ETSI TS 132 306 V13.1.0 (2016-08)

Universal Mobile Telecommunications System (UMTS);
LTE;
Telecommunication management;
Configuration Management (CM);
Notification Integration Reference Point (IRP):
Solution Set (SS) definitions
(3GPP TS 32.306 version 13.1.0 Release 13)

Reference
RTS/TSGS-0532306vd10

Keywords
LTE,UMTS

ETSI

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

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

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

Intellectual Property Rights

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

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

Foreword

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

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

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification 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)

32.306: Configuration Management (CM); Notification Integration Reference Point (IRP): Solution Set (SS) definitions

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 [1].

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, e.g. alarms or CM notification.

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 [20], and OMG Notification Service [14]. 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 contains the Solution Sets for the IRP whose semantics is specified in Notification IRP: Information Service (TS 32.302 [7]).

These Solution Set specifications are related to 3GPP TS 32.302 V13.0.X [7].

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

[2] 3GPP TS 32.102: "Telecommunication management; Architecture".

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

[4] 3GPP TS 32.111-6: "Telecommunication management; Fault Management; Alarm Integration Reference Point (IRP); Solution Set (SS) definitions".

[5] 3GPP TS 32.300: "Telecommunication management; Configuration Management (CM); Name convention for Managed Objects".

[6] 3GPP TS 32.301: "Telecommunication Management; Configuration Management (CM); Notification Integration Reference Point (IRP): Requirements".

[7] 3GPP TS 32.302: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Information Service (IS)".

[8] 3GPP TS 32.311: "Telecommunication management; Generic Integration Reference Point (IRP) management; Requirements".

[9] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[10] 3GPP TS 32.316: "Telecommunication management; Generic Integration Reference Point (IRP) management; Solution Set (SS) definitions".

[11] 3GPP TS 32.331: "Telecommunication management; Notification Log (NL) Integration Reference Point (IRP): Requirements".

[12] 3GPP TS 32.336: "Telecommunication management; Notification Log (NL) Integration Reference Point (IRP); Solution Set (SS) definitions".

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

[14] OMG TC Document telecom/98-11-01: "OMG Notification Service". http://www.omg.org/technology/documents/

[15] OMG CORBA Services: "Common Object Services Specification, Update: November 22, 1996" (Clause 4 contains the Event Service specification). http://www.omg.org/technology/documents/

[16] W3C SOAP 1.1 specification (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

[17] W3C SOAP 1.2 specification (http://www.w3.org/TR/soap12-part1/)

[18] W3C WSDL 1.1 specification (http://www.w3.org/TR/2001/NOTE-wsdl-20010315)

[19] W3C XPath 1.0 specification (http://www.w3.org/TR/1999/REC-xpath-19991116)

[20] ITU-T Recommendation X.734 (1992): "Information technology - Open Systems Interconnection - Systems management: Event report management function".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TS 32.101 [1], 3GPP TS 32.102 [2], 3GPP TS 32.301 [6], 3GPP TS 32.302 [7], and 3GPP TS 32.331 [11] apply.

IRP document version number string (or "IRPVersion"): See 3GPP TS 32.311 [8]

3.2 Abbreviations

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

CM	Configuration Management
CORBA	Common Object Request Broker Architecture
EC	Event channel (OMG)
EM	Element Manager
IDL	Interface Definition Language (OMG)
IOR	Interoperable Object Reference
IRP	Integration Reference Point
IS	Information Service
MOC	Managed Object Class
MOI	Managed Object Instance
NC	Notification Channel (OMG)
NE	Network Element
NL	Notification Log
NRM	Network Resource Model
NV	Name and Value pair
OMG	Object Management Group
QoS	Quality of Service
SS	Solution Set
TMN	Telecommunications Management Network
UML	Unified Modelling Language
WSDL	Web Service Description Language
WS-I	Web Services Interoperability Organization
XML	eXtensible Markup Language

4 Solution Set Definitions

This specification defines the following 3GPP Notification IRP Solution Set Definitions:

Annex A provides the CORBA Solution Set.
Annex B provides the XML Definitions.
Annex C provides the SOAP Solution Set.

Annex A (normative): CORBA Solution Set

This annex contains the CORBA Solution Set for the IRP whose semantics is specified in Notification IRP: Information Service (TS 32.302 [7]).

A.1 Architectural Features

The overall architectural feature of the Notification IRP is specified in 3G TS 32.302 [7]. This clause specifies features that are specific to the CORBA SS.

A.1.1 Syntax for Distinguished Names

The syntax of a Distinguished Name is defined in 3GPP TS 32.300 [5].

A.1.2 Notification Services

In the CORBA Solution Set, notifications are emitted by IRPAgent using CORBA Notification service (OMG TC Document telecom [13]) and Structured Events.

CORBA Event service (OMG CORBA services [15]) 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 [13]) defined methods, is identified in the present.

A.1.3 Support of Push and Pull Style

OMG Notification Service defines two styles of interaction. One is called push style. In this style, IRPAgent pushes notifications to IRPManager as soon as they are available. The other is called pull style. In this style, IRPAgent keeps the notifications till IRPManager requests for them.

This CORBA SS specifies that support of Push style is Mandatory (M) and that support of Pull style is Optional (O).

A.1.4 Support of multiple notifications in one `push` operation

For efficiency, IRPAgent uses the following OMG Notification Service (OMG TC Document telecom [13]) 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 [13]). 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 [13]) defined IDL module is shown below.

```
module CosNotifyComm {

        …

        Interface SequencePushConsumer : NotifyPublish {

                void push_structured_events(

                        in CosNotification::EventBatch notifications)

                raises( CosEventComm::Disconnected);

                        …

            }; // SequencePushConsumer

…

}; // CosNotifyComm
```

A.1.5 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.4.6:

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

A.2 Mapping

A.2.1 Operation mapping

Notification IRP: IS (3GPP TS 32.302 [7]) defines semantics of operations visible across this IRP. These operations are the operations of the IOCs defined in 3GPP TS 32.302 [7].

Table 1 maps the operations defined in Notification IRP: IS (3GPP TS 32.302 [7]) 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 [7] to their equivalents in this SS. Since one of the IOCs, the `NotificationIRP` IOC, inherits from the `ManagedGenericIRP` IOC [9], 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 [7]	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.4. 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.4. If IRPAgent 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 shall be modify_constraints. The method is defined by OMG Notification Service Filter Interface (OMG TC Document telecom [13]). The IDL specification of this method is not included in annex A.4. If IRPAgent 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 [13]). The IDL specification of this method is not included in annex A.4. If IRPAgent 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 [13]). The IDL specification of this method is not included in annex A.4. If IRPAgent 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 [9]. The NotificationIRP IOC of 3GPP TS 32.302 [7] inherits from it.		

