

Source: SA5 (Telecom Management)
Title: 32.106 CR, "Split of TS - Part 3: Notification Integration Reference Point (IRP): CORBA Solution Set (SS)" (S5-000325)
Document for: Approval
Agenda Item: 6.5.3

SA5 has split TS 32.106 Configuration Management (CM) into a multi-part TS as identified below:

Part 1: "3G Configuration Management";
Part 2: "Notification IRP Information Service";

Part 3: "Notification IRP CORBA Solution Set";

Part 4: "Notification IRP CMIP Solution Set";
Part 5: "Basic Configuration Management IRP Information Model (including NRM)";
Part 6: "Basic Configuration Management IRP CORBA Solution Set";
Part 7: "Basic Configuration Management IRP CMIP Solution Set";
Part 8: "Name Convention for Managed Objects"

Five (5) CRs are submitted to SA#8 for approval; the present one is highlighted in **yellow**:

Spec	CR	Phase	Subject	Cat	Version - Current	Version -New	Doc-2nd-Level
32.106	001	R99	Split of TS - Part 1: Main part of spec - Concept and Requirements	F	3.0.1	3.1.0	S5-000323
32.106	002	R99	Split of TS - Part 2: Notification IRP Information Service (IS)	F	3.0.1	3.1.0	S5-000324
32.106	003	R99	Split of TS - Part 3: Notification IRP CORBA SS	F	3.0.1	3.1.0	S5-000325
32.106	004	R99	Split of TS - Part 4: Notification IRP CMIP SS	F	3.0.1	3.1.0	S5-000326
32.106	005	R99	Split of TS - Part 8: Name Convention for Managed Objects	F	3.0.1	3.1.0	S5-000327

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

32.106 CR 003

Current Version: **3.0.1**

GSM (AA.BB) or 3G (AA.BBB) specification number ↑

↑ CR number as allocated by MCC support team

For submission to: **SA#8**
list expected approval meeting # here ↑

for approval
for information

strategic (for SMG use only)
non-strategic

Form: CR cover sheet, version 2 for 3GPP and SMG The latest version of this form is available from: <ftp://ftp.3gpp.org/Information/CR-Form-v2.doc>

Proposed change affects: (U)SIM ME UTRAN / Radio Core Network
(at least one should be marked with an X)

Source: SA5#12 **Date:** 20 June 2000

Subject: Split of TS - Part 3: Notification Integration Reference Point (IRP): CORBA Solution Set (SS)

Work item: 32.106 Configuration Management

Category: F Correction **Release:** Phase 2
A Corresponds to a correction in an earlier release Release 96
(only one category shall be marked with an X) B Addition of feature Release 97
C Functional modification of feature Release 98
D Editorial modification Release 99
Release 00

Reason for change: This essential part of 32.106, which was missing in version 3.0.1, has now been completed and agreed by SA5 for release 99.

Clauses affected: All.

Other specs affected: Other 3G core specifications → List of CRs:
Other GSM core specifications → List of CRs:
MS test specifications → List of CRs:
BSS test specifications → List of CRs:
O&M specifications → List of CRs:

Other comments:

3G TS 32.106-3 V3.0.1da (2000-0654)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Part 3: Notification IRP: CORBA Solution Set Version 1:1 (Release 1999)



The present document has been developed within the 3rd Generation Partnership Project (3GPP™) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPP Organisational Partners and shall not be implemented. This Specification is provided for future development work within 3GPP only. The Organisational Partners accept no liability for any use of this Specification. Specifications and reports for implementation of the 3GPP™ system should be obtained via the 3GPP Organisational Partners' Publications Offices.

Keywords

Configuration Management

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

<http://www.3gpp.org>

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.

© 2000, 3GPP Organizational Partners (ARIB, CWTS, ETSI, T1, TTA, TTC).
All rights reserved.

Contents

Foreword.....	4
Introduction.....	4
1 Scope.....	5
2 References	5
3 Definitions and abbreviations.....	5
3.1 Definitions	5
3.2 Abbreviations.....	7
4 Architectural Features	8
4.1 Notification Services.....	8
4.1.1 Support of Push and Pull Interface.....	8
4.1.2 Support of multiple notifications in one push operation	8
5 Mapping.....	9
5.1 Operation mapping	9
5.2 Operation parameter mapping.....	10
5.3 Notification parameter mapping	14
5.4 Attribute mapping	14
6 Use of OMG Notification StructuredEvent	14
7 IRPAgent's Behaviour.....	17
7.1 Subscription.....	17
7.2 IRPAgent Supports Multiple Categories of Notifications.....	17
7.3 IRPAgent's Integrity Risk of attach_push_b Method.....	17
8 Example.....	19
Appendix A: Notification IRP CORBA IDL.....	20
Annex A (informative): Change history	30
Foreword.....	4
Introduction.....	4
1 Scope.....	5
2 References	5
3 Definitions and abbreviations.....	5
3.1 Definitions	5
3.2 Abbreviations.....	7
4 Notification IRP: CORBA Solution Set.....	7
Annex A (informative): Change history.....	8

Foreword

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

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 NEs and NRs, and they may be initiated by the operator or functions in the 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. The CM actions are initiated either as a single action on a network element of the 3G network or as part of a complex procedure involving actions on many network elements.

