

ETSI TS 132 303 V4.0.0 (2001-06)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
Telecommunication Management;
Configuration Management;
Notification Integration Reference Point;
CORBA solution set version 1:1
(3GPP TS 32.303 version 4.0.0 Release 4)**



Reference

DTS/TSGS-0532303Uv4

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 2001.

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 3rd Generation Partnership Project (3GPP).

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

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

Contents

Foreword	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 Parameter mapping	13
6 IRP Agent's Behaviour	13
6.1 Subscription	13
6.2 IRP Agent supports multiple categories of Notifications	14
6.3 IRP Agent's integrity risk of attach_push_b Method	14
6.4 Quality of Service Parameters	14
Annex A (normative): Notification IRP CORBA IDL	16
Annex B (informative): Change history	22

Foreword

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

The present document is part the 32.300-series covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects; Telecommunication Management; Notification Management, as identified below:

- 32.301: “Notification Integration Reference Point: Requirements”;
- 32.302: “Notification Integration Reference Point: Information Service Version 2”;
- 32.303: “Notification Integration Reference Point: CORBA Solution Set Version 2:1”;**
- 32.304: “Notification Integration Reference Point: CMIP Solution Set Version 2:1”;

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 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 3GPP TS 32.101 [5] and 3GPP TS 32.102 [6].

Network Elements (NEs) under management and element managers generate notifications of events about occurrences within the network. Different kinds of events carry different kinds of information. For instance a new alarm as specified in Alarm IRP: Information Service [1], is one possible kind of event, an object creation as specified in Basic CM IRP: Information Service [8] is another possible kind of event.

Information of an event is carried in notification. An IRPAgent (typically an EM or a NE) emits notifications. IRPManager (typically a network management system) receives notifications. The purpose of Notification IRP is to define an interface through which an IRPManager can subscribe to IRPAgent for receiving notifications.

This IRP bases its design on work captured in ITU-T Recommendation X.734 [2], OMG Notification Service [4]. The central design ideas are:

- Separation of notification Consumers (IRPManagers) from Producers (IRPAgents);
- Notifications are sent to IRPManagers without the need for IRPManagers to periodically check for new notifications.

Common characteristics related to notifications in all other IRPs are gathered in one IRP.

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 [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. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

- [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 32.312: "Generic IRP Management: Information Service".
- [5] 3GPP TS 32.302: "Notification IRP: Information Service".
- [6] 3GPP TS 32.111-2: "Alarm IRP: Information Service".
- [7] 3GPP TS 32.101: "3G Telecom Management principles and high level requirements".
- [8] 3GPP TS 32.102: "3G Telecom Management architecture".
- [9] 3GPP TS 32.301: "Notification IRP: Requirements".
- [10] 3GPP TS 32.111-3: "Alarm IRP: CORBA Solution Set".
- [11] 3GPP TS 32.311: "Generic IRP Management: Information Service"

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 32.10 [7], 3GPP TS 32.102 [8] and 3GPP TS 32.301 [9].

- IRP document version number string (or "IRPVersion"). See 3GPP TS 32.311 [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
IOR	Interoperable Object Reference
NC	Notification Channel (OMG)
NE	Network Element
NV	Name and Value pair
EM	Element Manager
OMG	Object Management Group
QoS	Quality of Service
SS	Solution Set
UML	Unified Modelling Language (OMG)

4 Architectural features

The overall architectural feature of Notification IRP is specified in 3GPP TS 32.302 [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 32.302 [5]) defines semantics of operations visible across this IRP. These operations are the operations of the IOCs defined in [5].

Table 1 maps the operations defined in Notification IRP: IS (3GPP TS 32.302 [5]) to their equivalents (methods) in this Solution Set (SS). Specifically, the table 1 maps the operations of the IOCs defined in [5] to their equivalents in this SS. Since one of the IOCs, the NotificationIRP IOC, inherits from the ManagedGenericIRP IOC [4], the table 1 also maps the operations of ManagedGenericIRP IOC to their equivalents (methods) in this SS.

