3GPP TS 32.303 V9.0.0 (2009-12)

Technical Specification

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS) (Release 9)

Keywords
UMTS, 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

Contents

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

The present document is part of a TS-family covering the 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; as identified below:

32.301: Configuration Management (CM); Notification Integration Reference Point (IRP): Requirements

32.302: Configuration Management (CM); Notification Integration Reference Point (IRP): Information Service (IS)

32.303: Configuration Management (CM); Notification Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS)

32.305: Configuration Management (CM); Notification Integration Reference Point (IRP): eXtensible Markup Language (XML) definition

32.307: Configuration Management (CM); Notification Integration Reference Point (IRP): Simple Object Access Protocol (SOAP) Solution Set (SS)

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 (3GPP TS 32.302 [5]).

This Solution Set specification is related to 3GPP TS 32.302.

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: "Information technology - Open Systems Interconnection - Systems Management: Security alarm reporting function".

[2] OMG TC Document telecom (98-11-01): "Summary of responses to real time survey".

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

[4] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[5] 3GPP TS 32.302: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Information Service (IS)".

[6] 3GPP TS 32.111-2: "Telecommunication management; Fault Management; Part 2: Alarm Integration Reference Point (IRP): Information Service (IS)".

[7] 3GPP TS 32.101: "Telecommunication management; Principles and high level requirements".

[8] 3GPP TS 32.102: "Telecommunication management; Architecture".

[9] 3GPP TS 32.301: "Telecommunication Management; Configuration Management (CM); Notification Integration Reference Point (IRP): Requirements".

[10] 3GPP TS 32.111-3: "Telecommunication management; Fault Management; Part 3: Alarm Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS)".

[11] 3GPP TS 32.311: "Telecommunication management; Generic Integration Reference Point (IRP) management: Requirements".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TS 32.101 [7], 3GPP TS 32.102 [8] and 3GPP TS 32.301 [9] and the following apply:

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)
EM	Element Manager
IDL	Interface Definition Language (OMG)
IOR	Interoperable Object Reference
IS	Information Service
NC	Notification Channel (OMG)
NE	Network Element
NV	Name and Value pair
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]) and Structured Events.

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
```

4.1.3 Support of filterable and non-filterable notification parameters

The OMG Notification service defined IDL `CosNotification::StructuredEvent` and `CosNotification::EventBatch` data types are shown below.

```
struct StructuredEvent {
    EventHeader header;
    FilterableEventBody filterable_data;
    any remainder_of_body;
}; // StructuredEvent

typedef sequence<StructuredEvent> EventBatch;
```

Notification IS parameters are mapped:

- either to the Structured Event header, i.e. above IDL `StructuredEvent` data structure field `header`;
- or to the Structured Event body, and in this case:
  - when defined in the IS as filterable, to the Structured Event filterable body fields, i.e. above IDL `StructuredEvent` data structure field `filterable_data`;
  - when defined in the IS as non-filterable, to the Structured Event remaining body, i.e. above IDL `StructuredEvent` data structure field `remainder_of_body`.

The OMG Notification service defined IDL `CosNotification::FilterableEventBody` data type and its supporting types are shown below.

```
struct Property {
    PropertyName name;
    PropertyValue value;
};

typedef sequence<Property> PropertySeq;

typedef PropertySeq FilterableEventBody;
```

In order to ensure uniform implementation for notification IS parameters mapped to Structured Event Name-Value pairs whether defined in the IS as filterable or as non-filterable, IDL `StructuredEvent` data structure field `remainder_of_body` of type `any` shall be mapped to the IDL data structure `NotificationIRPNotifications::NonFilterableEventBody` defined in annex clause A.5:

```
struct NonFilterableEventBody {
    CosNotification::PropertySeq name_value_pairs;
    any remainder_of_non_filterable_body;
};
```

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 3GPP TS 32.302 [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 3GPP TS 32.302 [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.

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_versions	M
getSubscriptionStatus	get_subscription_status	O
getSubscriptionIds	get_subscription_ids	O
changeSubscriptionFilter	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). 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 IRP Agent supports the optional attach_push_b method, it shall support this method as mandatory. If subscription is established using attach_pull method, the SS equivalent s hall 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 IRP Agent supports the optional attach_pull method, it shall support this method as mandatory.	See box on the left.
suspendSubscription	If subscription is established using attach_push, there is no SS equivalent. In other words, IRPManager cannot suspend subscription. If subscription is established using attach_push_b, the SS equivalent shall be suspend_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]). The IDL specification of this method is not included in annex A. If IRP Agent supports the optional attach_push_b method, it shall support this method as mandatory. If subscription is established using attach_pull, there is no SS equivalent.	See box on the left.
resumeSubscription	If subscription is established using attach_push, there is no SS equivalent. In other words, IRPManager cannot resume subscription. If subscription is established using attach_push_b, the SS equivalent shall be resume_connection. This method is defined by OMG Notification Service (OMG TC Document telecom [2]). The IDL specification of this method is not included in annex A. If IRP Agent supports the optional attach_push_b method, it shall support this method as mandatory. If subscription is established using attach_pull, there is no SS equivalent.	See box on the left.
getNotificationCategories	get_notification_categories	O
getOperationProfile (see note.)	get_notification_irp_operations_profile	O
getNotificationProfile (see note.)	get_notification_irp_notification_profile	O
NOTE: These 3 operations are operations of ManagedGenericIRP IOC specified in 3GPP TS 32.312 [4]. The NotificationIRP IOC of 3GPP TS 32.302 [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
managerReference	string manager_reference (see note 1)	M
timeTick	ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick	O
notificationCategories	NotificationIRPConstDefs::NotificationCategorySetOpt notification_categories	O
filter	ManagedGenericIRPConstDefs::StringOpt filter (see note 2)	O
subscriptionId	Return value of type NotificationIRPConstDefs::SubscriptionId	M
status	Attach, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported	M
NOTE 1: IRPManager creates a CosNotifyComm::SequencePushConsumer object and invokes CORBA::ORB::object_to_string to obtain the stringified IOR, say s1. IRPManager stores the s1. IRPManager sends s1 as input parameter of attach_push to IRPAgent. IRPAgent receives s1, performs CORBA::ORB::string_to_object to obtain the IRPManager's IOR and uses it for its future methods. IRPAgent also stores the s1 for future comparisons. IRPManager later calls detach with s1. IRPAgent receives the stringified IOR s1, compares it with those stored stringified IORs (e.g. s1), finds a match, and performs the detach process. IRPAgent pushes sequence of Structured Events towards IRPManager via the CosNotifyComm::SequencePushConsumer object push_structured_events method, depending on the supplied notification categories and filter. NOTE 2: The grammar of the filter string is extended_TCL 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
managerReference	string manager_reference (see note 1)	M
timeTick	ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick	O
notificationCategories	NotificationIRPConstDefs::NotificationCategorySetOpt notification_categories	O
filter	ManagedGenericIRPConstDefs::StringOpt filter (see note 2)	O
subscriptionId	Return value of type NotificationIRPConstDefs::SubscriptionId	M
Not specified in IS.	CosNotifyChannelAdmin::SequenceProxyPushSupplier system_reference (see note 3)	M
status	Attach, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported	M
NOTE 1: IRPManager creates a CosNotifyComm::SequencePushConsumer object and invokes CORBA::ORB::object_to_string to obtain the stringified IOR, say s1. IRPManager stores the s1. IRPManager sends s1 as input parameter of attach_push_b to IRPAgent. IRPAgent receives s1 and stores the s1 for future comparisons. IRPManager later calls detach with s1. IRPAgent receives the stringified IOR s1, compares it with those stored stringified IORs (e.g. s1), finds a match, and performs the detach process. NOTE 2: The grammar of the filter string is extended_TCL 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 CosNotifyChannelAdmin::SequenceProxyPushSupplier and CosNotifyComm::SequencePushConsumer. IRPManager is expected to invoke connect_sequence_push_consumer method of this interface to connect its own cosNotifyComm::SequencePushConsummer with this reference. After successful connection, IRPAgent pushes sequence of Structured Events towards IRPManager.		

Table 4: Mapping from IS `subscribe` parameters to SS `attach_pull` equivalents

IS Operation parameter	SS Method parameter	Qualifier
managerReference	string manager_reference (see note 1)	M
timeTick	ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick	O
notificationCategories	NotificationIRPConstDefs::NotificationCategorySetOpt notification_categories	O
filter	ManagedGenericIRPConstDefs::StringOpt filter (see note 2)	O
subscriptionId	Return value of type NotificationIRPConstDefs::SubscriptionId	M
Not specified in IS.	CosNotifyChannelAdmin::SequenceProxyPullSupplier system_reference (see note 3)	M
status	Attach, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported	M

NOTE 1: IRPManager creates a CosNotifyComm::SequencePullConsumer object and invokes CORBA::ORB::object_to_string to obtain the stringified IOR, say s1. IRPManager stores the s1. IRPManager sends s1 as input parameter of attach_pull to IRPAgent. IRPAgent receives s1 and stores the s1 for future comparisons. IRPManager later calls detach with s1. IRPAgent receives the stringified IOR s1, compares it with those stored stringified IORs (e.g. s1), finds a match, and performs the detach process.

NOTE 2: The grammar of the filter string is extended_TCL 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 CosNotifyChannelAdmin::SequenceProxyPullSupplier and CosNotifyComm::SequencePullConsumer. IRPManager is expected to invoke connect_sequence_pull_consumer method of this interface to connect its own CosNotifyComm::SequencePullConsummer 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
managerReference	string manager_reference	M
subscriptionId	NotificationIRPConstDefs::SubscriptionIdOpt subscription_id	O
status	DetachException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table 6: Mapping from IS `getIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
versionNumberSet	Return value of type ManagedGenericIRPConstDefs::VersionNumberSet	M
status	GetNotificationIRPVersions	M

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

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	NotificationIRPConstDefs::SubscriptionId subscription_id	M
notificationCategorySet	Return value of type NotificationIRPConstDefs::NotificationCategorySet	M
filterInEffect	ManagedGenericIRPConstDefs::StringOpt filter_in_effect	O
subscriptionState	NotificationIRPConstDef::SubscriptionStateOpt subscription_state	O
timeTick	ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick	O
status	GetSubscriptionStatus, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

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

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

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

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

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

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

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

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

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

IS Operation parameter	SS Method parameter	Qualifier
notificationCategoryList	Return value of type NotificationIRPConstDefs::NotificationCategorySet	M
Not specified in IS.	NotificationIRPConstDefs::NotificationTypesSetOpt notification_type_list	O
status	GetNotificationCategories, ManagedGenericIRPSystem::OperationNotSupported	M

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

IS Operation parameter	SS Method parameter	Qualifier
iRPVersion	ManagedGenericIRPConstDefs::VersionNumber notification_irp_version	M
operationNameProfile, operationParameterProfile	Return of type ManagedGenericIRPConstDefs::MethodList	M
status	GetNotificationIRPOperationsProfile, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

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

IS Operation parameter	SS Method parameter	Qualifier
iRPVersion	ManagedGenericIRPConstDefs::VersionNumber notification_irp_version	M
notificationNameProfile, notificationParameterProfile	Return value of type ManagedGenericIRPConstDefs::MethodList	M
status	GetNotificationIRPNotificationProfile, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

5.3 Notification 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 IRPAgent 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): IDL specifications

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