~~In this document, Clauses 4 through 6 are here provided to give an introduction and description of the main concepts of configuration management, which is not mandatory for the compliance to this specification in this release. Clause 7 contains the specific definitions for the standardised N interface, which are necessary to follow for compliance.~~

~~Clause 4 provides a brief background of CM while Clause 5 explains CM services available to the operator. Clause 6 breaks these services down into individual CM functions, which support the defined services. Clause 7 defines the N-INTERFACE interface to be used for 3G CM~~

The Itf-N interface for Configuration Management 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 [1] and [2]. For CM, a number of IRPs (and the Name Convention) are defined herein, used by this as well as other technical specifications for telecom management produced by 3GPP. All these documents are included in Parts 2-N the 3G TS 32.106.

This document constitutes 32.106 Part 3 (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 [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 describes the Configuration Management (CM) aspects of managing a 3G network. This is described from the management perspective outlined in the two 3GPP specifications 32.101 [1] and 32.102 [2].

The present document defines a set of controls to be employed to effect set-up and changes to a 3G network in such a way that operational capability and quality of service, network integrity and system inter-working are ensured. In this way, the present document describes the interface definition and behaviour for the management of relevant 3G network NEs in the context of the described management environment. The context is described for both the management systems (OS) and NE functionality.

Clause 7 contains the specific definitions for the standardised N interface, which are necessary to follow for compliance to this specification.

This document specifies the CORBA Solution Set (SS) for the IRP whose semantics is specified in Notification IRP: Information Service (IS) [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 Notification Service OMG TC Document telecom/98-11-01

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

[4] 3G TS 32.106-8: "Name Convention for Managed Objects"

[5] 3G TS 32.106-2: "Notification IRP: Information Service"

[6] 3G TS 32.111-2: "Alarm IRP: Information Service"

[7] 3G TS 32.111-3: "Alarm IRP: CORBA solution set, version 1:1"

[8] ITU-T Recommendation X.733: Alarm Reporting function

[9] 3G TSPP 32.101: "3G Telecom Management principles and high level requirements".

[10] 3G TSPP 32.102: "3G Telecom Management architecture".

[11] 3G TS 32.106-1: "3G Configuration Management".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply: Please refer to [9],[10] and [11].

Data: is any information or set of information required to give software or equipment or combinations thereof a specific state of functionality

Element Manager (EM): provides a package of end-user functions for management of a set of closely related types of network elements. These functions can be divided into two main categories:

- *Element Management Functions* for management of network elements on an individual basis. These are basically the same functions as supported by the corresponding local terminals.
- *Sub Network Management Functions* that are related to a network model for a set of network elements constituting a clearly defined sub network, which may include relations between the network elements. This model enables additional functions on the sub network level (typically in the areas of network topology presentation, alarm correlation, service impact analysis and circuit provisioning).

Equipment: is one or more hardware items which correspond to a manageable or supervisable unit or is described in an equipment model

Firmware: is a term used in contrast to software to identify the hard coded program, which is not downloadable on the system

Hardware: is each and every tangible item

IRP Information Model: See [1].

IRP Information Service: See [1].

IRP Solution Set: See [1].

Managed Object (MO): an abstract entity which may be accessed through an open interface between two or more systems, and representing a Network Resource for the purpose of management. The MO is an instance of a Managed Object Class (MOC) as defined in a Management Information Model (MIM). The MIM does not define how the MO or NR is implemented; only what can be seen in the interface

Managed Object Class (MOC): a description of all the common characteristics for a number of MOs, such as their attributes, operations, notifications and behaviour

Managed Object Instance (MOI): an instance of a MOC, which is the same as a MO as described above

Management Information Base (MIB): the set of existing managed objects in a management domain, together with their attributes, constitutes that management domain's MIB. The MIB may be distributed over several OS/Nes

Management Information Model (MIM): also referred to as NRM—see the definition below. There is a slight difference between the meaning of MIM and NRM—the term MIM is generic and can be used to denote any type of management model, while NRM denotes the model of the actual managed telecommunications network resources

