ETSI TS 132 153 V8.1.0 (2016-04)

**Digital cellular telecommunications system (Phase 2+) (GSM);
Universal Mobile Telecommunications System (UMTS);
LTE;
Telecommunication management;
Integration Reference Point (IRP)
technology specific templates, rules and guidelines
(3GPP TS 32.153 version 8.1.0 Release 8)**

Reference
RTS/TSGS-0532153v810

Keywords
GSM,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:

TS 32.150 "Integration Reference Point (IRP) Concept and definitions"

TS 32.151 "Integration Reference Point (IRP) Information Service (IS) template"

TS 32.152 "Integration Reference Point (IRP) Information Service (IS) Unified Modelling Language (UML) repertoire"

TS 32.153 "Integration Reference Point (IRP) technology specific templates, rules and guidelines"

TS 32.154 "Backward and Forward Compatibility (BFC); Concept and definitions"

TS 32.155 "Requirements template".

1 Scope

The present document contains the templates to be used for the production of Integration Reference Point (IRP) technology-specific specifications.

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 32.150: "Telecommunication management; Integration Reference Point (IRP) Concept and definitions".

[4] Void.

[5] Void.

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

[7] Void.

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

[9] OMG IDL Style Guide, ab/98-06-03, June 17, 1998.

[10] W3C WSDL 1.1 specification (http://www.w3.org/TR/wsdl) .

[11] Void.

[12] Void.

[13] Void.

[14] Void.

[15] W3C XML Specifications (http://www.w3.org/XML/Core/).

[16] IETF RFC 3629 (http://tools.ietf.org/html/rfc3629#ref-UNICODE).

[17] 3GPP TS 32.642: "Telecommunication management; Universal Terrestrial Radio Access Network (UTRAN) Network Resource Model (NRM) Integration Reference Point (IRP); Information Service (IS)".

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 32.150 [3] and the following apply:

IRPAgent: See 3GPP TS 32.150 [3].

IRPManager: See 3GPP TS 32.150 [3].

SOAP: See [8].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TS 32.101 [1], 3GPP TS 32.102 [2], 3GPP TS 32.150 [3] and the following apply:

CM	Conditional Mandatory
CM	Configuration Management
CO	Conditional Optional
CORBA	Common Object Request Broker Architecture (OMG)
FF	File Format
IDL	Interface Definition Language
IOC	Information Object Class
IOC	Information Object Class
IRP	Integration Reference Point
IS	Information Service
M	Mandatory
MO	Managed Object
MOC	Managed Object Class
NRM	Network Resource Model
O	Optional
OMG	Object Management Group
TS	Technical Specification
SOAP	W3C acronym
SS	Solution Set
SuM	Subscription Management
UML	Unified Modelling Language (OMG)
URI	Uniform Resource Identifier
WSDL	Web Service Description Language
XML	eXtensible Markup Language
XSD	XML Schema Definition

4 Interface IRP SS template

This clause contains the Interface IRP SS template.

The clauses in this template (that shall be used in the SS specifications) are numbered starting with "X", which in general should correspond to clause 4 that is the beginning of the normative part of a TS. However, if there is a need in a specific SS to introduce additional clauses in the TS body, X may correspond to a number higher than 4.

The SS templates use qualifiers M, O, CM and CO. The semantics of these qualifiers are defined in TS 32.150 [3].

The introductory clauses (from clause 1 to clause 3) for the SS should be modelled similarly to that of clause 1 to 3 of this specification.

Instructions are written in *italics*.

Usage of fonts shall be according to the following table.

Table: Usage of fonts

Item	Font
Class names	Courier New
Attribute names	Courier New
Operation names	Courier New
Parameter names	Courier New
Assertion names	Courier New
Notification names	Courier New
Exception names	Courier New
State names	Arial
Enumerated values	Arial

X Architectural Features

"X" represents a clause number in the actual SS. It contains at least the following this subclause:

X.1 General

This subclause contains the following paragraph:

*"*The overall architectural feature of <subject IRP> is specified in 3GPP TS <relevant specification number> [<relevant reference>]. This clause specifies features that are specific to the <specific technology> SS."

Example:

"The overall architectural feature of Alarm IRP is specified in 3GPP TS 32.111-2 [6]. This clause specifies features that are specific to the CORBA SS."

Specific for SOAP SS: the supported XSD, WSDL and SOAP versions shall be indicated together with the style and encoding style, and the namespaces with corresponding prefixes used in the WSDL specifications. Therefore the SOAP SS also contains the information quoted below:

'The SOAP <supported version> specification [relevant reference], XSD <supported version> specification [relevant reference] and WSDL <supported version> specification [relevant reference] are supported.

This specification uses "<RPC/Document>" style in WSDL file.

This specification uses "<Literal/Encoded>" encoding style in WSDL file.

This specification uses a number of namespace prefixes which are listed in Table X.1.

Prefix	Namespace
http	http://schemas.xmlsoap.org/wsdl/http/
soap	http://schemas.xmlsoap.org/wsdl/soap/
xs or xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
<IRP specific prefix>	<Namespace"s URI>

Table X.1: Prefixes and Namespaces used in this specification

'

X.a<Feature [a-1]>

<Description of architectureal feature 'a'>

"a" represents a number, starting at 2 and increasing by 1 with each new feature.