A.2.2 Operation parameter mapping

3GPP TS 32.302 [7] 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 [4] 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 [13]). This SS and the Alarm IRP: CORBA SS [4] 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 [13]). This SS and the Alarm IRP: CORBA SS [4] 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 [13]) 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 [13]) 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 [13]) 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 [13]) 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 [13]) 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 [13]) 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

A.2.3 Notification parameter mapping

Notification IRP: IS (3GPP TS 32.302 [7]) 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 [3]) 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.

A.3 IRPAgent's Behaviour

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

A.3.1 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.

A.3.2 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`.

A.3.3 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.

A.3.4 Quality of Service Parameters

The OMG Notification Service [13] 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) [13].
2. The message priority shall be set to 0, i.e. no priority [13].
3. The Start Time Supported shall be set to false, i.e. do not use Start Time [13].
4. The Stop Time Supported shall be set to false, i.e. do not use Stop Time [13].

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.

A.4 Solution Set definitions

A.4.1 IDL definition structure

Clause A.4.2 defines the constants and types Generic IRP Operations [9] used by the Notification IRP.

Clause A.4.3 defines the Generic IRP Operations as defined by Generic NRM IRP [9] and performed by the Notification IRP Agent.

Clause A.4.4 defines the constants and types used by the Notification IRP.

Clause A.4.5 defines the operations that are performed by the Notification IRP agent.

Clause A.4.6 defines the notifications header to be used by IRP agent when emitting notifications.

A.4.2 IDL specification "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.4.3 IDL specification '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.4.4 IDL specification '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.4.5 IDL specification '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.4.6 IDL specification "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 (normative): XML Definitions

This annex contains the XML Definitions for the Notification Integration Reference Point (Notification IRP) as it applies to Itf-N, in accordance with Notification IRP IS definitions [7].

This XML Definitions specification defines the Notification IRP XML Notification header format.

This specification is related to 3GPP TS 32.302 V10.0.X [7].

B.1 Architectural Features

The overall architectural feature of Notification IRP is specified in 3G TS 32.302 [7]. This clause specifies features that are specific to the XML definitions.

B.1.1 Syntax for Distinguished Names

The syntax of a Distinguished Name is defined in 3GPP TS 32.300 [5].

B.2 Mapping

Not present in the current version of this specification.

B.3 Solution Set definitions

B.3.1 Notification IRP XML notification header Definitions

This clause provides XML definitions of Notification IRP notification header as defined in 3GPP TS 32.302 [3]. These definitions are to be used in conjunction with Notification Log IRP XML Definitions for Notification Log IRP XML Data File and the NL IRP XML Notification Format (3GPP TS 32.335 [4]).

The structure of the Notification IRP XML Notification header Format is shown in clause B.3.2.

The Notification IRP XML Notification header Format uses the definition of notifications in clause B.3.3.

B.3.2 Graphical Representation

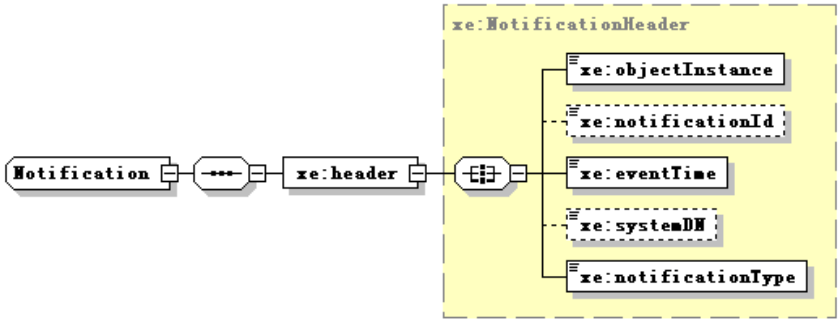

Figure B-1: Notification IRP XML Notification header WSDL structure

B.3.3 XML Schema 'notification.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>

<!--

    3GPP TS 32.306 Notification IRP

    3GPP Notification IRP XML Solution Definitions, Schema Definition

    notification.xsd

-->

<schema targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#notification"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xe="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#notification"
xmlns:xn="http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm" elementFormDefault="qualified"
attributeFormDefault="unqualified">