Network Element: is a discrete telecommunications entity, which can be, managed over a specific interface e.g. the RNC

Network Manager (NM): provides a package of end-user functions with the responsibility for the management of a network, mainly as supported by the EM(s) but it may also involve direct access to the network elements. All communication with the network is based on open and well standardized interfaces supporting management of multi-vendor and multi technology network elements

Network Resource: is a component of a Network Element which can be identified as a discrete separate entity and is in an object oriented environment for the purpose of management represented by an abstract entity called Managed Object

Network Resource Model (NRM): a model representing the actual managed telecommunications network resources that a System is providing through the subject IRP. An NRM describes managed object classes, their associations, attributes and operations. The NRM is also referred to as "MIM" (see above) which originates from the ITU T TMN

Object Management Group (OMG): see <http://www.omg.org>

Operations System (OS): indicates a generic management system, independent of its location level within the management hierarchy

Operator: is either

- a human being controlling and managing the network; or
- a company running a network (the 3G network operator)

Optimisation: of the network is each up-date or modification to improve the network handling and/or to enhance subscriber satisfaction. The aim is to maximise the performance of the system

Re-configuration: is the re-arrangement of the parts, hardware and/or software that make up the 3G network. A re-configuration can be of the parts of a single NE or can be the re-arrangement of the NEs themselves, as the parts of the 3G network. A re-configuration may be triggered by a human operator or by the system itself

Reversion: is a procedure by which a configuration, which existed before changes were made, is restored

Software: is a term used in contrast to firmware to refer to all programs which can be loaded to and used in a particular system

Up-Dates: generally consist of software, firmware, equipment and hardware, designed only to consolidate one or more modifications to counter-act errors. As such, they do not offer new facilities or features and only apply to existing Nes

Up-Grades: can be of the following types:

- enhancement—the addition of new features or facilities to the 3G network;
- extension—the addition of replicas of existing entities.

3.2 Abbreviations

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

<u>CM</u>	<u>Configuration Management</u>
<u>CORBA</u>	<u>Common Object Request Broker Architecture (OMG)</u>
<u>EC</u>	<u>Event channel (OMG)</u>
<u>IDL</u>	<u>Interface Definition Language (OMG)</u>
<u>IS</u>	<u>Information Service</u>
<u>NC</u>	<u>Notification Channel (OMG)</u>
<u>NE</u>	<u>Network Element</u>
<u>EM</u>	<u>Element Manager</u>
<u>OMG</u>	<u>Object Management Group</u>
<u>SS</u>	<u>Solution Set</u>
<u>UML</u>	<u>Unified Modelling Language (OMG)</u>
CM	Configuration Management
CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture
EM	Element Manager
FM	Fault Management
FW	Firmware
HW	Hardware
MIB	Management Information Base
MIM	Management Information Model
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NM	Network Manager
NR	Network Resource
NRM	Network Resource Model
OMG	Object Management Group

OS	Operations System
OSF	Operations System Function
SW	Software
TRX	Transceiver
TS	Technical Specification
UML	Unified Modelling Language (OMG)

~~4 Notification IRP: CORBA Solution Set~~

4 Architectural Features

The overall architectural feature of Notification IRP is specified in Reference [5]. This clause specifies features that are specific to the CORBA solution set.

4.1 Notification Services

In the CORBA solution set, notifications are emitted by IRPAgent using CORBA Notification service [2].

CORBA Event service [3] provides event routing and distribution capabilities. CORBA Notification service provides, in addition to Event service, event filtering and support for quality of service 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 [2] defined methods, is identified in this document.

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 [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 [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 [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 [5] defines semantics of operations visible across this IRP.

The table below maps the operations defined in Notification IRP: IS [5] to their equivalents (methods) in this SS. It also qualifies if a method is mandatory (M) or optional (O)

Table 14: Mapping from IS Operation to SS Equivalents

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

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

5.2 Operation parameter mapping

Reference [5] defines semantics of parameters carried in operations across the Notification IRP. The tables below indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

Table 22: Mapping from IS subscribe parameters to SS attach_push equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>managerReference</u>	Object <u>manager_reference</u>	<u>M</u>
<u>timeTick</u>	long <u>time_tick</u>	<u>O</u>
<u>notification Categories</u>	NotificationCategorySet string <u>notification_category_set</u>	<u>O</u>
<u>filter</u>	string <u>filter</u> (See note below table.)	<u>O</u>
<u>subscriptionId</u>	Return value of type SubscriptionId string <u>subscription_id</u>	<u>M</u>
<u>status</u>	CommonIRPConstDefs::Signal AttachException, ParameterNotSupportedException, InvalidParameter, AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported Exception	<u>M</u>

Note: The grammar of the filter string is extended TCL defined by OMG Notification Service [2]. This grammar shall be the only one used for Alarm IRP: CORBA SS.