NOTE: This subclause is optional.

Y Mapping

"Y" represents a clause number, immediately following "X".

Y.1 Operation and Notification mapping

It contains this leading paragraph:

*"*The <subject IRP>: IS (see 3GPP TS <relevant specification number> [<relevant reference number>]) defines semantics of operation and notification visible across the Itf-N. Table Y.1.1 indicates mapping of these operations and notifications to their equivalents defined in this SS."

Example:

"Alarm IRP: IS 3GPP TS 32.111-2 [6] defines semantics of operation and notification visible across the Itf-N. Table Y.1.1 indicates mapping of these operations and notifications to their equivalents defined in this SS."

The table includes all operations and notifications, including those inherited. The table lists the operations first and then the notifications. The list order is the same as the one used in the IS. Semantics for the Qualifier is defined in Ref [x].

Table Y.1.1: Mapping from IS Operation/Notification to SS equivalents

IS Operation/Notification in 3GPP TS <relevant TS number > [<relevant reference number>]	SS Method	Qualifier
<operation1>	<method1>	M, O, CM or CO
<operation2>	<method2>	M, O, CM or CO
…		
..		
notify<notification1>	<notify1>	M, O, CM or CO
…	…	

Y.2 Operation parameter mapping

It contains this leading paragraph:

*"*Reference 3GPP TS <relevant version> [<relevant reference number>] defines semantics of parameters carried in operations across the Itf-N. The following set of tables indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS."

Example:

"Reference 3GPP TS 32.111-2 [6] defines semantics of parameters carried in operations across the Itf-N. The following set of tables indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS."

Table Y.2.1: Mapping from IS <operation1> parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<parameter-1>	<type definition> <parameter1>	M, O, CM or CO
<parameter-2>	<type definition> <parameter2>	M, O, CM or CO
…	…	…
…	….	…
status	…	M

Table Y.2.2: Mapping from IS <operation2> parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
<parameter-1>	<type definition> <parameter1>	M, O, CM or CO
…	…	…
status	…	M

Table Y.2.x:

IS Operation parameter	SS Method parameter	Qualifier
…	…	...
…	…	...
status	…	M

Y.3 Notification parameter mapping

It contains this leading paragraph:

"Reference 3GPP TS <relevant version number> [<relevant referenced number] defines semantics of parameters carried in notifications. The following tables indicate the mapping of these parameters to their SS equivalents."

Specific for CORBA SS: CORBA SS uses OMG CORBA Structured Event to carry IS-defined notification parameters. The Structured Event has the OMG defined parameters. These parameters will be related to the SS equivalents. So, the SS contains this paragraph.

"

The following tables indicate the mapping of these parameters to their OMG CORBA Structured Event (defined in OMG Notification Service [<relevant reference number>]) equivalents. The composition of OMG Structured Event, as defined in the OMG Notification Service [<relevant reference number>], is:

```
Header
      Fixed Header
           domain_name
           type_name
           event_name
      Variable Header
Body
      filterable_body_fields
      remaining_body
```

The following tables list all OMG Structured Event attributes in the second column. The first column identifies the <subject> IRP: IS [<relevant reference number>] defined notification parameters.

"

Table Y.3.1: Mapping for `notifyKkk`

IS Parameters	<SS> Parameters	Qualifier	Comment
<parameter-1>	<parameter-A>	M or O	…
<parameter-2>	<parameter-B>	M or O	…
…	…	M or O	…

Z Notification Interface

"Z" represents a number, immediately following "Y".

This sub-clause captures SS technology specific details required to realize the IS-defined <interface> for notifications.

Specific for CORBA SS:

The SS contains this:

"

OMG CORBA Notification push operation is used to realise the notification of `AlarmIRPNotifications`. All the notifications in this interface are implemented using this `push_structured_event` method.

<Z>.1 Method `push` (M)

```
module CosNotifyComm {
…
Interface SequencePushConsumer : NotifyPublish {
             void push_structured_events(

                    in CosNotification::EventBatch notifications)

          raises( CosEventComm::Disconnected);

             …

      }; // SequencePushConsumer

      …

}; // CosNotifyComm
```

NOTE 1: The `push_structured_events` method takes an input parameter of type `EventBatch` as defined in the OMG `CosNotification` module (OMG Notification Service [1]). This data type is the same as a sequence of Structured Events. Upon invocation, this parameter will contain a sequence of Structured Events being delivered to IRPManager by IRPAgent to which it is connected.

NOTE 2: The maximum number of events that will be transmitted within a single invocation of this operation is controlled by IRPAgent wide configuration parameter.

NOTE 3: The amount of time the supplier (IRPAgent) of a sequence of Structured Events will accumulate individual events into the sequence before invoking this operation is controlled by IRPAgent wide configuration parameter as well.

NOTE 4: IRPAgent may push `EventBatch` with only one Structured Event.

