet
        notification_category_set,
    in string filter,
    out CosNotifyChannelAdmin::SequenceProxyPullSupplier system_reference
)
raises (Attach, OperationNotSupported, ParameterNotSupported,
        InvalidParameter, AlreadySubscribed,

```

```
        AtLeastOneNotificationCategoryNotSupported);

/* ## Operation: detach
*/
void detach (
    in Object manager_reference,
    in string subscription_id
)
raises (DetachException,InvalidParameter);

/* ## Operation: get_notification_IRP_version
*/
CommonIRPConstDefs::VersionNumberSet get_notification_IRP_version ()
raises (GetNotificationIRPVersion);
;

/* ## Operation: get_subscription_status
*/
NotificationIRPConstDefs::NotificationCategorySet get_subscription_status (
    in string subscription_id,
    out string filter_in_effect,
    out NotificationIRPConstDefs::SubscriptionState subscription_state,
    out long time_tick
)
raises (GetSubscriptionStatus,OperationNotSupported,InvalidParameter);

/* ## Operation: get_subscription_ids
*/
NotificationIRPConstDefs::SubscriptionIdSet get_subscription_ids (
    in Object manager_reference
)
raises (GetSubscriptionIds,OperationNotSupported,InvalidParameter);

/* ## Operation: change_subscription_filter
*/
void change_subscription_filter (
    in string subscription_id,
    in string filter
)
raises (ChangeSubscriptionFilter,OperationNotSupported,InvalidParameter);

/* ## Operation: get_notification_categories
*/
NotificationIRPConstDefs::NotificationCategorySet
    get_notification_categories (
        out NotificationIRPConstDefs::EventTypesSet event_type_list,
        out NotificationIRPConstDefs::ExtendedEventTypesSet
            extended_event_type_list
    )
raises (GetNotificationCategories,OperationNotSupported);
};

};

#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	003	SP-000243	3.1.0	Split of TS - Part 3: Notification Integration Reference Point (IRP): CORBA Solution Set (SS)
S_10	3.1.0	001R1	SP-000519	3.2.0	Add pragma statement to Notification IRP IDL
S_10	3.1.0	002	SP-000519	3.2.0	Correction of IDL Errors
S_10	3.1.0	003	SP-000519	3.2.0	Spelling Errors in the CORBA IDL
S_10	3.1.0	004	SP-000519	3.2.0	Ensure consistency with IDL exception

---

## History

<b>Document history</b>		
V3.1.0	July 2000	Publication
V3.2.0	December 2000	Publication