<import namespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"/>

    <simpleType name="NotificationId">

        <restriction base="long"/>

    </simpleType>

    <complexType name="NotificationHeader">

        <all>

            <element name="objectInstance" type="xn:dn"/>

            <element name="notificationId" type="xe:NotificationId" minOccurs="0"/>

            <element name="eventTime" type="dateTime"/>

            <element name="systemDN" type="xn:dn" minOccurs="0"/>

            <element name="notificationType" type="string"/>

        </all>

    </complexType>

    <complexType name="Notification" abstract="true">

        <sequence>

            <element name="header" type="xe:NotificationHeader"/>

        </sequence>

    </complexType>

    <element name="Notification" type="xe:Notification"/>

</schema>
```

Annex C (normative): SOAP Solution Set

This annex specifies the SOAP Solution Set for the IRP whose semantics are specified in Notification IRP: Information Service (3GPP TS 32.302 [7]).

C.1 Architectural features

The overall architectural feature of Notification IRP is specified in 3GPP TS 32.302 [7]. This clause specifies features that are specific to the SOAP solution set.

The SOAP 1.1 specification [16] and WSDL 1.1 specification [18] are supported.

The SOAP 1.2 specification [17] is supported optionally.

This specification uses "document" style in WSDL file.

This specification uses "literal" encoding style in WSDL file.

The IRPAgent shall support the push interface model that means IRPAgent sends notifications to IRPManager as soon as new events occur. IRPManager does not need to check ("pull") for events.

The 'Notification' definition is imported from Notification IRP XML Definition, documented in Annex B of this specification.

This specification uses a number of namespace prefixes throughout that are listed in the Table below.

Prefixes and Namespaces used in this specification

PREFIX	NAMESPACE
http	http://schemas.xmlsoap.org/wsdl/http/
soap	http://schemas.xmlsoap.org/wsdl/soap/
SOAP-ENV	http://schemas.xmlsoap.org/soap/envelope/
SOAP-ENC or soapenc	http://schemas.xmlsoap.org/soap/encoding/
xs or xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
ntfIRPSystem	http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPSystem
ntfIRPData	http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPData
ntfIRPNtfSystem	http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfSystem
ntfIRPNtfData	http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfData
genericIRPSystem	http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRPSystem

The WSDL structure is as outlined in the figures below:

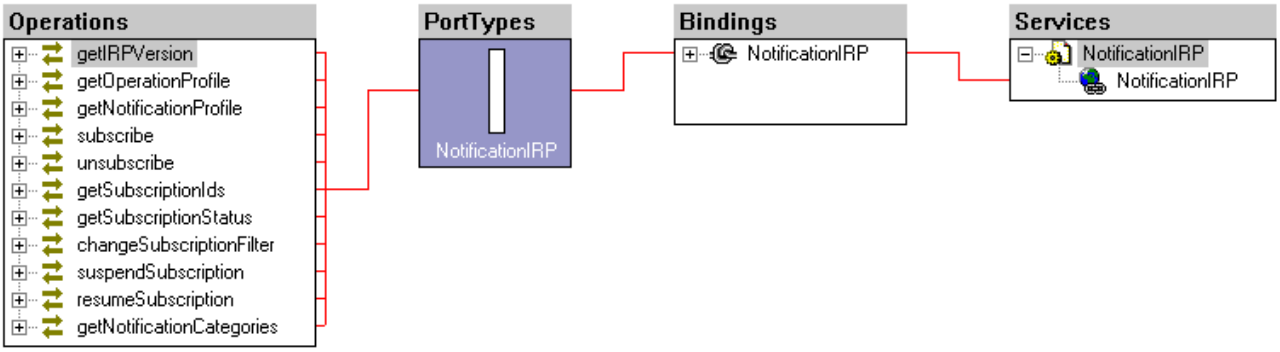

Figure C-1: Notification IRP SOAP Solution Set WSDL structure

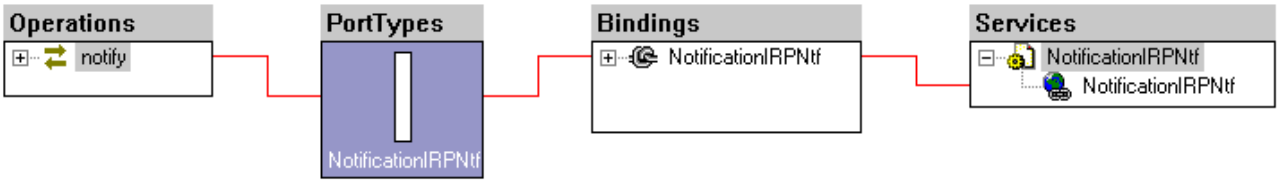

Figure C-2: Notification IRP SOAP Solution Set Notification WSDL structure

C.1.2 Syntax for Distinguished Names

The syntax of a Distinguished Name is defined in 3GPP TS 32.300 [5].

C.2 Mapping

C.2.1 Operation mapping

Notification IRP: IS (3GPP TS 32.302 [7]) defines semantics of operations visible across this IRP. These operations are the operations of the IOCs defined in 3GPP TS 32.302 [7].

The table maps the operations defined in Notification IRP: IS (3GPP TS 32.302 [7]) to their equivalents (methods) in this Solution Set (SS). Specifically, it maps the operations of the IOCs defined in 3GPP TS 32.302 [7] to their equivalents in this SS. Since one of the IOCs, the `NotificationIRP` IOC, inherits from the `ManagedGenericIRP` IOC [9], the table below also maps the operations of `ManagedGenericIRP` IOC to their equivalents (methods) in this SS.

The table below also qualifies if a method is Mandatory (M) or Optional (O).

Mapping from IS Operation to SS Equivalents

IS Operations in 3GPP TS 32.302 [7]	SS Operations	Qualifier
subscribe	subscribe	M
unsubscribe	unsubscribe	M
getSubscriptionIds	getSubscriptionIds	O
getSubscriptionStatus	getSubscriptionStatus	O
changeSubscriptionFilter	changeSubscriptionFilter	O
suspendSubscription	suspendSubscription	O
resumeSubscription	resumeSubscription	O
getNotificationCategories	getNotificationCategories	O
getIRPVersion (see note.)	getIRPVersion	M
getOperationProfile (see note.)	getOperationProfile	O
getNotificationProfile (see note.)	getNotificationProfile	O
NOTE: These 3 operations are operations of ManagedGenericIRP IOC specified in 3GPP TS 32.312 [9]. The NotificationIRP IOC of 3GPP TS 32.302 [7] inherits from it.		

C.2.2 Filter language

The filter language used in the SS is the XPath Language (see W3C XPath 1.0 specification [19]). IRPAgents may throw a FilterComplexityLimit fault when a given filter is too complex.

C.2.3 Common datatype definition

C.2.3.1 NotificationCategorySetType

```
<complexType name="NotificationCategorySetType">
<complexContent>
<extension base="ntfIRPData:VersionNumberSetType">
    </extension>
</complexContent>
</complexType>
```

C.2.3.2 SubscriptionStateType

```
<simpleType name="SubscriptionStateType">
```

```
<restriction base="string">
        <enumeration value="Suspended"/>
        <enumeration value="NotSuspended"/>
    </restriction>
</simpleType>
```

C.2.4 Operation parameter mapping

3GPP TS 32.302 [7] defines semantics of parameters carried in operations across the Notification IRP. The tables below show the mapping of these parameters, as per operation, to their equivalents defined in this SS.

C.2.4.1 Operation `subscribe`

C.2.4.1.1 Input parameters

Mapping from IS subscribe input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
managerReference	anyURI managerReference	M
timeTick	long timeTick	O
notificationCategories	ntfIRPData:NotificationCategorySetType notificationCategories	O
filter	string filter (see note 1)	O
	`anyURI` ntfTransServiceNS (see note 2)	M
NOTE1: The grammar of the filter string is XPath defined by W3C XPath 1.0 specification [19].		
NOTE2: This parameter is used to specify the name space of the Web Notification Transmission Service that will be used for sending notification. The 'http://www.3gpp.org/ftp/specs/archive/32_series/32.306#notification' URI should be supported.		

Here is the XML schema fragment of the subscribe request:

```
<!-- subscribe Request -->
<element name="subscribe">
<complexType>
<sequence>
   <element name="managerReference" type="anyURI"/>
   <element name="timeTick" type="long" minOccurs="0"/>
   <element name="notificationCategories" type="ntfIRPData:NotificationCategorySetType" minOccurs="0"/>
   <element name="filter" type="string" minOccurs="0"/>
   <element name="ntfTransServiceNS" type="anyURI"/>
</sequence>
</complexType>
</element>
```

C.2.4.1.2 Output parameters

Mapping from IS subscribe output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscriptionId	M
status	ntfIRPData:subscribeFault	M

Here is the XML schema fragment of the subscribe response:

```
<!-- subscribe Response -->
<element name="subscribeResponse">
<complexType>
<sequence>
   <element name="subscriptionId" type="string"/>
</sequence>
</complexType>
</element>
```

C.2.4.1.3 Fault definition

```
<!-- subscribe Fault -->
<element name="subscribeFault">
<complexType>
<choice>
   <element name="AlreadySubscribedFault" type="string"/>
   <element name="AtLeastOneNotificationCategoryNotSupportedFault" type="string"/>
   <element name="subscribeFault" type="string"/>
   <element ref="ntfIRPData:InvalidParameterFault"/>
   <element ref="ntfIRPData:ParameterNotSupportedFault"/>
</choice>
</complexType>
</element>
```

C.2.4.2 Operation unsubscribe

C.2.4.2.1 Input parameters

Mapping from IS unsubscribe input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
managerReference	anyURI managerReference	M
subscriptionId	string subscriptionId	O

Here is the XML schema fragment of the unsubscribe request:

```
<!-- unsubscribe Request -->
<element name="unsubscribe">
<complexType>
<sequence>
 <element name="managerReference" type="anyURI"/>
 <element name="subscriptionId" type="string" minOccurs="0"/>
</sequence>
</complexType>
</element>
```

C.2.4.2.2 Output parameters

Mapping from IS unsubscribe output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
Status	ntfIRPData:unsubscribeFault	M

Here is the XML schema fragment of the unsubscribe response:

```
<!-- unsubscribe Response -->
<element name="unsubscribeResponse">
</element>
```

C.2.4.2.3 Fault definition

```
<!-- unsubscribe Fault -->
<element name="unsubscribeFault">
<complexType>
<choice>
   <element name="unsubscribeFault" type="string"/>
   <element ref="ntfIRPData:InvalidParameterFault"/>
```

```
    <element ref="ntfIRPData:ParameterNotSupportedFault"/>
  </choice>
  </complexType>
  </element>
```

C.2.4.3 Operation getSubscriptionIds

C.2.4.3.1 Input parameters

Mapping from IS `getSubscriptionIds` input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
managerReference	anyURI managerReference	M

Here is the XML schema fragment of the getSubscriptionIds request:

```
<!-- getSubscriptionIds Request -->
<element name="getSubscriptionIds">
<complexType>
<sequence>
 <element name="managerReference" type="anyURI"/>
</sequence>
</complexType>
</element>
```

C.2.4.3.2 Output parameters

Mapping from IS `getSubscriptionIds output` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
subscriptionIdSet	sequence of string subscriptionIdSet	M
status	ntfIRPData:getSubscriptionIdsFault	M

Here is the XML schema fragment of the getSubscriptionIds response:

```
<!-- getSubscriptionIds Response -->
<element name="getSubscriptionIdsResponse">
<complexType>
    <sequence>
        <element name="subscriptionIdSet">
            <complexType>
              <sequence>
                 <element name="subscriptionId" type="string" maxOccurs="unbounded"/>
```

```
                    </sequence>
                  </complexType>
                </element>
            </sequence>
        </complexType>
        </element>
```

C.2.4.3.3 Fault definition

```
<!-- getSubscriptionIds Fault -->
<element name="getSubscriptionIdsFault">
<complexType>
<choice>
    <element name="getSubscriptionIdsFault" type="string"/>
    <element ref="ntfIRPData:OperationNotSupportedFault"/>
    <element ref="ntfIRPData:InvalidParameterFault"/>
</choice>
</complexType>
</element>
```

C.2.4.4 Operation getSubscriptionStatus

C.2.4.4.1 Input parameters

Mapping from IS getSubscriptionStatus input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscriptionId	M

Here is the XML schema fragment of the getSubscriptionStatus request:

```
<!-- getSubscriptionStatus Request -->
<element name="getSubscriptionStatus">
  <complexType>
   <sequence>
       <element name="subscriptionId" type="string"/>
   </sequence>
  </complexType>
</element>
```

C.2.4.4.2 Output parameters

Mapping from IS getSubscriptionStatus output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
notificationCategories	ntfIRPData:NotificationCategorySetType notificationCategories	C
filterInEffect	string filterInEffect (see note)	O
SubscriptionState	ntfIRPData:SubscriptionStateType SubscriptionState	O
timeTick	long timeTick	O
status	ntfIRPData:getSubscriptionStatusFault	M
NOTE: The grammar of the filter string is XPath defined by W3C XPath 1.0 specification [19].		

Here is the XML schema fragment of the getSubscriptionStatus response:

```
<!-- getSubscriptionStatus Response -->
<element name="getSubscriptionStatusResponse">
<complexType>
<sequence>
    <element name="notificationCategories" type="ntfIRPData:NotificationCategorySetType"
    minOccurs="0"/>
 <element name="filterInEffect" type="string" minOccurs="0"/>
 <element name="SubscriptionState" type="ntfIRPData:SubscriptionStateType" minOccurs="0"/>
 <element name="timeTick" type="long" minOccurs="0"/>
</sequence>
</complexType>
</element>
```

C.2.4.4.3 Fault definition

```
<!-- getSubscriptionStatus Fault -->
<element name="getSubscriptionStatusFault">
  <complexType>
    <choice>
      <element name="getSubscriptionStatusFault" type="string"/>
      <element ref="ntfIRPData:OperationNotSupportedFault"/>
      <element ref="ntfIRPData:InvalidParameterFault"/>
    </choice>
  </complexType>
</element>
```

C.2.4.5 Operation changeSubscriptionFilter

C.2.4.5.1 Input parameters

Mapping from IS changeSubscriptionFilter input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscriptionId	M
filter	string filter (see note)	M
NOTE: The grammar of the filter string is XPath defined by W3C XPath 1.0 specification [19].		

Here is the XML schema fragment of the changeSubscriptionFilter request:

```
<!-- changeSubscriptionFilter Request -->
<element name="changeSubscriptionFilter">
<complexType>
<sequence>
    <element name="subscriptionId" type="string"/>
    <element name="filter" type="string"/>
</sequence>
</complexType>
</element>
```

C.2.4.5.2 Output parameters

Mapping from IS changeSubscriptionFilter output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
status	ntfIRPData: changeSubscriptionFilterFault	M

Here is the XML schema fragment of the changeSubscriptionFilter response:

```
<!-- changeSubscriptionFilter Response -->
<element name="changeSubscriptionFilterResponse">
</element>
```

C.2.4.5.3 Fault definition

```
<!-- changeSubscriptionFilter Fault -->
<element name="changeSubscriptionFilterFault">
 <complexType>
    <choice>
        <element name="changeSubscriptionFilterFault" type="string"/>
        <element ref="ntfIRPData:OperationNotSupportedFault"/>
```