"

5 NRM IRP SS template

This clause contains the NRM IRP SS template.

The clauses in this template (that shall be used in the SS specifications) are numbered starting with "X", which in general should correspond to clause 4 that is the beginning of the normative part of a TS. However, if there is a need in a specific SS to introduce additional clauses in the TS body, X may correspond to a number higher than 4.

The introductory clauses (from clause 1 to clause 3) for the SS should be modelled similarly to that of this specification.

Usage of fonts shall be according to the following table.

Table: Usage of fonts

Item	Font
Class names	Courier New
Attribute names	Courier New
Assertion names	Courier New
Notification names	Courier New
State names	Arial
Enumerated values	Arial

X Architectural Features

"X" represents a clause number in the actual SS. It contains at least the following subclause:

X.1 General

This subclause contains the following paragraph:

*"*The overall architectural feature of <subject IRP> is specified in 3GPP TS <relevant specification number> [<relevant reference>]. This clause specifies features that are specific to the <specific technology> SS.".

Example:

"The overall architectural feature of UTRAN Network Resources IRP is specified in 3GPP TS 32.642 [17]. This clause specifies features that are specific to the CORBA SS."

X.a<Feature [a-1]>

<Description of architectureal feature 'a'>

"a" represents a number, starting at 2 and increasing by 1 with each new feature.

NOTE: This subclause is optional.

Y Mapping

"Y" represents a number, immediately following "X".

Y.1 General

This subclause contains a general introduction to the mapping.

For CORBA SSs, it contains the following paragraph:

"Attributes modelling associations as defined in the NRM (here also called "reference attributes") are in this SS mapped to attributes. The names of the reference attributes in the NRM are mapped to the corresponding attribute names in the MOC. When the cardinality for an association is 0..1 or 1..1 the datatype for the reference attribute is defined as an MOReference. The value of an MO reference contains the distinguished name of the associated MO. When the cardinality for an association allows more than one referred MO, the reference attribute will be of type MOReferenceSet, which contains a sequence of MO references."

For XML FF specifications, it contains the following paragraph:

'An IOC maps to an XML element of the same name as the IOC's name in the IS. An IOC attribute maps to a sub-element of the corresponding IOC's XML element, and the name of this sub-element is the same as the attribute's name in the IS.'

Y.2 Information Object Class (IOC) mapping

This subclause is only applicable to NRM IRP CORBA Solution Sets. This clause contains the mapping tables of all iS-defined IOCs to corresponding SS-level MOCs, excluding those inherited.

Y.2.a IOC <IOC name >

It contains one table with Mapping from NRM IS IOC attribute name to the SS equivalent MOC attribute name and attribute types.

"a" in the subclause heading represents a number, starting at 1 and increasing by 1 for each IOC.

Attribute of IOC <IOC name > in 3GPP TS <relevant TS number> [<relevant reference number>]	SS Attribute	SS Type	Support Qualifier	Read Qualifier	Write Qualifier
attribute-a	attribute-1	<type>	…	...	...
attribute-b	attribute-2	<type>	…	...	...
…	…	…	…	…	…

Y.2 Information Object Class (IOC) mapping

This subclause is only applicable to NRM IRP XML FF specifications. This subclause describes the mapping of all IS-defined IOCs to corresponding XML definitions, excluding the inherited IOCs.

Note: This subclause is also numbered Y.2, directly following Y.1, since every NRM SS-level TS is made for only one SS technology.

Annex A (normative): General rules for Solution Sets (SS)

A.1 Introduction

The intent of this annex is twofold.

The first intent is for 3GPP-internal use to document how a 3GPP Solution Set is (SS) produced and what it shall contain.

The second intent is to give the reader of an Information Service (IS) or a Solution Set (SS) a better understanding on how to interpret the IS or SS specifications.

A.2 Solution Set (SS) versioning

Editor's note: For Further Study.

A.3 Referenced Information Service (IS) specification

A sentence shall be included in the clause "Scope" of all SS specifications. The sentence shall read as follows:

"This Solution Set specification is related to Z".

where Z is the 3GPP Information Service (IS) specification number including the version, such as "TS 32.111-2 V4.1.X" for the case of Alarm Integration Reference Point (IRP): Information Service (IS).

NOTE: that "X", rather than the actual digit, is actually used in the sentence. This is because the value of X is not relevant for the reference purpose since different values of X identify different 3GPP published specifications that reflect only editorial changes.

Annex B (normative): Technology specific rules & guidelines for CORBA Solution Sets

B.1 Rules

B.1.1 Introduction

The intent of this annex is threefold.

1) The first intent is for 3GPP internal use to document how a 3GPP CORBA SS is produced and how it is structured.

2) The second intent with the annex is to give the reader or implementer of a CORBA SS a better understanding on how to interpret the CORBA SS specification.

3) The third and maybe most important intent is to put requirement on an implementer of a CORBA SS.

It is expected that this annex is to be extended in later versions of the present document.

B.1.2 Rules for specification of CORBA Solution Sets

B.1.2.1 Introduction

This subclause identifies rules for specification of CORBA SSs. This subclause is mainly for 3GPP-internal use. It is only for information for the implementer of a CORBA SS.