Table 33: Mapping from IS subscribe parameters to SS attach_push_b equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>managerReference</u>	Object <u>manager_reference</u>	<u>M</u>
<u>timeTick</u>	long <u>time_tick</u>	<u>O</u>
<u>notification Categories</u>	NotificationCategorySet string <u>notification_category_set</u>	<u>O</u>
<u>filter</u>	string <u>filter</u>	<u>O</u>
<u>subscriptionId</u>	Return value of type SubscriptionId string <u>subscription_id</u>	<u>M</u>

<u>Not specified in IS</u>	<u>CosNotifyChannelAdmin::SequenceProxyPushSupplier</u> <u>system_reference (See note below table.)</u>	<u>M</u>
<u>status</u>	<u>CommonIRPConstDefs::Signal</u> <u>AttachException, OperationNotSupportedException,</u> <u>ParameterNotSupportedException, InvalidParameter,</u> <u>AlreadySubscribed,</u> <u>AtLeastOneNotificationCategoryNotSupported</u> <u>Exception</u>	<u>M</u>

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 SequencePushSupplier:proxySupplier, CosNotifyComm::SequencePushSupplier{}. IRPManager is expected to invoke connect_sequence_push_consumer() of this interface to connect its own cosNotifyComm::sequencePushConsumer with this reference. After successful connection, IRPAgent pushes sequence of Structured Events towards IRPManager.

Table 44: Mapping from IS subscribe parameters to SS attach_pull equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>managerReference</u>	<u>Object manager_reference</u>	<u>M</u>
<u>timeTick</u>	<u>long time_tick</u>	<u>O</u>
<u>notification</u> <u>Categories</u>	<u>NotificationCategorySet string</u> <u>notification_category_sety</u>	<u>O</u>
<u>filter</u>	<u>string filter</u>	<u>O</u>
<u>subscriptionId</u>	<u>Return value of type SubscriptionIdstring</u> <u>subscription_id</u>	<u>M</u>
<u>Not specified in IS.</u>	<u>CosNotifyChannelAdmin::SequenceProxyPullSupplier</u> <u>system_reference</u>	<u>M</u>
<u>status</u>	<u>CommonIRPConstDefs::Signal</u> <u>AttachException, OperationNotSupportedException,</u> <u>ParameterNotSupportedException,</u> <u>InvalidParameter,Exception AlreadySubscribed,</u> <u>AtLeastOneNotificationCategoryNotSupported</u>	<u>M</u>

Table 55: Mapping from IS unsubscribe parameters to SS equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>managerReference</u>	<u>Object manager_reference</u>	<u>M</u>
<u>subscriptionId</u>	<u>string subscription_id</u>	<u>O</u>
<u>status</u>	<u>CommonIRPConstDefs::Signal</u> <u>DetachException, InvalidParameterException</u>	<u>M</u>

Table 66: Mapping from IS getNotificationIRPVersion parameters to SS equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>versionNumber</u> <u>List</u>	<u>Return value of type CommonIRPConstDefs::VersionNumberSet</u> <u>CommonIRPConstDefs::VersionNumberSeq</u> <u>version_number_list</u>	<u>M</u>
<u>status</u>	<u>CommonIRPConstDefs::Signal</u> <u>GetNotificationIRPVersionException, InvalidParameter</u> <u>Exception</u>	<u>M</u>

Table 77: Mapping from IS `getSubscriptionStatus` parameters to SS equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>subscriptionId</u>	string <u>subscription_id</u>	<u>M</u>
<u>notificationCategoryList</u>	<u>Return value of type</u> NotificationIRPConstDefs::NotificationCategorySet NotificationIRPConstDefs::NotificationCategorySeq <u>notification_category_list</u>	<u>M</u>
<u>filterInEffect</u>	string <u>filter_in_effect</u>	<u>O</u>
<u>subscriptionState</u>	NotificationIRPConstDef::SubscriptionState <u>subscription_state</u>	<u>O</u>
<u>timeTick</u>	long <u>time_tick</u>	<u>O</u>
<u>status</u>	CommonIRPConstDefs::Signal GetSubscriptionStatusException, OperationNotSupported Exception, InvalidParameterException	<u>M</u>

Table 88: Mapping from IS `getSubscriptionIds` parameters to SS equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>managerReference</u>	Object <u>manager_reference</u>	<u>M</u>
<u>subscriptionIdList</u>	<u>Return value of type</u> NotificationIRPConstDefs::SubscriptionIdSet NotificationIRPConstDefs::SubscriptionIdSeq <u>subscription_id_list</u>	<u>M</u>
<u>status</u>	CommonIRPConstDefs::Signal GetSubscriptionIdsException, OperationNotSupported Exception, InvalidParameterException	<u>M</u>