```
            <element ref="ntfIRPData:InvalidParameterFault"/>
        </choice>
    </complexType>
</element>
```

C.2.4.6 Operation suspendSubscription

C.2.4.6.1 Input parameters

Mapping from IS suspendSubscription input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscriptionId	M

Here is the XML schema fragment of the suspendSubscription request:

```
<!-- suspendSubscription Request -->
<element name="suspendSubscription">
    <complexType>
        <sequence>
            <element name="subscriptionId" type="string"/>
        </sequence>
    </complexType>
</element>
```

C.2.4.6.2 Output parameters

Mapping from IS suspendSubscription output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
status	ntfIRPData:suspendSubscriptionFault	M

Here is the XML schema fragment of the suspendSubscription response:

```
<!-- suspendSubscription Response -->
<element name="suspendSubscriptionResponse">
</element>
```

C.2.4.6.3 Fault definition

```
<!-- suspendSubscription Fault -->
<element name="suspendSubscriptionFault">
    <complexType>
```

```
            <choice>
                <element name="suspendSubscriptionFault" type="string"/>
                <element ref="ntfIRPData:OperationNotSupportedFault"/>
                <element ref="ntfIRPData:InvalidParameterFault"/>
            </choice>
        </complexType>
    </element>
```

C.2.4.7 Operation resumeSubscription

C.2.4.7.1 Input parameters

Mapping from IS resumeSubscription input parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
subscriptionId	string subscriptionId	M

Here is the XML schema fragment of the resumeSubscription request:

```
    <!-- resumeSubscription Request -->
    <element name="resumeSubscription">
        <complexType>
            <sequence>
                <element name="subscriptionId" type="string"/>
            </sequence>
        </complexType>
    </element>
```

C.2.4.7.2 Output parameters

Mapping from IS resumeSubscription output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
status	ntfIRPData:resumeSubscriptionFault	M

Here is the XML schema fragment of the resumeSubscription response:

```
    <!-- resumeSubscription Response -->
    <element name="resumeSubscriptionResponse">
    </element>
```

C.2.4.7.3 Fault definition

```
<!-- resumeSubscription Fault -->
<element name="resumeSubscriptionFault">
  <complexType>
    <choice>
      <element name="resumeSubscriptionFault" type="string"/>
      <element ref="ntfIRPData:OperationNotSupportedFault"/>
      <element ref="ntfIRPData:InvalidParameterFault"/>
    </choice>
  </complexType>
</element>
```

C.2.4.8 Operation getNotificationCategories

C.2.4.8.1 Input parameters

None.

Here is the XML schema fragment of the getNotificationCategories request:

```
<!-- getNotificationCategories  Request -->
<element name="getNotificationCategories">
</element>
```

C.2.4.8.2 Output parameters

Mapping from IS getNotificationCategories output parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
NotificationCategoryList	ntfIRPData:NotificationCategorySetType NotificationCategoryList	M
status	ntfIRPData:getNotificationCategoriesFault	M

Here is the XML schema fragment of the getNotificationCategories response:

```
<!-- getNotificationCategories  Response -->
<element name="getNotificationCategoriesResponse">
  <complexType>
    <sequence>
      <element name="NotificationCategoryList" type="ntfIRPData:NotificationCategorySetType"/>
    </sequence>
  </complexType>
```

```
</element>
```

C.2.4.8.3 Fault definition

```
<!-- getNotificationCategories  Fault -->
<element name="getNotificationCategoriesFault">
    <complexType>
        <choice>
            <element name="getNotificationCategoriesFault" type="string"/>
            <element ref="ntfIRPData:OperationNotSupportedFault"/>
        </choice>
    </complexType>
</element>
```

C.2.5 Parameter mapping

Notification IRP: IS (3GPP TS 32.302 [7]) defines the semantics of common attributes carried in notifications. This SS does not provide the mapping of these attributes to their SOAP SS equivalents. Other IRPs such as Alarm IRP: IS (3GPP TS 32.111-2 [3]) 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.

C.2.6 NotificationIRPNotification Interface definition

The operation name is defined as 'notify'.

C.2.6.1 Input parameters

notify input parameters

IS Operation parameter	SS Method parameter	Qualifier
-	xe:Notification notification	M

The notifications of the various IRPs extend the input parameter above.

Here is the XML schema fragment of the notify request:

```
<!-- notify Request -->
<element name="notify">
    <complexType>
        <sequence>
            <element name="notificationHeaderAndBody"
type="ntfIRPNtfData:AnySequenceType"/>
        </sequence>
    </complexType>
```

```
</element>
<complexType name="AnySequenceType">
    <sequence>
        <any namespace="##any" processContents="lax" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

C.2.6.2 Output parameters

None.

C.3 IRPAgent's Behaviour

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

C.3.1 Subscription

IRPManager can invoke multiple `subscribe` operation using different `managerReference`(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 `subscribe` using the same `managerReference` and with an already subscribed `notificationCategory`, IRPAgent shall raise `AlreadySubscribedFault` fault to all invocations except one.

IRPManager can invoke multiple `subscribe` using the same `managerReference` and with one or more not-yet-subscribed `notificationCategories`. In this case, if IRPAgent supports all the notification categories requested, IRPAgent shall accept the invocation; otherwise, it raises `AtLeastOneNotificationCategoryNotSupportedFault` fault.

C.4 Solution Set definitions

C.4.1 WSDL definition structure

Clause C.4.2 provides a graphical representation of the Notification IRP service.

Clause C.4.3 defines the services that are supported by the Notification IRP agent.

Clause C.4.4 defines the Notification Header definitions.

C.4.2 Graphical Representation

Not present in the current version of this specification.