B.1.2.2 Pragma prefix

All IDL-code shall define the pragma prefix using the following statement:

#pragma prefix "3gppsa5.org"

See clause B.2.1.4.3 for information of this `#pragma` statement in relation to other IDL statements.

B.1.3 Implementation aspects of Interface IRP CORBA Solution Sets

B.1.3.1 Introduction

This subclause identifies rules for the implementation of CORBA SSs. This subclause is normative for the implementer of a CORBA SS.

B.1.3.2 IRPAgent behaviour on incoming optional method

The IRPAgent, claiming compliance to a particular SS version of a particular IRP such as the Alarm IRP, shall implement all Mandatory and all Optional methods. Each method implementation shall have a signature specifying all Mandatory and all Optional parameters.

- If the IRPAgent does not support a particular optional method, it shall throw the `OperationNotSupported` exception when the IRPManager invokes that method.
- If the IRPAgent have not implemented a particular method (because it is compiled with an IDL version that does not define the method), the CORBA ORB of the IRPAgent shall throw a system exception if the IRPManager invokes that method.

In all the above cases when an exception is thrown, the IRPAgent shall restore its state before the method invocation.

B.1.3.3 IRPAgent behaviour on incoming optional parameter of operation

An IRPAgent must implement all optional parameters, as well as mandatory parameters, in all methods.

If the IRPAgent supports the implemented method but does not support its (one or more) optional input parameters, upon method invocation, the IRPAgent shall check if those parameters carry "no information" or absence semantics (defined later in subclause B.1.3.5). If the check is negative, the IRPAgent shall throw the `ParameterNotSupported` exception with a string carrying the name of the unsupported optional parameter.

B.1.3.4 IRPAgent behaviour on outgoing attributes of notification

CORBA SS uses OMG defined structured event to carry notification. The structured event is partitioned into header and body.

The absence semantics of attribute in the header is realized by a string of zero length.

The body consists of one or more name-value pair attributes. The absence semantics of these attributes is realized by their absence.

For optional sub-attributes of an attribute carried by the name-value pair, their absence semantics is realized by the encoding rule of "absence semantics". See subclause B.1.3.5.

B.1.3.5 Encoding rule of absence semantics

The operation parameters are mapped to method parameters of CORBA SS. The absence semantics for an operation (input and output) parameter is method parameter type dependent.

- For a string type, if the parameter is specified as a string type, the absence semantics is a string of zero length. If the parameter is specified as a union structure (preferred), the absence semantics is conveyed via a FALSE Boolean value switch.
- For an integer type, if the parameter is specified as a signed, unsigned, long, etc type, the absence semantics is the highest possible positive number. If the parameter is specified as a union structure (preferred), the absence semantics is conveyed via a FALSE Boolean value switch.
- For a boxed valueType (supported by CORBA 2.3), it is the null value.

The notification parameters are mapped to attributes of the CORBA Structured Events. The absence semantics for a notification parameter is attribute position (within the Structured Event) dependent.

- For the fixed header of the Structured Event header, the absence semantics is realized by a string of zero length.
- For the filterable body fields of the Structured Event body, the absence semantics is realized by the absence of the corresponding attribute.

B.1.4 Rules for NRM IRP CORBA SS extensions

This clause discusses how the models and IDL definitions provided in a particular NRM IRP CORBA SS can be extended for a particular implementation and still remain compliant with IRP specifications.

Note: Protocol-neutral rules for NRM IRP IS-level extensions are defined in TS 32.150 [3] Annex G.

B.1.4.1 Allowed extensions

Vendor-specific MOCs may be supported. The vendor-specific MOCs may support new types of attributes.
The standardised notifications may be issued referring to the vendor-specific MOCs and vendor-specific attributes. New MOCs shall be distinguishable from standardised MOCs by name. Standardised and vendor-specific attributes may be used in vendor-specific MOCs. Vendor-specific attribute names shall be distinguishable from existing attribute names.

MOCs may be subclassed. Subclassed MOCs shall maintain the specified behaviour of the superior classes. They may add vendor-specific behaviour with vendor-specific attributes. When subclassing, naming attributes cannot be changed. The subclassed MOC shall support all attributes of its superior class. Vendor-specific attributes cannot be added to MOCs without subclassing.

When subclassing, the standardised containment rules and their specified cardinality shall still be followed.
As an example, ManagementNode (or its subclasses) shall be contained under SubNetwork (or its subclasses).

Managed Object Instances may be instantiated as CORBA objects. This requires that the MOCs be represented in IDL. MOCs are not currently specified in IDL, but may be specified in IDL for instantiation or subclassing purposes. However, management information models should not require that IRPManagers access the instantiated managed objects other than through supported methods in the present document.

Extension rules related to notifications (Notification categories, Event Types, Extended Event Types etc.) are for further study.

B.1.4.2 Extensions not allowed

The IDL specifications in the present document cannot be edited or altered. Any additional IDL specifications shall be specified in separate IDL files.