Table 99: Mapping from IS `changeSubscriptionFilter` parameters to SS equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>subscriptionId</u>	string <u>subscription_id</u>	<u>M</u>
<u>filter</u>	string <u>filter</u>	<u>M</u>
<u>status</u>	CommonIRPConstDefs::Signal ChangeSubscriptionFilterException, OperationNotSupported Exception, InvalidParameterException	<u>M</u>

Table 1040: Mapping from IS `suspendSubscription` parameters to SS equivalents

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

Table 1144: Mapping from IS `resumeSubscription` parameters to SS equivalents

<u>IS Operation parameter</u>	<u>SS Method parameter</u>	<u>Qualifier</u>
<u>subscriptionId</u>	<p>If subscription is established using <code>attach_push</code>, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.</p> <p>If subscription is established using <code>attach_push_b</code>, the SS equivalent method is <code>resume_connection</code>. This method is defined by <u>OMG Notification Service [2]</u> and requires no parameter. Therefore, there is no SS equivalent for this IS parameter.</p> <p>If subscription is established using <code>attach_pull</code>, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.</p>	<u>M</u>
<u>status</u>	<p>If subscription is established using <code>attach_push</code>, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.</p> <p>If subscription is established using <code>attach_push_b</code>, the SS equivalent method is <code>resume_connection</code>. This method is defined by <u>OMG Notification Service [2]</u> and returns a void. Therefore, there is no SS equivalent for this IS parameter. This <code>resume_connection</code> method can raise throw <u>OMG Notification Service [2]</u> defined exception called <code>ConnectionAlreadyActive</code>.</p> <p>If subscription is established using <code>attach_pull</code>, there is no SS equivalent method. Therefore, there is no SS equivalent for this IS parameter.</p>	<u>M</u>

Table 1242: Mapping from IS `getNotificationIRPCategories` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<code>notificationCategoryList</code>	Return value of type <code>NotificationIRPCConstDefs::NotificationCategorySet</code> <code>NotificationIRPCConstDefs::NotificationCategorySeq</code> <code>notification_category_list</code>	M
<code>eventTypeList</code>	<code>NotificationIRPCConstDefs::EventTypesSet</code> <code>event_type_list</code>	O
<code>extendedEventTypeList</code>	<code>NotificationIRPCConstDefs::ExtendedEventTypesSet</code> <code>extended_event_type_list</code>	O
<code>status</code>	<code>CommonIRPCConstDefs::Signal</code> <code>GetNotificationIRPCategoriesException</code> , <code>OperationNotSupportedException</code>	M

5.3 Notification parameter mapping

Notification IRP: IS [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 such as Alarm IRP: IS [6] extends the generic `notify` for its specific use. Their corresponding SS documents shall define the mapping from their specific notification parameters (defined in their IS document) to their SS equivalents. The SS documents shall qualify their SS equivalents as well.

5.4 Attribute mapping

Notification IRP: IS [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 [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 [5] defines attributes that are commonly present in notifications of all notification categories such as notifications emitted from Alarm IRP IRP Agent.

In CORBA SS, OMG defined StructuredEvent [2] is used to carry notification. This clause identifies the OMG defined StructuredEvent attributes that carry the common attributes defined in [5].

The composition of OMG StructuredEvent is:

```

Header
  Fixed Header
    Domain_name
    Type_name
    Event_name
  Variable Header
Body
  Filterable_body_fields
  Remaining_body

```

Following table shows the OMG Structured Event attributes (middle column) that are used to carry the common notification attributes defined in Notification IRP: IS [5].