C.4.3 WSDL specification 'NotificationIRPSystem'

<?xml version="1.0" encoding="UTF-8"?>

<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:genericIRPSystem="http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRPSystem" xmlns:ntfIRPSystem="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPSystem"

xmlns:ntfIRPData="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPData"

targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPSystem">

<import namespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRPSystem" />

<types>

<schema targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPData" xmlns="http://www.w3.org/2001/XMLSchema">

```
        <!-- subscribe Request -->
    <element name="subscribe">
        <complexType>
            <sequence>
                <element name="managerReference" type="anyURI"/>
                <element name="timeTick" type="long" minOccurs="0"/>
                <element name="notificationCategories" type="ntfIRPData:NotificationCategorySetType"
minOccurs="0"/>
                <element name="filter" type="string" minOccurs="0"/>
                        <element name="ntfTransServiceNS" type="anyURI"/>
            </sequence>
        </complexType>
    </element>
    <!-- subscribe Response -->
    <element name="subscribeResponse">
        <complexType>
            <sequence>
                <element name="subscriptionId" type="string"/>
            </sequence>
        </complexType>
    </element>
    <complexType name="NotificationCategorySetType">
        <complexContent>
            <extension base="ntfIRPData:VersionNumberSetType">
            </extension>
        </complexContent>
    </complexType>
    <!-- subscribe Fault -->
    <element name="subscribeFault">
        <complexType>
            <choice>
                <element name="AlreadySubscribedFault" type="string"/>
                <element name="AtLeastOneNotificationCategoryNotSupportedFault" type="string"/>
                <element name="subscribeFault" type="string"/>
                <element ref="ntfIRPData:InvalidParameterFault"/>
```

```
            <element ref="ntfIRPData:ParameterNotSupportedFault"/>
        </choice>
    </complexType>
</element>
<!-- unsubscribe Request -->
<element name="unsubscribe">
    <complexType>
        <sequence>
            <element name="managerReference" type="anyURI"/>
            <element name="subscriptionId" type="string" minOccurs="0"/>
        </sequence>
    </complexType>
</element>
<!-- unsubscribe Response -->
<element name="unsubscribeResponse">
</element>
<!-- unsubscribe Fault -->
<element name="unsubscribeFault">
    <complexType>
        <choice>
            <element name="unsubscribeFault" type="string"/>
            <element ref="ntfIRPData:InvalidParameterFault"/>
            <element ref="ntfIRPData:ParameterNotSupportedFault"/>
        </choice>
    </complexType>
</element>
<!-- getSubscriptionIds Request -->
<element name="getSubscriptionIds">
    <complexType>
        <sequence>
            <element name="managerReference" type="anyURI"/>
        </sequence>
    </complexType>
</element>
<!-- getSubscriptionIds Response -->
```

```
<element name="getSubscriptionIdsResponse">
    <complexType>
        <sequence>
            <element name="subscriptionIdSet">
                <complexType>
                    <sequence>
                        <element name="subscriptionId" type="string" maxOccurs="unbounded"/>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
<!-- getSubscriptionIds Fault -->
<element name="getSubscriptionIdsFault">
    <complexType>
        <choice>
            <element name="getSubscriptionIdsFault" type="string"/>
            <element ref="ntfIRPData:OperationNotSupportedFault"/>
            <element ref="ntfIRPData:InvalidParameterFault"/>
        </choice>
    </complexType>
</element>
<!-- getSubscriptionStatus Request -->
<element name="getSubscriptionStatus">
    <complexType>
        <sequence>
            <element name="subscriptionId" type="string"/>
        </sequence>
    </complexType>
</element>
<!-- getSubscriptionStatus Response -->
<element name="getSubscriptionStatusResponse">
    <complexType>
        <sequence>
```

```
                    <element name="notificationCategories" type="ntfIRPData:NotificationCategorySetType"
minOccurs="0"/>
                    <element name="filterInEffect" type="string" minOccurs="0"/>
                    <element name="SubscriptionState" type="ntfIRPData:SubscriptionStateType" minOccurs="0"/>
                    <element name="timeTick" type="long" minOccurs="0"/>
                </sequence>
            </complexType>
        </element>
        <simpleType name="SubscriptionStateType">
            <restriction base="string">
                <enumeration value="Suspended"/>
                <enumeration value="NotSuspended"/>
            </restriction>
        </simpleType>
        <!-- getSubscriptionStatus Fault -->
        <element name="getSubscriptionStatusFault">
            <complexType>
                <choice>
                    <element name="getSubscriptionStatusFault" type="string"/>
                    <element ref="ntfIRPData:OperationNotSupportedFault"/>
                    <element ref="ntfIRPData:InvalidParameterFault"/>
                </choice>
            </complexType>
        </element>
        <!-- changeSubscriptionFilter Request -->
        <element name="changeSubscriptionFilter">
            <complexType>
                <sequence>
                    <element name="subscriptionId" type="string"/>
                    <element name="filter" type="string"/>
                </sequence>
            </complexType>
        </element>
        <!-- changeSubscriptionFilter Response -->
        <element name="changeSubscriptionFilterResponse">
```

```
</element>
<!-- changeSubscriptionFilter Fault -->
<element name="changeSubscriptionFilterFault">
    <complexType>
        <choice>
            <element name="changeSubscriptionFilterFault" type="string"/>
            <element ref="ntfIRPData:OperationNotSupportedFault"/>
            <element ref="ntfIRPData:InvalidParameterFault"/>
        </choice>
    </complexType>
</element>
<!-- suspendSubscription Request -->
<element name="suspendSubscription">
    <complexType>
        <sequence>
            <element name="subscriptionId" type="string"/>
        </sequence>
    </complexType>
</element>
<!-- suspendSubscription Response -->
<element name="suspendSubscriptionResponse">
</element>
<!-- suspendSubscription Fault -->
<element name="suspendSubscriptionFault">
    <complexType>
        <choice>
            <element name="suspendSubscriptionFault" type="string"/>
            <element ref="ntfIRPData:OperationNotSupportedFault"/>
            <element ref="ntfIRPData:InvalidParameterFault"/>
        </choice>
    </complexType>
</element>
<!-- resumeSubscription Request -->
<element name="resumeSubscription">
    <complexType>
```

```
            <sequence>
                <element name="subscriptionId" type="string"/>
            </sequence>
        </complexType>
    </element>
    <!-- resumeSubscription Response -->
    <element name="resumeSubscriptionResponse">
    </element>
    <!-- resumeSubscription Fault -->
    <element name="resumeSubscriptionFault">
        <complexType>
            <choice>
                <element name="resumeSubscriptionFault" type="string"/>
                <element ref="ntfIRPData:OperationNotSupportedFault"/>
                <element ref="ntfIRPData:InvalidParameterFault"/>
            </choice>
        </complexType>
    </element>
    <!-- getNotificationCategories  Request -->
    <element name="getNotificationCategories">
    </element>
    <!-- getNotificationCategories  Response -->
    <element name="getNotificationCategoriesResponse">
        <complexType>
            <sequence>
                <element name="NotificationCategoryList" type="ntfIRPData:NotificationCategorySetType"/>
            </sequence>
        </complexType>
    </element>
    <!-- getNotificationCategories  Fault -->
    <element name="getNotificationCategoriesFault">
        <complexType>
            <choice>
                <element name="getNotificationCategoriesFault" type="string"/>
                <element ref="ntfIRPData:OperationNotSupportedFault"/>
```

```
            </choice>
        </complexType>
    </element>

    <element name="OperationNotSupportedFault" type="string"/>
    <element name="ParameterNotSupportedFault" type="string"/>
    <element name="InvalidParameterFault" type="string"/>
    <simpleType name="VersionNumberType">
        <restriction base="string"/>
    </simpleType>
    <complexType name="VersionNumberSetType">
        <sequence>
            <element name="versionNumber" type="ntfIRPData:VersionNumberType"
maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <complexType name="ParameterSetType">
        <sequence>
            <element name="parameterName" type="string" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <complexType name="OperationType">
        <sequence>
            <element name="operationName" type="string"/>
            <element name="parameterSet" type="ntfIRPData:ParameterSetType"/>
        </sequence>
    </complexType>
    <complexType name="OperationSetType">
        <sequence>
            <element name="operation" type="ntfIRPData:OperationType" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <complexType name="NotificationType">
        <sequence>
            <element name="notificationName" type="string"/>
            <element name="parameterSet" type="ntfIRPData:ParameterSetType"/>
```

```
            </sequence>
        </complexType>
        <complexType name="NotificationSetType">
            <sequence>
                <element name="notification" type="ntfIRPData:NotificationType" maxOccurs="unbounded"/>
            </sequence>
        </complexType>

    </schema>
</types>
<message name="subscribeRequest">
    <part name="parameter" element="ntfIRPData:subscribe"/>
</message>
<message name="subscribeResponse">
    <part name="parameter" element="ntfIRPData:subscribeResponse"/>
</message>