IDL interfaces (note: not MOCs) specified in the present document may not be subclassed or extended. New interfaces may be defined with vendor-specific methods.

B.2 Guidelines (Style Guide for CORBA SS IDL)

This subclause describes the style guide for writing IDL statements for Interface IRP and NRM IRP. The guidelines are largely based on the OMG IDL Style Guide (OMG document: ab/98-06-03) [9] with extensions for IRP use.

The guide sets out consistent naming, structural conventions and usage of SS interface for the IDL in IRP CORBA SS specifications.

B.2.1 Modules and File

B.2.1.1 Use of Modules

All declarations of IDL shall be contained in modules. No declarations of interfaces and definitions shall appear in the global scope.

Nesting modules is a useful technique when dealing with large namespaces to avoid name clashes and clarify relationships. A module nested within another module shall not have the same name as a top-level module in any other IRP CORBA SS specification.

B.2.1.2 File Names

CORBA SS specifications contain IDL statements.

The rule defined below specifies:

a) How to partition/extract these IDL statements to be placed in a file; and

b) How to name the file.

Note that IDL uses `"#include "X""` statement where `X` is a name of a file containing IDL statements.

Rule:

In the annex where IDL statements are defined, use a special marker to indicate that a set of IDL statements shall be contained in one file. The name of the file shall be the name of the first IDL module, concatenated with four characters "`.idl`". Within a CORBA SS, multiple markers (implying multiple files), can be used.

It is not allowed to have an IDL module split into multiple files.

B.2.1.3 Include Conventions

All included IDL files shall be specified using the `<…>` form of `#include`. For example:

```
#include <ManagedGenericIRPConstDefs.idl>
```

B.2.1.4 File Structure

B.2.1.4.1 File Internal Identification

The first line of the IDL file shall contain "`//File:`" followed by a single space followed by the name of the file. For example,

```
//File: ExampleIRPConstDefs.idl
```

B.2.1.4.2 File Guard

An IDL file shall use a *guard* (consisting of three pre-processor lines) to avoid multiple definition errors. An example of a guard for the file called `TestManagementIRPConstDefs.idl` is:

```
#ifndef _TestManagementIRPConstDefs_idl_
#define _TestManagementIRPConstDefs_idl_

...remainder of the IDL

#endif // _TestManagementIRPConstDefs_idl_
```

B.2.1.4.3 Required Contents

If any other files are to be included, the `#include` statements come after the guard.

After `#include` lines, if any, and immediately before the `module` statement, the following line shall appear:

```
#pragma prefix "3gppsa5.org"
```

B.2.1.4.4 Example illustrating a File Structure

```
//File: ExampleIRPConstDefs.idl
#ifndef _EXAMPLE_IRP_CONST_DEFS_IDL_
#define _EXAMPLE_IRP_CONST_DEFS_IDL_

// This module describes/is part of…
#include "ExampleIncludeOne.idl"
#include "ExampleIncludeTwo.idl"

#pragma prefix "3gppsa5.org"
module ExampleIRPConstDefs {

// IDL Definitions here

};
#endif // _EXAMPLE_IRP_CONST_DEFS_IDL_
```

B.2.2 Identifiers

B.2.2.1 Mixed Case, Beginning Upper, No Underscores

The following categories of identifiers follow the *Mixed Case, Beginning Upper, No Underscores* rules:

- `module`
- `interface`
- `typedef`
- Constructed types (`struct, union, enum`)
- `exception`

The "No underscores" rule is also applicable to all words that begin with an upper case letter with the remaining letters being lower case.

As a further note on naming, it is not necessary to append the value "*Type*" to an identifier. The fact that it is a type is obvious from the consistent application of this naming convention.

Examples:

```
module PMIRPConstDefs(…);
interface AttributeNameValue(…);
```

B.2.2.2 Lower Case with Underscores

The following categories of identifiers follow the *Lower Case with Underscores* rules. All letters are lower case and words (if more than one) are separated with underscores.

- Operation name and notification name
- Attribute name
- Parameter name
- Structure member name

Examples:

```
get_notification_categories(…);
string comment_text;
void get_alarm_count (…, out unsigned long critical_count,..);
  struct Comment {…; string user_id; string system_id;..};
```

B.2.2.3 Upper Case with Underscores

The following categories of identifiers follow *Upper Case with Underscores* rules. All letters are in upper case and words have an underscore separating them.

- Enum value
- Constant

Examples:

```
enum SubscriptionState {ACTIVE, SUSPENDED, INVALID};
const string JOB_ID = "JOB_ID";
```

B.2.2.4 Naming IDL Sequence Types

Typically a new type declared as an IDL sequence of another type will have the text "`List`" appended to the name of the base type. Another convention is to declare such types as unordered sequences or ordered sets for consistency with ASN.1 notation. In this case they should have the "`Seq`" or "`Set`" (instead of "`List`") appended respectively.

Example of an "ordered set":

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

B.2.3 Interface IRP

Every Interface IRP should have 3 IDL modules (each specified in a separate IDL file):

```
module YyyIRPConstDefs {…};  // no change from Rel-5 practice.

module YyyIRPSystem {…}; // no change from Rel-5 practice.

module YyyIRPNotifications {…}; // new compared to Rel-5 practice
```

The first module defines all necessary IDL constructs, such as constant strings and type definitions, for the methods and notifications. The second module defines the methods. The third module defines the notifications.