The table 1 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 32.302 [5]	SS Methods	Qualifier
subscribe	attach_push, attach_push_b, attach_pull	M, O, O
unsubscribe	detach	M
getIRPVersion (see note.)	get_notification_IRP_version	M
getSubscriptionStatus	get_subscription_status	O
getSubscriptionIds	get_subscription_ids	O
changeSubscriptionFilter	<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.
suspendSubscription	<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</p>	See box on the left

	shall be <code>suspend_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. If subscription is established using <code>attach_pull</code> , there is no SS equivalent.	
<code>resumeSubscription</code>	If subscription is established using <code>attach_push</code> , there is no SS equivalent. In other words, IRPManager cannot resume subscription. 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. If subscription is established using <code>attach_pull</code> , there is no SS equivalent.	See box on the left
<code>getNotificationCategories</code>	<code>get_notification_categories</code>	O
<code>getOperationProfile</code> (see note.)	<code>get_notification_IRP_operation_profile</code>	O
<code>getNotificationProfile</code> (see note.)	<code>get_notification_IRP_notification_profile</code>	O

Note: These 3 operations are operations of `ManagedGenericIRP` IOC specified in [4]. The `NotificationIRP` IOC of [5] inherits from it.

5.2 Operation parameter mapping

3GPP TS 32.302 [5] defines semantics of parameters carried in operations across the Notification IRP. Table 2 through table 14 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>	<code>string manager_reference</code> (see NOTE 1)	M
<code>timeTick</code>	<code>long time_tick</code>	O
<code>notificationCategories</code>	<code>NotificationIRPConstDefs::NotificationCategorySet notification_category_set</code>	O
<code>filter</code>	<code>string filter</code> (see NOTE 2)	O
<code>subscriptionId</code>	Return value of type <code>NotificationIRPConstDefs::SubscriptionId</code>	M
<code>status</code>	<code>Attach, ParameterNotSupported, InvalidParameter, AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported</code>	M

NOTE 1:	IRPManager creates a <code>CosNotifyComm::SequencePushConsumer</code> object and invokes <code>CORBA::ORB::object_to_string</code> to obtain the stringified IOR, say <code>s1</code> . IRPManager stores the <code>s1</code> . IRPManager sends <code>s1</code> as input parameter of <code>attach_push</code> to IRPAgent. IRPAgent receives <code>s1</code> , performs <code>CORBA::ORB::string_to_object</code> to obtain the IRPManager's IOR and uses it for its future methods. IRPAgent also stores the <code>s1</code> for future comparisons. IRPManager later calls <code>detach</code> with <code>s1</code> . IRPAgent receives the stringified IOR <code>s1</code> , compares it with those stored stringified IORs (e.g., <code>s1</code>), finds a match, and performs the detach process. IRPAgent pushes sequence of Structured Events towards IRPManager via the <code>CosNotifyComm::SequencePushConsumer</code> object <code>push_structured_events</code> method, depending on the supplied notification categories and filter.
NOTE 2:	The grammar of the filter string is <code>extended_TCL</code> defined by OMG Notification Service (OMG TC Document telecom [2]). This SS and the Alarm IRP: CORBA SS [10] shall use this grammar only..

Table 3: Mapping from IS subscribe parameters to SS attach_push_b equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>managerReference</code>	string <code>manager_reference</code> (see NOTE 1)	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 2)	O
<code>subscriptionId</code>	Return value of type <code>NotificationIRPConstDefs::SubscriptionId</code>	M
Not specified in IS	<code>CosNotifyChannelAdmin::SequenceProxyPushSupplier</code> <code>system_reference</code> (see NOTE 3)	M
<code>status</code>	<code>Attach</code> , <code>OperationNotSupported</code> , <code>ParameterNotSupported</code> , <code>InvalidParameter</code> , <code>AlreadySubscribed</code> , <code>AtLeastOneNotificationCategoryNotSupported</code>	M
NOTE 1:	IRPManager creates a <code>CosNotifyComm::SequencePushConsumer</code> object and invokes <code>CORBA::ORB::object_to_string</code> to obtain the stringified IOR, say <code>s1</code> . IRPManager stores the <code>s1</code> . IRPManager sends <code>s1</code> as input parameter of <code>attach_push_b</code> to IRPAgent. IRPAgent receives <code>s1</code> and stores the <code>s1</code> for future comparisons. IRPManager later calls <code>detach</code> with <code>s1</code> . IRPAgent receives the stringified IOR <code>s1</code> , compares it with those stored stringified IORs (e.g., <code>s1</code>), finds a match, and performs the detach process.	
NOTE 2:	The grammar of the filter string is <code>extended_TCL</code> defined by OMG Notification Service (OMG TC Document telecom [2]). This SS and the Alarm IRP: CORBA SS [10] shall use this grammar only.	
NOTE 3:	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>CosNotifyChannelAdmin::SequenceProxyPushSupplier</code> and <code>CosNotifyComm::SequencePushConsumer</code> . IRPManager is expected to invoke <code>connect_sequence_push_consumer</code> method 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
<code>managerReference</code>	string <code>manager_reference</code> (see NOTE 1)	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 2)	O
<code>subscriptionId</code>	Return value of type <code>NotificationIRPConstDefs::SubscriptionId</code>	M
Not specified in IS.	<code>CosNotifyChannelAdmin::SequenceProxyPullSupplier</code> <code>system_reference</code> (see NOTE 3)	M
<code>status</code>	<code>Attach</code> , <code>OperationNotSupported</code> , <code>ParameterNotSupported</code> , <code>InvalidParameter</code> , <code>AlreadySubscribed</code> , <code>AtLeastOneNotificationCategoryNotSupported</code>	M