```
//File: ManagedGenericIRPConstDefs.idl
#ifndef _MANAGED_GENERIC_IRP_CONST_DEFS_IDL_
#define _MANAGED_GENERIC_IRP_CONST_DEFS_IDL_

#include <TimeBase.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, PARTIAL_FAILURE};

   /*
   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;

   /*
   StringOpt is a type carrying an optional parameter.
   If the boolean is TRUE, then the value is present.
   Otherwise the value is absent.
   */
   union StringOpt switch (boolean)
```

```
    {
       case TRUE: string value;
    };

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

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

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

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

};

#endif // _MANAGED_GENERIC_IRP_CONST_DEFS_IDL_
```

A.2 IDL specification (file name "ManagedGenericIRPSystem.idl")

```
//File: ManagedGenericIRPSystem.idl
#ifndef _MANAGED_GENERIC_IRP_SYSTEM_IDL_
#define _MANAGED_GENERIC_IRP_SYSTEM_IDL_

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

/* ## 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 a valid but unsupported parameter value is passed.
   The parameter shall be the actual parameter name.
   */
   exception ValueNotSupported { string parameter; };

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

};

#endif // _MANAGED_GENERIC_IRP_SYSTEM_IDL_
```

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

```
//File: NotificationIRPConstDefs.idl
#ifndef _NOTIFICATION_IRP_CONST_DEFS_IDL_
#define _NOTIFICATION_IRP_CONST_DEFS_IDL_