B.2.3.1 Constant String and Type Definitions

This first module defines all necessary IDL constructs used by the methods (defined in the second module) and notifications (defined in the third module). The name of this module is `YyyIRPConstDefs` where `Xxx` is the name of the subject Interface IRP. An example is "`PMIRPConstDefs`".

Within this module, define data types used in the methods.

Also, define the data types of the attribute values used in the notifications.

CORBA SS authors should always check the generic types defined in "`ManagedGenericIRPConstDefs`" before creating a new type.

For the attribute names of the structured notifications, define an interface `AttributeNameValue` that captures the string definitions. Make sure these definitions do not clash with those defined for the notification header, i.e. notification id, event time, system DN, managed object class and managed object instance (see `NotificationIRPNotification::Notify`).

An example from `PMIRPConstDefs`:

```
/**
 * This block identifies attributes which are included as part of the
 * PMIRP. These attribute values should not
 * clash with those defined for the attributes of notification
 * header (see IDL of Notification IRP).
 */

interface AttributeNameValue
{
    const string JOB_ID = "JOB_ID";
    const string JOB_STATUS = "JOB_STATUS";
    const string REASON = "REASON";
    const string MONITOR_ID = "MONITOR_ID";
    const string MONITOR_STATUS = "MONITOR_STATUS";
};
```

B.2.3.2 Operations

The second module defines the methods. The name of the module is `YyyIRPSystem` where `Yyy` is the name of the subject Interface IRP. An example is `AlarmIRPSystem`.

At the beginning of this module, define all required exceptions. Naming conventions for `exception` are covered in B.2.2.1 above. CORBA SS authors should always check if the generic exceptions defined in the `ManagedGenericIRPSystem` can be reused before declaring new `exception` types.

Then define one interface called `YyyIRP` encapsulating all methods of the subject `Yyy` Interface IRP. If the subject Interface IRP IS specifies that its `YyyIRP` inherits from `XxxIRP`, then reflect the inheritance relation in the interface definition. The following is an example of `AlarmIRP` that inherits from `ManagedGenericIRP.`

```
module AlarmIRPSystem
{
…
…
interface AlarmIRP : ManagedGenericIRPSystem:: ManagedGenericIRP {…};
…
};
```

Naming conventions for operations are covered in B.2.2.2 above.

B.2.3.3 Notifications

Use a separate module to define the notifications. The name the module is `YyyIRPNotifications` where `Yyy` is the name of the subject Interface IRP. Examples are `KernelCMIRPNotifications` and `PMIRPNotifications`.

For `NotificationIRPNotifications`, do:

- Define one IDL interface `Notify`. Capture the four constant strings that are the names of the four NV (name value) pairs of `filterable_body_field` of the CORBA structured event. These four CORBA NV pairs are mapped from the five notification header attributes (defined by the Notification IRP IS), i.e. the `objectClass, objectInstance, notificationId`, `eventTime` and `systemDN`.

For `YyyIRPNotifications` where `Yyy` is not `Notification`, do:

- At the beginning of this module, define the const strings for the notification types that correspond to the set of notifications specified by (and not inherited by and not imported by) the subject Interface IRP.
- Then define a number of IDL interfaces corresponding to notifications specified in the subject Interface IRP. These interfaces should inherit from `NotificationIRPNotifications`::`Notify`. Within each interface, the first IDL statement defines the notification type (that is used as the second field of the fixed header of the structured notification). The second and subsequent IDL statements define the attribute names of this notification type, excepting those already defined by `NotificationIRPNotifications`::`Notify`. The data type of the attribute value, which is defined in `YyyIRPConstDefs`, should be mentioned in the comment block of this IDL statement.
- Then define a number of IDL interfaces corresponding to notifications imported, if any. These interfaces should inherit from the imported interface. An example is `interface NotifyObjectCreation : KernelCMIRPNotifications:: NotifyObjectCreation`. Within this interface, define all necessary IDL constructs, if any, which are not defined in the imported interface. This interface may contain no IDL statement if the IDL constructs defined in the imported interface are sufficient. For each interface imported, insert a comment "The first field of this notification carries the IRPVersion of this CORBA SS."
- There is no need to re-define interfaces for notifications that are already specified in other Interface IRP, and from which the subject IRP inherits.

The following is an extract from `PMIRPNotifications`.

```
module PMIRPNotifications
{

   const string ET_MEASUREMENT_JOB_STATUS_CHANGED = "notifyMeasurementJobStatusChanged";
   const string ET_THRESHOLD_MONITOR_STATUS_CHANGED = "notifyThresholdMonitorStatusChanged";

   interface NotifyMeasurementJobStatusChanged: NotificationIRPNotifications::Notify
   {
     const string EVENT_TYPE = ET_MEASUREMENT_JOB_STATUS_CHANGED;

     /**
     * This constant defines the name of the jobId property,
     * which is transported in the filterable_body fields.
     * The data type for the value of this property
     * is PMIRPConstDefs::JobIdType.
     */
     const string JOB_ID = PMIRPConstDefs::AttributeNameValue::JOB_ID;

     …
     …
   };


   interface NotifyXXX : NotificationIRPNotifications::Notify
   {
       …
   };

   …
};
```

B.2.4 NRM IRP

Use one module to define the IDL constructs for the managed object classes. The name of this module is `XxxNRIRPConstDefs` where `Xxx` is the name of the subject NRM IRP.