Table 1313: Attributes of StructuredEvent

Common attributes defined in Notification IRP: IS [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 interface <u>IRPNotificationCategoryValue</u>.module.</p> <p>For Alarm IRP version 1:1, the value is <u>ALARM_IRP_VERSION_1_1</u>. For Alarm IRP version 1:1, the value is IRPNotificationCategoryValue.alarmIRPVersion_1_1,</p>
eventType	type_name	<p>It indicates event types as defined in ITU-T TMN Recommendations. For Alarm IRP version 1:1, i.e., the value of domain_type is <u>IRPNotificationCategoryValue.alarmIRPVersion_1_1</u>. The legal values of this are: EVENT_COMMUNICATIONS_ALARM (section 8.1.1 of [8]); EVENT_QUALITY_OF_SERVICE_ALARM (section 8.1.1 of [8]); EVENT_PROCESSING_ERROR_ALARM (section 8.1.1 of [8]); EVENT_EQUIPMENT_ALARM (section 8.1.1 of [8]); EVENT_ENVIRONMENTAL_ALARM (section 8.1.1 of [8]); EVENT_PHYSICAL_VIOLATION [1]; EVENT_INTEGRITY_VIOLATION [1]; EVENT_SECURITY_VIOLATION [1]; EVENT_TIME_DOMAIN_VIOLATION [1]; EVENT_OPERATIONAL_VIOLATION [1]; 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. That 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><u>COMMUNICATIONS_ALARM (section 8.1.1 of [8]),</u> <u>QUALITY_OF_SERVICE_ALARM (section 8.1.1 of [8]),</u> <u>PROCESSING_ERROR_ALARM (section 8.1.1 of [8]),</u> <u>EQUIPMENT_ALARM (section 8.1.1 of [8]),</u> <u>ENVIRONMENTAL_ALARM (section 8.1.1 of [8]),</u> <u>PHYSICAL_VIOLATION [1], INTEGRITY_VIOLATION [1],</u> <u>SECURITY_VIOLATION [1], TIME_DOMAIN_VIOLATION [1],</u> <u>OPERATIONAL_VIOLATION [1].</u></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 individual IRP SS module 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 <u>domain_name</u> of the same notification.</p> <p>It is a string. See interface <u>EventTypeValue</u> of module</p>

		<u>CommonIRPConstDefs of this IRP IDL for all possible legal values.</u>
<u>extended EventType</u>	<u>event_name</u>	<p>The legal values carried in this attribute are specified by the IRP using the notification. For example, Alarm IRP: CORBA SS [7] defines and uses the following values:</p> <p><u>NOTIFY_FM_NEW_ALARM, NOTIFY_FM_CHANGED_ALARM, NOTIFY_FM_ACK_STATE_CHANGED, NOTIFY_FM_CLEARED_ALARM and NOTIFY_FM_ALARM_LIST_REBUILT.</u></p> <p>It is a string. See individual IRP SS module for legal values used by that IRP version, the constant string definition of the IRP using the notification. For example, for Alarm IRP: CORBA SS [7], see module <u>AlarmIRPConstDefs of the IDL file.</u></p> <p>Since each IRP except Notification IRP specifies its own set of <u>extended event type extendedEventType</u>, the values specified by each IRP are only unique within one IRP. For uniqueness among all IRPs' specification, the values of extended event type <u>extendedEventType</u> shall be coupled with the notification category, the value carried in <u>domain_name</u> of the same notification.</p>
<u>There is no corresponding SS attribute.</u>	<u>variable Header</u>	
<u>managed Object Class, managed Object Instance</u>	<u>One NV (name- value) pair of filterable_ body_fields</u>	<p>Name of NV pair is a string, <u>NV_MANAGED_OBJECT_INSTANCEAttributeNameValue.managedObjectInstance.</u></p> <p>Value of NV pair is a string. Syntax and semantics of this string conform to the Managed Object string representation specified in [4]. Note that two SS attributes are carried in this one NV pair since the string representation specified in [4] can convey the semantics of <u>managedObjectClass</u> and <u>managedObjectInstance</u> in one string.</p>
<u>notificatio nId</u>	<u>One NV pair of filterable_ body_fields</u>	<p>Name of NV pair is a string, <u>NV_NOTIFICATION_IDAttributeNameValue.notificationId.</u></p> <p>Value of NV pair is an unsigned long.</p>
<u>eventTime</u>	<u>One NV pair of filterable_ body_fields</u>	<p>Name of NV pair is a string, <u>AttributeNameValue.NV_EVENT_TIMEeventTime.</u></p> <p>Value of NV pair is an IRPTime.</p>
<u>systemDN</u>	<u>One NV pair of filterable_ body_fields</u>	<p>Name of NV pair is a string, <u>NV_SYSTEM_DNAttributeNameValue.systemDN.</u></p> <p>Value of NV pair is a string. Syntax and semantics of this string conforms to the Managed Object string representation specified in [4].</p>
<u>There is no corresponding SS attribute.</u>	<u>remaining_ Body</u>	

7 IRPAgent's Behaviour

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

7.1 Subscription

IRPManager can invoke multiple `attach_push`, `attach_push_b` or `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 ~~throw~~ `AlreadySubscribedException` 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 ~~different~~ `notification_category`s. In this case, if IRPAgent supports all the ~~the~~ `notification_category`s requested, IRPAgent shall accept the invocation; otherwise, it raises ~~throws~~ `AtLeastOneNotificationCategoryNotSupportedUnsupportedCategoryException`. 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_category`s 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. ~~A notification category is defined as the different kinds of notifications specified by one IRP such as Alarm IRP [7].~~ 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_IRP_categories`.