<message name="subscribeFault">
    <part name="parameter" element="ntfIRPData:subscribeFault"/>
</message>
<message name="unsubscribeRequest">
    <part name="parameter" element="ntfIRPData:unsubscribe"/>
</message>
<message name="unsubscribeResponse">
    <part name="parameter" element="ntfIRPData:unsubscribeResponse"/>
</message>
<message name="unsubscribeFault">
    <part name="parameter" element="ntfIRPData:unsubscribeFault"/>
</message>
<message name="getSubscriptionIdsRequest">
    <part name="parameter" element="ntfIRPData:getSubscriptionIds"/>
</message>
<message name="getSubscriptionIdsResponse">
    <part name="parameter" element="ntfIRPData:getSubscriptionIdsResponse"/>
</message>
<message name="getSubscriptionIdsFault">
```

```
    <part name="parameter" element="ntfIRPData:getSubscriptionIdsFault"/>
</message>
<message name="getSubscriptionStatusRequest">
    <part name="parameter" element="ntfIRPData:getSubscriptionStatus"/>
</message>
<message name="getSubscriptionStatusResponse">
    <part name="parameter" element="ntfIRPData:getSubscriptionStatusResponse"/>
</message>
<message name="getSubscriptionStatusFault">
    <part name="parameter" element="ntfIRPData:getSubscriptionStatusFault"/>
</message>
<message name="changeSubscriptionFilterRequest">
    <part name="parameter" element="ntfIRPData:changeSubscriptionFilter"/>
</message>
<message name="changeSubscriptionFilterResponse">
    <part name="parameter" element="ntfIRPData:changeSubscriptionFilterResponse"/>
</message>
<message name="changeSubscriptionFilterFault">
    <part name="parameter" element="ntfIRPData:changeSubscriptionFilterFault"/>
</message>
<message name="suspendSubscriptionRequest">
    <part name="parameter" element="ntfIRPData:suspendSubscription"/>
</message>
<message name="suspendSubscriptionResponse">
    <part name="parameter" element="ntfIRPData:suspendSubscriptionResponse"/>
</message>
<message name="suspendSubscriptionFault">
    <part name="parameter" element="ntfIRPData:suspendSubscriptionFault"/>
</message>
<message name="resumeSubscriptionRequest">
    <part name="parameter" element="ntfIRPData:resumeSubscription"/>
</message>
<message name="resumeSubscriptionResponse">
    <part name="parameter" element="ntfIRPData:resumeSubscriptionResponse"/>
</message>
```

```
<message name="resumeSubscriptionFault">
    <part name="parameter" element="ntfIRPData:resumeSubscriptionFault"/>
</message>
<message name="getNotificationCategoriesRequest">
    <part name="parameter" element="ntfIRPData:getNotificationCategories"/>
</message>
<message name="getNotificationCategoriesResponse">
    <part name="parameter" element="ntfIRPData:getNotificationCategoriesResponse"/>
</message>
<message name="getNotificationCategoriesFault">
    <part name="parameter" element="ntfIRPData:getNotificationCategoriesFault"/>
</message>

<portType name="NotificationIRP">
    <operation name="getIRPVersion">
        <input message="genericIRPSystem:getIRPVersionRequest"/>
        <output message="genericIRPSystem:getIRPVersionResponse"/>
        <fault name="getIRPVersionFault" message="genericIRPSystem:getIRPVersionFault"/>
    </operation>
    <operation name="getOperationProfile">
        <input message="genericIRPSystem:getOperationProfileRequest"/>
        <output message="genericIRPSystem:getOperationProfileResponse"/>
        <fault name="getOperationProfileFault" message="genericIRPSystem:getOperationProfileFault"/>
    </operation>
    <operation name="getNotificationProfile">
        <input message="genericIRPSystem:getNotificationProfileRequest"/>
        <output message="genericIRPSystem:getNotificationProfileResponse"/>
        <fault name="getNotificationProfileFault" message="genericIRPSystem:getNotificationProfileFault"/>
    </operation>
    <operation name="subscribe">
        <input message="ntfIRPSystem:subscribeRequest"/>
        <output message="ntfIRPSystem:subscribeResponse"/>
        <fault name="subscribeFault" message="ntfIRPSystem:subscribeFault"/>
    </operation>
    <operation name="unsubscribe">
```

```
        <input message="ntfIRPSystem:unsubscribeRequest"/>
        <output message="ntfIRPSystem:unsubscribeResponse"/>
        <fault name="unsubscribeFault" message="ntfIRPSystem:unsubscribeFault"/>
    </operation>
    <operation name="getSubscriptionIds">
        <input message="ntfIRPSystem:getSubscriptionIdsRequest"/>
        <output message="ntfIRPSystem:getSubscriptionIdsResponse"/>
        <fault name="getSubscriptionIdsFault" message="ntfIRPSystem:getSubscriptionIdsFault"/>
    </operation>
    <operation name="getSubscriptionStatus">
        <input message="ntfIRPSystem:getSubscriptionStatusRequest"/>
        <output message="ntfIRPSystem:getSubscriptionStatusResponse"/>
        <fault name="getSubscriptionStatusFault" message="ntfIRPSystem:getSubscriptionStatusFault"/>
    </operation>
    <operation name="changeSubscriptionFilter">
        <input message="ntfIRPSystem:changeSubscriptionFilterRequest"/>
        <output message="ntfIRPSystem:changeSubscriptionFilterResponse"/>
        <fault name="changeSubscriptionFilterFault" message="ntfIRPSystem:changeSubscriptionFilterFault"/>
    </operation>
    <operation name="suspendSubscription">
        <input message="ntfIRPSystem:suspendSubscriptionRequest"/>
        <output message="ntfIRPSystem:suspendSubscriptionResponse"/>
        <fault name="suspendSubscriptionFault" message="ntfIRPSystem:suspendSubscriptionFault"/>
    </operation>
    <operation name="resumeSubscription">
        <input message="ntfIRPSystem:resumeSubscriptionRequest"/>
        <output message="ntfIRPSystem:resumeSubscriptionResponse"/>
        <fault name="resumeSubscriptionFault" message="ntfIRPSystem:resumeSubscriptionFault"/>
    </operation>
    <operation name="getNotificationCategories">
        <input message="ntfIRPSystem:getNotificationCategoriesRequest"/>
        <output message="ntfIRPSystem:getNotificationCategoriesResponse"/>
        <fault name="getNotificationCategoriesFault" message="ntfIRPSystem:getNotificationCategoriesFault"/>
    </operation>
```

```
    </portType>
    <binding name="NotificationIRP" type="ntfIRPSystem:NotificationIRP">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="getIRPVersion">
            <soap:operation soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#getIRPVersion"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="getIRPVersionFault">
                <soap:fault name="getIRPVersionFault" use="literal"/>
            </fault>
        </operation>
        <operation name="getOperationProfile">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#getOperationProfile"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="getOperationProfileFault">
                <soap:fault name="getOperationProfileFault" use="literal"/>
            </fault>
        </operation>
        <operation name="getNotificationProfile">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#getNotificationProfile"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
```

```
            <soap:body use="literal"/>
        </output>
        <fault name="getNotificationProfileFault">
            <soap:fault name="getNotificationProfileFault" use="literal"/>
        </fault>
    </operation>
    <operation name="subscribe">
        <soap:operation soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#subscribe"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
        <fault name="subscribeFault">
            <soap:fault name="subscribeFault" use="literal"/>
        </fault>
    </operation>
    <operation name="unsubscribe">
    <soap:operation soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#unsubscribe"/>

        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
        <fault name="unsubscribeFault">
            <soap:fault name="unsubscribeFault" use="literal"/>
        </fault>
    </operation>
    <operation name="getSubscriptionIds">
        <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#getSubscriptionIds"/>
        <input>
```

```
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
        <fault name="getSubscriptionIdsFault">
            <soap:fault name="getSubscriptionIdsFault" use="literal"/>
        </fault>
    </operation>
    <operation name="getSubscriptionStatus">
        <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#getSubscriptionStatus"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
        <fault name="getSubscriptionStatusFault">
            <soap:fault name="getSubscriptionStatusFault" use="literal"/>
        </fault>
    </operation>
    <operation name="changeSubscriptionFilter">
        <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#changeSubscriptionFilter"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
        <fault name="changeSubscriptionFilterFault">
            <soap:fault name="changeSubscriptionFilterFault" use="literal"/>
        </fault>
    </operation>
    <operation name="suspendSubscription">
```

```
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#suspendSubscription"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="suspendSubscriptionFault">
                <soap:fault name="suspendSubscriptionFault" use="literal"/>
            </fault>
        </operation>
        <operation name="resumeSubscription">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#resumeSubscription"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="resumeSubscriptionFault">
                <soap:fault name="resumeSubscriptionFault" use="literal"/>
            </fault>
        </operation>
        <operation name="getNotificationCategories">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#getNotificationCategories"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="getNotificationCategoriesFault">
                <soap:fault name="getNotificationCategoriesFault" use="literal"/>
```

```
            </fault>
        </operation>
    </binding>
    <service name="NotificationIRP">
        <port name="NotificationIRP" binding="ntfIRPSystem:NotificationIRP">
            <soap:address location="To be defined."/>
        </port>
    </service>
</definitions>
```