An example is `UtranNRIRPConstDefs`.

Within the module, define a set of IDL interfaces each of which corresponds to a managed object class specified. The interface definition respects the inheritance relation specified.

After the interface definition, the type definition for each attribute defined for the managed object class is defined (including inherited attributes). The type is defined in one of two ways:

1. With a typedef of the type that matches the attribute definition in the NRM Information Object Class (IOC) mapping table, or
2. With a CORBA IDL comment if the same attribute name is already defined in managed object classes already defined in this NRM.

An example of managed object class `RncFunction,` which inherits from `GenericNRIRPConstDefs::ManagedFunction`, is shown below.

```
module UtranNRIRPConstDefs
{
…

/**
       *  Definitions for MO class RncFunction
       */
      interface RncFunction : GenericNRIRPConstDefs::ManagedFunction
      {
         const string CLASS = "RncFunction";

         // Attribute Names
         //
         const string rncFunctionId = "rncFunctionId";

         const string mcc= "mcc";
         const string mnc= "mnc";
         const string rncId= "rncId";
      };
      typedef string rncFunctionId;
      typedef long mcc;
      typedef long mnc;
      typedef long rncId;
      // userLabel
…
};
```

Annex C (normative): Technology specific rules & guidelines for SOAP SSs

C.1 Rules

C.1.1 Introduction

The intent of this annex is threefold.

4) The first intent is for 3GPP internal use to document how a 3GPP SOAP SS is produced and how it is structured.

5) The second intent with the annex is to give the reader or implementer of a SOAP SS a better understanding on how to interpret the SOAP SS specification.

6) The third and maybe most important intent is to put requirement on an implementer of a SOAP SS.

It is expected that this annex is to be extended in later versions of the present document.

C.1.2 Rules for specification of SOAP Solution Sets

C.1.2.1 Introduction

This subclause identifies rules for specification of SOAP SSs. This subclause is mainly for 3GPP-internal use. It is only for information for the implementer of a SOAP SS.

C.1.2.2 File names

The WSDL files should follow the naming convention:

`IRPNumber-3digitversion.wsdl`

with:

- IRPNumber being the number after the 3GPP document version number, with any '.' removed,
- 3digitversion being the 3 digits after the 'V' in the 3GPP document version number, with any '.' removed.

Examples:

If the 3GPP document version number is "3GPP TS 32.667 V7.0.2 (2007-06)", then the WSDL filename shall be `32667-702.wsdl`.

If the 3GPP document version number is "3GPP TS 32.111-7 V0.1.0 (2008-10)", then the WSDL filename shall be `32111-7-101.wsdl`.

C.1.2.3 Files location

The WSDL files should be stored in a zip format under the 3GPP server with the following naming rule:

http://www.3gpp.org/ftp/Specs/archive/32_series/*IRPNumber*/schema/*IRPNumber-3digitversion*-wsdl.zip

With *IRPNumber* and *3digitversion* being defined in paragraph C.1.2.2 File names.

Example:

http://www.3gpp.org/ftp/Specs/archive/32_series/32.667/schema/32667-700-wsdl.zip

C.1.2.4 XML version encoding

The first line of each wsdl and xsd file shall indicate the XML version and encoding. The supported XML version is 1.0 (refer to [15]) and encoding is UTF-8 (refer to [16]).

Therefore, the wsdl and xsd files should begin with:

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

C.1.3 Implementation aspects of Interface IRP SOAP Solution Sets

C.1.3.1 Introduction

This subclause identifies rules for the implementation of SOAP SSs. This subclause is normative for the implementer of a SOAP SS.

C.1.3.n XXXX rule

C.2 Guidelines (Style Guide for SOAP SS WSDL)

This subclause describes the style guide for writing WSDL documents for Interface IRP.

The guide sets out consistent naming, structural conventions and usage of SS interface for the WSDL in IRP SOAP SS specifications.

C.2.1 File structure

The structure of the wsdl file shall follow the W3C WSDL Document Structure defined in [10] specifications.

C.2.1.1 Definitions

Guidelines for the Definitions section.

C.2.1.2 Namespaces

Guidelines for the Namespaces section.

C.2.1.3 Documentation

Guidelines for the optional Documentation section.

C.2.1.4 Types

Guidelines for Types

C.2.1.5 Message

Guidelines for Messages

C.2.1.6 Port type

Guidelines for Port types

C.2.1.7 Binding

Guidelines for Bindings

C.2.1.8 Port

Guidelines for Ports

C.2.1.9 Service

Guidelines for Services

C.2.2 Identifiers

C.2.2.n *Rule n*

Identify the categories of identifiers which follow the Rule n rules.