NOTE 1:	IRPManager creates a <code>CosNotifyComm::SequencePullConsumer</code> object and invokes <code>CORBA::ORB::object_to_string</code> to obtain the stringified IOR, say <code>s1</code> . IRPManager stores the <code>s1</code> . IRPManager sends <code>s1</code> as input parameter of <code>attach_pull</code> to IRPAgent. IRPAgent receives <code>s1</code> and stores the <code>s1</code> for future comparisons. IRPManager later calls <code>detach</code> with <code>s1</code> . IRPAgent receives the stringified IOR <code>s1</code> , compares it with those stored stringified IORs (e.g., <code>s1</code>), finds a match, and performs the detach process.
NOTE 2:	The grammar of the filter string is <code>extended_TCL</code> defined by OMG Notification Service (OMG TC Document telecom [2]). This SS and the Alarm IRP: CORBA SS [10] shall use this grammar only.
NOTE 3:	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>CosNotifyChannelAdmin::SequenceProxyPullSupplier</code> and <code>CosNotifyComm::SequencePullConsumer</code> . IRPManager is expected to invoke <code>connect_sequence_pull_consumer</code> method of this interface to connect its own <code>CosNotifyComm::SequencePullConsumer</code> with this reference. After successful connection, IRPManager pulls sequence of Structured Events from IRPAgent.

Table 5: Mapping from IS unsubscribe parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>managerReference</code>	string <code>manager_reference</code>	M
<code>subscriptionId</code>	<code>NotificationIRPConstDefs::SubscriptionId</code> <code>subscription_id</code>	O
<code>status</code>	<code>Detach, InvalidParameter</code>	M

Table 6: Mapping from IS getIRPVersion parameters to SS equivalents

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

Table 7: Mapping from IS getSubscriptionStatus parameters to SS equivalents

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

Table 8: Mapping from IS getSubscriptionIds parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>managerReference</code>	string <code>manager_reference</code>	M
<code>subscriptionIdList</code>	Return value of type <code>NotificationIRPConstDefs::SubscriptionIdSet</code>	M
<code>status</code>	<code>GetSubscriptionIds, OperationNotSupported, InvalidParameter</code>	M

Table 9: Mapping from IS changeSubscriptionFilter parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>subscriptionId</code>	<code>NotificationIRPConstDefs::SubscriptionId</code> <code>subscription_id</code>	M
<code>filter</code>	string <code>filter</code>	M
<code>status</code>	<code>ChangeSubscriptionFilter, OperationNotSupported, InvalidParameter</code>	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
notificationCategoryList	Return value of type NotificationIRPConstDefs::NotificationCategorySet	M
eventTypeList	NotificationIRPConstDefs::EventTypesSet event_type_list	O
extendedEventTypeList	NotificationIRPConstDefs::ExtendedEventTypesSet extended_event_type_list	O
status	GetNotificationCategories, OperationNotSupported	M

Table 13: Mapping from IS `getOperationProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>irpVersion</code>	<code>ManagedGenericIRPConstDefs::VersionNumber</code> <code>notification_irp_version</code>	M
<code>operationNameProfile, operationParameterProfile</code>	Return of type <code>ManagedGenericIRPConstDefs::MethodList</code>	M
<code>status</code>	<code>GetNotificationIRPOperationsProfile, OperationNotSupported, InvalidParameter</code>	M

Table 14: Mapping from IS `getNotificationProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>irpVersion</code>	<code>ManagedGenericIRPConstDefs::VersionNumber</code> <code>notification_irp_version</code>	M
<code>notificationNameProfile, notificationParameterProfile</code>	Return value of type <code>ManagedGenericIRPConstDefs::MethodList</code>	M
<code>status</code>	<code>GetNotificationIRPNotificationProfile, OperationNotSupported, InvalidParameter</code>	M

5.3 Parameter mapping

Notification IRP: IS (3GPP TS 32.302 [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 IRPAgent's Behaviour

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

6.1 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.