#include <ManagedGenericIRPConstDefs.idl>

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

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

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

   /*
   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;

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

   /*
   It defines a sequence of SubscriptionIds.
   */
```

```
   typedef string SubscriptionId;
   typedef sequence <SubscriptionId> SubscriptionIdSet;

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

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

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

};

#endif // _NOTIFICATION_IRP_CONST_DEFS_IDL_
```

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

```
//File: NotificationIRPSystem.idl
#ifndef _NOTIFICATION_IRP_SYSTEM_IDL_
#define _NOTIFICATION_IRP_SYSTEM_IDL_

#include <CosNotifyChannelAdmin.idl>
#include <ManagedGenericIRPConstDefs.idl>
#include <ManagedGenericIRPSystem.idl>
#include <NotificationIRPConstDefs.idl>

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

/* ## Module: NotificationIRPSystem
This module implements capabilities of Notification IRP.
================================================================
*/
module NotificationIRPSystem
{
   /*
   System fails to complete the operation. System can provide reason
   to qualify the exception. The semantics carried in reason
   is outside the scope of this IRP.
   */
   exception GetNotificationIRPVersions { string reason; };
   exception GetNotificationIRPOperationsProfile { string reason; };
   exception GetNotificationIRPNotificationProfile { string reason; };
   exception Attach { string reason; };
   exception DetachException { string reason; };
   exception GetSubscriptionStatus { string reason; };
   exception ChangeSubscriptionFilter { string reason; };
   exception GetNotificationCategories { string reason; };
   exception GetSubscriptionIds { string reason; };

   exception AlreadySubscribed {};
   exception AtLeastOneNotificationCategoryNotSupported {};

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

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

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

```
      in ManagedGenericIRPConstDefs::VersionNumber
         notification_irp_version
)
raises (GetNotificationIRPNotificationProfile,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

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

NotificationIRPConstDefs::SubscriptionId attach_push (
   in string manager_reference,
   in ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick,
   in NotificationIRPConstDefs::NotificationCategorySetOpt
      notification_categories,
   in ManagedGenericIRPConstDefs::StringOpt filter
)
raises (Attach, ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter, AlreadySubscribed,
        AtLeastOneNotificationCategoryNotSupported);

NotificationIRPConstDefs::SubscriptionId attach_push_b (
   in string manager_reference,
   in ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick,
   in NotificationIRPConstDefs::NotificationCategorySetOpt
      notification_categories,
   in ManagedGenericIRPConstDefs::StringOpt 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 ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick,
   in NotificationIRPConstDefs::NotificationCategorySetOpt
      notification_categories,
   in ManagedGenericIRPConstDefs::StringOpt filter,
   out CosNotifyChannelAdmin::SequenceProxyPullSupplier system_reference
)
raises (Attach, ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter,
        AlreadySubscribed, AtLeastOneNotificationCategoryNotSupported);

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

/*
Check the current state of the subscription.
*/
NotificationIRPConstDefs::NotificationCategorySet get_subscription_status
(
   in NotificationIRPConstDefs::SubscriptionId subscription_id,
   out ManagedGenericIRPConstDefs::StringOpt filter_in_effect,
   out NotificationIRPConstDefs::SubscriptionStateOpt subscription_state,
   out ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick
)
raises (GetSubscriptionStatus,
```

```
                ManagedGenericIRPSystem::OperationNotSupported,
                ManagedGenericIRPSystem::InvalidParameter);

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

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

};

#endif // _NOTIFICATION_IRP_SYSTEM_IDL_
```