C.3 XML Schema Guidelines

This subclause describes the style guide for writing XML Schema for Operation/Notification messages and parameters to be imported into the WSDL file for the IRP SOAP SS specifications.

C.3.1 XSD guidelines for messages definition

C.3.1.1 Request message

Guidelines for the Request message.

C.3.1.2 Response messages

Guidelines for the Response messages.

C.3.1.3 Fault messages

Guidelines for the Fault messages.

C.3.2 XSD guidelines for parameters definition

C.3.1.1 General guidelines

General guidelines for the parameters definitions.

C.3.1.1 Simple type definitions

Guidelines for the simple type definitions.

C.3.1.2 Complex type definition

Guidelines for the complex type.

C.3.3 Identifiers

The identifiers should follow the rules defined in paragraph C.2.2.

Annex D (normative): Technology specific rules & guidelines for XML specifications

D.1 Rules

D.1.1 XSD Use Cases

D.1.1.1 Background

3GPP defines a number of IOCs, say in Release-N. 3GPP also defines an XSD schema that can capture these IOCs, in Release-N.

Release-N+1 authors can/may also extend these Release-N IOCs to capture the newly agreed capabilities and publish them as Release-N+1 IOCs. Authors also define a Release-N+1 XSD schema.

The Use Cases here illustrate the capabilities of the Release-N+1 XSD schema.

The "?" in the Use Case tables are for further study.

D.1.1.2 Use Cases Set 1

Suppose the old-XSD has SubNetwork (SN) containing ManagedElement (ME).

Suppose the new-XSD has enhanced-SN containing enhanced-ME.

Suppose we want the XML-instance-documents (doc-1, doc-2, etc) to contain instances as identified below:

- Doc-1 has SN instance containing ME instances
- Doc-2 has SN instance containing enhanced-ME instances
- Doc-3 has SN instance containing ME instances and enhanced-ME instances
- Doc-4 has enhanced-SN containing enhanced-ME instances
- Doc-5 has enhanced-SN containing ME instances
- Doc-6 has enhanced-SN containing enhanced-ME instances and ME instances

Can a XML-instance-doc-creator (IRPAgent or IRPManager) produce the doc (column) using the XSD (row) identified?

	Doc-1	Doc-2	Doc-3	Doc-4	Doc-5	Doc-6
Old-XSD	?	?	?	?	?	?
New-XSD	?	?	?	?	?	?

Can a XML-instance-doc-reader (IRPAgent or IRPManager) validate the doc (i.e. confirm that the document is well-formed) (column) using the XSD (row) identified?

	Doc-1	Doc-2	Doc-3	Doc-4	Doc-5	Doc-6
Old-XSD	?	?	?	?	?	?
New-XSD	?	?	?	?	?	?

D.1.1.3 Use Cases Set 2

Suppose one old-XSD has SubNetwork (SN) containing ManagedElement (ME) that in turn, contained an RNCFunction (RNC) defined by the other Old-XSD.

Suppose the new-XSD has enhanced-SN containing enhanced-ME. Suppose also another new-XSD has enhanced-RNC.

- Doc-7 has ME instance containing RNC instances
- Doc-8 has ME instances containing enhanced-RNC instances
- Doc-9 has ME instances containing RNC instances and enhanced-RNC instances
- Doc-10 has enhanced-ME instances containing enhanced-RNC instances
- Doc-11 has enhanced-ME instances containing RNC instances
- Doc-12 has enhanced-ME instances containing RNC instances and enhanced-RNC instances

Can a XML-instance-doc creator (IRPAgent or IRPManager) produce the doc (column) using the XSD (row) identified?

	Doc-7	Doc-8	Doc-9	Doc-10	Doc-11	Doc-12
Old-XSD	?	?	?	?	?	?
New-XSD	?	?	?	?	?	?

Can a reader (IRPAgent or IRPManager) validate the doc (i.e. confirm that the document is well-formed) (column) using the XSD (row) identified?

	Doc-1	Doc-2	Doc-3	Doc-4	Doc-5	Doc-6
Old-XSD	?	?	?	?	?	?
New-XSD	?	?	?	?	?	?

D.1.2 Rules for NRM IRP XML specification extensions

TBD

D.1.3 XML version encoding

The first line of each XML schema description file file shall indicate the XML version and encoding. The supported XML version is 1.0 (refer to [15]) and encoding is UTF-8 (refer to [16]).

Therefore, the xsd files should begin with:

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

D.2 Guidelines

TBD

Annex E (informative): Change history

Change history								
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Cat**	**Old**	**New**
Mar 2008	SA_39	SP-080070	--	--	Submitted to SA#39 for Information	--	--	1.0.0
Mar 2009	SA-43	SP-090057	--	--	Submitted to SA for approval	--	2.0.0	8.0.0
Mar 2016	SA-71	SP-160032	0005	-	Correcting references	F	8.0.0	8.1.0

History

Document history		
V8.0.0	April 2009	Publication
V8.1.0	April 2016	Publication