6.2 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`.

6.3 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 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.

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

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

6.4 Quality of Service Parameters

The OMG Notification Service [2] supports a variety of Quality of Service (QoS) properties, such as reliability and priority, that may be expressed to indicate the delivery characteristics of notifications. The following OMG Notification Service QoS parameter settings shall be required when the IRPAgent uses the OMG Notification Service to support this SS:

1. The order policy shall be set to `FifoOrder` (First-in, First-out) [2].
2. The message priority shall be set to 0, i.e., no priority [2].

3. The Start Time Supported shall be set to false, i.e., do not use Start Time [2].
4. The Stop Time Supported shall be set to false, i.e., do not use Stop Time [2].

When the OMG Notification Service is not used, the IRP Agent shall provide First-in, First-out notification ordering, not provide message priority and not provide the support of Start Time and Stop Time.

Annex A (normative): Notification IRP CORBA IDL

```
#include "TimeBase.idl"

#ifndef ManagedGenericIRP_idl
#define ManagedGenericIRP_idl

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

/* ## Module: ManagedGenericIRPConstDefs
This module contains definitions commonly used among all IRPs such as Alarm IRP.
=====
*/
module ManagedGenericIRPConstDefs
{
    /*
    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};

    /*
    The VersionNumber is a string that identifies the IRP specification name
    and its version number. See definition "IRP document version number
    string" or "IRPVersion".

    The VersionNumberSet is a sequence of such VersionNumber. It is returned
    by get_XXX_IRP_versions(). The sequence order has no significance.
    */
    typedef string VersionNumber;
    typedef sequence <VersionNumber> VersionNumberSet;

    typedef string MethodName;
    typedef string ParameterName;
    typedef sequence <ParameterName> ParameterList;

    /*
    The Method defines the structure to be returned as part of
    get_supported_operations_profile(). The name shall be the actual method
    name (ex. "attach_push", "change_subscription_filter", etc.)
    The parameter_list contains a list of strings. Each string shall be
    the actual parameter name (ex. "manager_reference", "filter", etc.)
    */
    struct Method
    {
        MethodName name;
        ParameterList parameter_list;
    };

    /*
    List of all methods and their associated parameters.
    */

```

```
typedef sequence <Method> MethodList;
};

/* ## Module: ManagedGenericIRPSystem
This module contains definitions commonly used among all IRPs such as Alarm IRP.
=====
*/
module ManagedGenericIRPSystem
{
    /*
    Exception thrown when an unsupported optional parameter
    is passed with information.
    The parameter shall be the actual unsupported parameter name.
    */
    exception ParameterNotSupported { string parameter; };

    /*
    Exception thrown when an invalid parameter value is passed.
    The parameter shall be the actual parameter name.
    */
    exception InvalidParameter { string parameter; };

    /*
    Exception thrown when an unsupported optional method is called.
    */
    exception OperationNotSupported {};
};

#endif

#include "CosNotifyChannelAdmin.idl"
#include "generic.idl"

#ifndef NotificationIRP_idl
#define NotificationIRP_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 sub-clause
    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};
};

/* ## 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.
    */
    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 string 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 string 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 string 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 string subscription_id,
    in string filter
)
raises (ChangeSubscriptionFilter,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Check the current state of the subscription.
*/
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,
        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 string subscription_id
    )
    raises (SuspendSubscription,
           ManagedGenericIRPSystem::OperationNotSupported);

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

    /*
    Terminates the subscription with the agent.
    */
    void detach (
        in string manager_reference,
        in string subscription_id
    )
    raises (DetachException);
};

#endif
```

Annex B (informative): Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
Jun 2001	S_12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	2.0.0	4.0.0

History

Document history		
V4.0.0	July 2001	Publication