IRPManager uses a parameter, `notification_category`s, in `attach_push`, `attach_push_b` and `attach_pull` to specify one or more `category`s of notifications wanted. ~~IRPManager, by invoking multiple `attach_push`, `attach_push_b`, or `attach_pull` methods specifying different categories in each invocation can receive multiple categories of notification from IRPAgent.~~

IRPManager uses a zero-length `NULL` sequence string in `notification_category`s 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 `NULL` sequence string in `notification_category`s and if the operation is successful, IRPAgent shall reject subsequent `attach_push` operation, regardless if the `notification_category`s contains a zero-length sequence `NULL` string or one or more a-specific notification `category`s. 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 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.

Note that attach_push is mandatory. IRPAgent compliant to this IRP shall implement it. Note also that attach_push_b is optional. It is recommended that IRPAgent concerned with integrity risk should not implement the attach_push_b option.

8 Example

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, "...",
  ...
}
```

Appendix A: 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>

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

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> NotifIDSet;

/*
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.
    NotifIDSet 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_DEFINITIONS = "t";
const string NV_MONITORED_ATTRIBUTES = "u";
const string NV_PROPOSED_REPAIRED_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"

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 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 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 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 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 ()
    ;

    /* ## 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);
}i

}i

#endif


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

#ifndef CommonIRPConstDefs_idl
#define CommonIRPConstDefs_idl
#include <CosTime.idl>

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, PARTIAL_FAILURE};

    typedef sequence <string> VersionNumberSeq;

    /*
    This interface encapsulates all TMN ITU T defined event types.
    */
    interface EventTypeValue {
        const string OBJECT_CREATION = "x0";
        const string OBJECT_DELETION = "x1";
        const string ATTRIBUTE_VALUE_CHANGE = "x2";
        const string STATE_CHANGE = "x3";
        const string RELATIONSHIP_CHANGE = "x4";
        const string COMMUNICATIONS_ALARM = "x5";
        const string PROCESSING_ERROR_ALARM = "x6";
        const string ENVIRONMENTAL_ALARM = "x7";
        const string QUALITY_OF_SERVICE_ALARM = "x8";
    }
}


```

```

----- const string EQUIPMENT_ALARM = "x9";
----- const string INTEGRITY_VIOLATION = "x10";
----- const string SECURITY_VIOLATION = "x11";
----- const string TIME_DOMAIN_VIOLATION = "x12";
----- const string OPERATIONAL_VIOLATION = "x13";
----- const string PHYSICAL_VIOLATION = "x14";
----- };

};

#endif

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

#ifndef NotificationIRPConstDefs_idl
#define NotificationIRPConstDefs_idl

module NotificationIRPConstDefs {

/*
This is a sequence of string identifying notification category.
The string must be one of that in IRPNotificationCategoryValue{}.
*/
typedef sequence <string> NotificationCategorySeq;

/*
This is a sequence of strings identifying event types of a particular
notification category.
The string must be one of that in EventTypeValue{}.
*/
typedef sequence <string> EventTypePerNotificationCategorySeq;

typedef sequence <EventTypePerNotificationCategorySeq> EventTypesSeq;

/*
This is a sequence of strings identifying extended event types of
a particular notification category.
The string must be one of that in ExtendedEventTypeValue{}.
*/
typedef sequence <string> ExtendedEventTypePerNotificationCategorySeq;

typedef sequence <ExtendedEventTypePerNotificationCategorySeq>
ExtendedEventTypesSeq;

/*
This information object associates the notification ID with the DN of
the managed object that emits that notification.
*/
struct CorrelatedNotificationType {
string source;
/*
If notifID scope of uniqueness is across IRPAgent, this source
may contain a NULL string. If the scope of uniqueness is per
managed object instance, this source shall contain a non-NULL string.
If this source contains a non-NULL string, it shall contain the
string representation of DN of the managed object instance.
See Name Convention for Managed Object for the specification of
the string representation of DN.
*/
};