C.4.4 WSDL specification 'NotificationIRPNtfSystem'

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ntfIRPNtfSystem="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfSystem"

xmlns:ntfIRPNtfData="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfData"

targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfSystem">

    <types>

        <schema targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfData"
xmlns="http://www.w3.org/2001/XMLSchema">


            <!-- notify Request -->
            <element name="notify">
                <complexType>
                    <sequence>
                        <element name="notificationHeaderAndBody"
            type="ntfIRPNtfData:AnySequenceType"/>
                    </sequence>
                </complexType>
            </element>
            <complexType name="AnySequenceType">
                <sequence>
                    <any namespace="##any" processContents="lax" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </schema>
```

```
    </types>

    <message name="notifyRequest">
        <part name="parameter" element="ntfIRPNtfData:notify"/>
    </message>

    <portType name="NotificationIRPNtf">
        <operation name="notify">
            <input message="ntfIRPNtfSystem:notifyRequest"/>
        </operation>
    </portType>
    <binding name="NotificationIRPNtf" type="ntfIRPNtfSystem:NotificationIRPNtf">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="notify">
            <soap:operation soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#notify"/>
            <input>
                <soap:body use="literal"/>
            </input>
        </operation>
    </binding>
    <service name="NotificationIRPNtf">
        <port name="NotificationIRPNtf" binding="ntfIRPNtfSystem:NotificationIRPNtf">
            <soap:address location="To be defined."/>
        </port>
    </service>
</definitions>
```

Annex D (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
05-2010	SA-48	SP-100271	--	--		Presentation to SA for information and approval	1.0.0
06-2010	SA-48	--	--	--		Publication	10.0.0
09-2010	SA-49	SP-100489	001	--		Add exception "ParameterNotSupported" in SOAP SS	10.1.0
09-2011	SA-53	SP-110526	002	--		Change direction of notifications in SOAP solution	10.2.0
09-2012	SA-57	-	-	-		Automatic upgrade from previous Release version 10.2.0	11.0.0
09-2014	SA-65	SP-140559	003	-		Update the link from Solution Set to Information Service due to the end of Release 12	12.0.0
2016-01	-	-	-	-		Update to Rel-13 version (MCC)	**13.0.0**
2016-06	SA#72	SP-160407	0004	-	F	Update the link from IRP Solution Set to IRP Information Service	13.1.0

History

Document history		
V13.0.0	February 2016	Publication
V13.1.0	August 2016	Publication