A.5 IDL specification (file name "NotificationIRPNotifications.idl")

```
//File: NotificationIRPNotifications.idl
#ifndef _NOTIFICATION_IRP_NOTIFICATIONS_IDL_
#define _NOTIFICATION_IRP_NOTIFICATIONS_IDL_

#include <CosNotification.idl>
#include <NotificationIRPConstDefs.idl>

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

module NotificationIRPNotifications
{

   interface Notify
   {
      /**
      Notification IRP IS defines 6 attributes for the notification header.
      They are: objectClass, objectInstance, notificationId, eventTime,
      systemDN and notificationType.

      The first 2 attributes are mapped into 1 name-value pair. The name of
      the mapped IDL construct is MANAGED_OBJECT_INSTANCE. The const
      string of this mapped IDL construct is defined here.

      The notificationId, eventTime and systemDN are respectively mapped
      into 3 name-value pairs. The const string(s) of these 3 mapped IDL
      constructs are defined here.

      The notificationType is not mapped into any name-value pair
      but is mapped into the type_name position-dependent
      field of the CORBA structured-event. There is no need for a const string
      definition for it.
      */

      const string MANAGED_OBJECT_INSTANCE =
         NotificationIRPConstDefs::AttributeNameValue::MANAGED_OBJECT_INSTANCE;

      const string NOTIFICATION_ID =
         NotificationIRPConstDefs::AttributeNameValue::NOTIFICATION_ID;

      const string EVENT_TIME =
         NotificationIRPConstDefs::AttributeNameValue::EVENT_TIME;

      const string SYSTEM_DN =
         NotificationIRPConstDefs::AttributeNameValue::SYSTEM_DN;
   };

   /**
   Type to which OMG CosNotification::StructuredEvent remainder_of_body any is to be mapped
   */

   struct NonFilterableEventBody {
      CosNotification::PropertySeq name_value_pairs;
      any remainder_of_non_filterable_body;
   };

};

#endif // _NOTIFICATION_IRP_NOTIFICATIONS_IDL_
```

Annex B (informative): Change history

Change history								
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Cat	Old	New
Jun 2001	S_12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	--	2.0.0	4.0.0
Sep 2001	S_13	SP-010522	0001	--	Eliminate guesses on IDL file names in Notification IRP: CORBA SS	F	4.0.0	4.1.0
Mar 2002	S_15	SP-020038	0002	--	Addition of missing generic CORBA exception "ValueNotSupported" into CORBA module "ManagedGenericIRPSystem"	F	4.1.0	4.2.0
Mar 2002	S_15	--	--	--	Automatic upgrade to Rel-5 (no Rel-5 CR)	--	4.2.0	5.0.0
Sep 2002	S_17	SP-020482	0004	--	Corrections to CORBA IDL specification "NotificationIRPSystem"	A	5.0.0	5.1.0
Sep 2002	S_17	SP-020479	0005	--	Add optional parameters in CORBA Solution Set	F	5.0.0	5.1.0
Sep 2002	--	--	--	--	Corrected history box CR# & TS version#	--	5.1.0	5.1.1
Dec 2002	--	--	--	--	Cosmetics	--	5.1.1	5.1.2
Mar 2003	S_19	SP-030137	0008	--	Remove unused suspend_subscription and resume_subscription methods	A	5.1.2	5.2.0
Mar 2003	S_19	SP-030137	0009	--	Corrections of CORBA IDL syntax errors	F	5.1.2	5.2.0
Mar 2003	S_19	SP-030064	0010	--	Update the usage IRP_VERSION in line with adopted release 5 policy - alignment with 32.111-3	F	5.1.2	5.2.0
Mar 2004	S_23	SP-040105	--	--	Automatic upgrade to Rel-6 (no CR)	--	5.2.0	6.0.0
Sep 2004	S_25	SP-040562	0012	--	Update 32.303 using IDL Style Guide	F	6.0.0	6.1.0
Dec 2004	S_26	SP-040793	0013	--	Remove filter requirement in IDL comments in the Notification IRP CORBA SS	F	6.1.0	6.2.0
Mar 2005	S_27	SP-050035	0014	--	Generic System Context, update of reference to IS specification	F	6.2.0	6.3.0
Jun 2005	S_28	SP-050286	0015	--	Add missing type for support of Structured Event non-filterable Name-Value pairs	F	6.3.0	6.4.0
Sep 2005	SA_29	SP-050461	0016	--	Align the CORBA SS IDL with TS 32.150 Style Guide	F	6.4.0	6.5.0
Dec 2005	SA_30	SP-050710	0017	--	Correct CORBA method optional parameter types to optional CORBA types	F	6.5.0	6.6.0
Jun 2007	SA_36	--	--	--	Automatic upgrade to Rel-7 (no CR) at freeze of Rel-7. Deleted reference to CMIP SS, discontinued from R7 onwards.	--	6.6.0	7.0.0
Dec 2008	SA_42	--	--	--	Upgrade to Release 8	--	7.0.0	8.0.0
Dec 2009	-	-	-	-	Update to Rel-9 version (MCC)	-	8.0.0	9.0.0