```

```


unsigned long notifID;
};

/*
This is a sequence of Correlated Notification.
*/
typedef sequence <CorrelatedNotificationType> CorrelatedNotificationSetType;

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

/*
This interface encapsulates valid strings carried in domain_name of
structured event header. It carries the name of IRP and its
corresponding CORBA SS version number.
*/
interface IRPNotificationCategoryValue {
const string alarmIRPVersion_1_1 = "1f1"; //alarm IRP 1:1
const string configurationIRPVersion_1_1 = "1e1"; //CM IRP 1:1
};

/*
This interface encapsulates string used in the name of the Name Value
pair of the structured event.
*/
interface AttributeNameValue {
const string notificationId = "a";
const string correlatedNotifications = "b";
const string eventTime = "c";
const string systemDN = "d";
const string managedObjectClass = "e";
const string managedObjectInstance = "f";
const string probableCause = "g";
const string perceivedSeverity = "h";
const string specificProblem = "i";
const string additionalText = "j";
const string alarmId = "k";
const string ackUserId = "l";
const string ackTime = "m";
const string ackSystemId = "n";
const string ackState = "o";
const string backedUpStatus = "p";
const string backUpObject = "q";
const string thresholdInfo = "r";
const string trendIndication = "s";
const string stateChangeDefinitions = "t";
const string monitoredAttributes = "u";
const string proposedRepairedActions = "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


```



```

_____ in long time_tick,
_____ in string notification_category,
_____ in string filter,
_____ out string subscription_id,
_____ out CosNotifyChannelAdmin::SequenceProxyPullSupplier system_reference
_____ )
_____ raises
(AttachException,OperationNotSupportedException,ParameterNotSupportedException,I
nvalidParameterException);

_____ /* ## Operation: detach
_____ */
_____ CommonIRPConstDefs::Signal detach (
_____ in Object manager_reference,
_____ in string subscription_id
_____ )
_____ raises (DetachException,InvalidParameterException);

_____ /* ## Operation: get_notification_IRP_version
_____ */
_____ CommonIRPConstDefs::Signal get_notification_IRP_version (
_____ out CommonIRPConstDefs::VersionNumberSeq version_number_list
_____ )
_____ raises (GetNotificationIRPVersionException,InvalidParameterException);

_____ /* ## Operation: get_subscription_status
_____ */
_____ CommonIRPConstDefs::Signal get_subscription_status (
_____ in string sub,
_____ out NotificationIRPConstDefs::NotificationCategorySeq
_____ notification_category_list,
_____ out string filter_in_effect,
_____ out NotificationIRPConstDefs::SubscriptionState subscription_state,
_____ out long time_tick
_____ )
_____ raises
(GetSubscriptionStatusException,OperationNotSupportedException,InvalidParameterE
xception);

_____ /* ## Operation: get_subscription_ids
_____ */
_____ CommonIRPConstDefs::Signal get_subscription_ids (
_____ in Object manager_reference,
_____ out NotificationIRPConstDefs::SubscriptionIdSeq subscription_id_list
_____ )
_____ raises
(GetSubscriptionIdsException,OperationNotSupportedException,InvalidParameterExce
ption);

_____ /* ## Operation: change_subscription_filter
_____ */
_____ CommonIRPConstDefs::Signal change_subscription_filter (
_____ in string subscription_id,
_____ in string filter
_____ )
_____ raises
(ChangeSubscriptionFilterException,OperationNotSupportedException,InvalidParamet
erException);

_____ /* ## Operation: get_notification_IRP_categories
_____ */
_____ CommonIRPConstDefs::Signal get_notification_IRP_categories (
_____ out NotificationIRPConstDefs::NotificationCategorySeq
_____ notification_category_list,
_____ out NotificationIRPConstDefs::EventTypesSeq event_type_list,
_____ out NotificationIRPConstDefs::ExtendedEventTypes

```

```
        Seqextended_event_type_list  
        ====>  
        ==== raises  
(GetNotificationIRPCategoriesException, OperationNotSupportedException);  
        ====};  
  
        };  
  
#endif
```

Annex A (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
Post S5#10S_04	3.0.03-0-0	-004	- S5-000227SP-99308	3.0.13-1-0	Updated by MCC staff with editorial changes according to documentation rules. Mechanism for data integrity of signalling messages
S S5#11 S_04	3.0.13-0-0	-002	?SP-99308	3.0.1a3-1-0	Updated according to S5#10bis (S5-000192) and S5#11 (decision to create separate parts for main body and earlier annexes). To be agreed at S5#11bis and approved at S5 #12, together with possible new updates according to S5#11bis. Description of layer on which ciphering takes place

SA5 internal Change history				
<u>SA/SA5 meeting</u>	<u>Version</u>	<u>Tdoc SA/SA5</u>	<u>New version</u>	<u>Subject/comment</u>
Post S5#11bis	3.0.1a	S5C000047	3.0.1b	Updated according to agreements at meeting #11bis (including 32.106 split into 8 parts).
Post S5#12	3.0.1b	S5C000064	3.0.1c	Updated according to agreements at meeting #12.
Post S5#12	3.0.1c	S5C000067	3.0.1d	Updated according to e-mail comments after meeting #12.