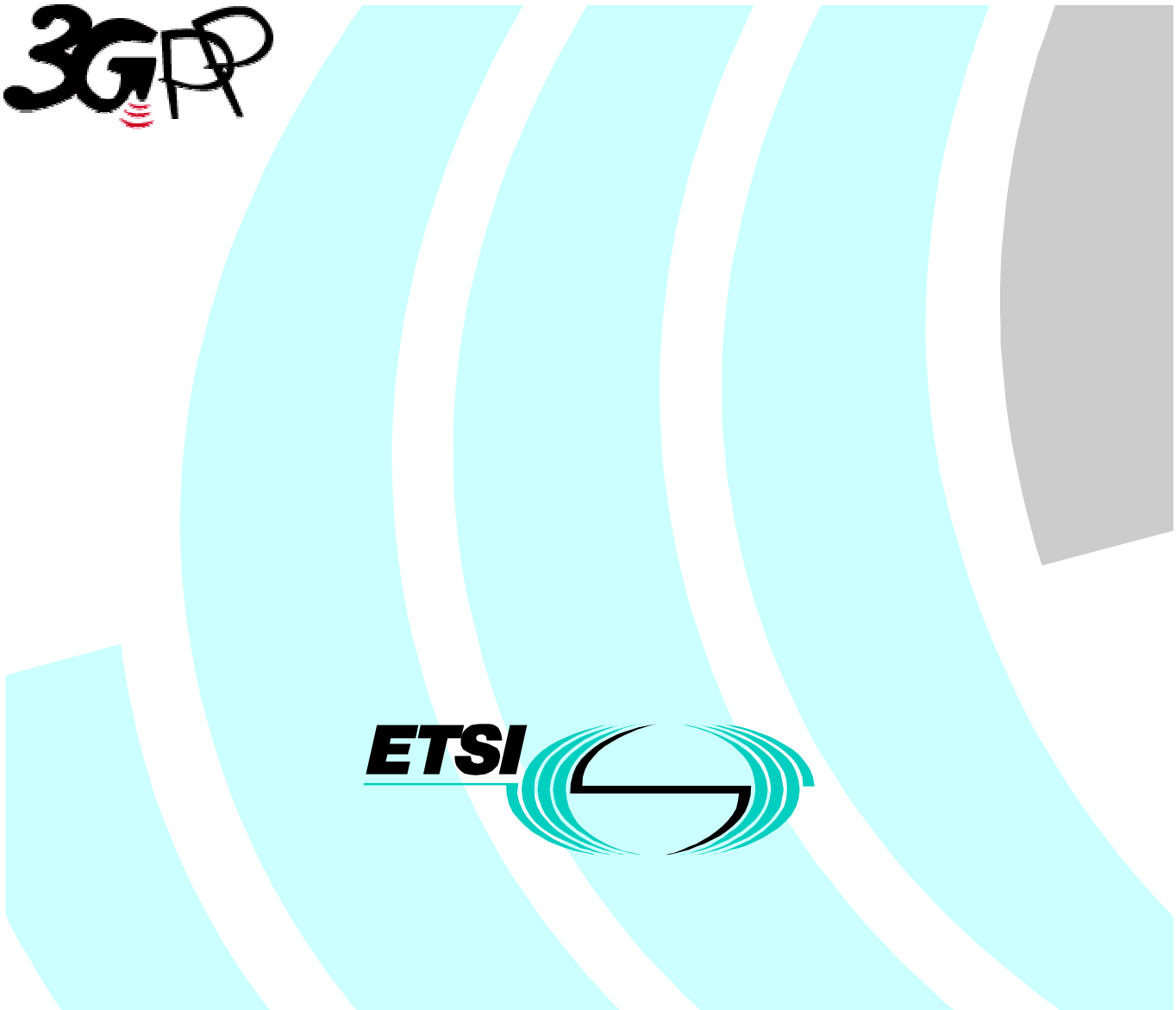
ETSI TS 132 106-6 V3.0.0 (2000-12)

Technical Specification

Universal Mobile Telecommunications System (UMTS);
Telecommunication Management;
Configuration Management;
Part 6: Basic Configuration
Management IRP: CORBA Solution Set Version 1:1
(3GPP TS 32.106-6 version 3.0.0 Release 1999)



Reference
RTS/TSGS-0532106-6U

Keywords
UMTS

ETSI

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

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

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

Intellectual Property Rights

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

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

Foreword

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

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

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

Contents

Foreword

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

The present document is part 6 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Services and System Aspects; Telecommunication Management; Configuration Management, as identified below:

Part 1: "3G Configuration Management: Concept and Requirements";
Part 2: "Notification Integration Reference Point: Information Service Version 1";
Part 3: "Notification Integration Reference Point: CORBA Solution Set Version 1:1";
Part 4: "Notification Integration Reference Point: CMIP Solution Set Version 1:1";
Part 5: "Basic Configuration Management IRP: Information Model Version 1";
Part 6: "Basic Configuration Management IRP CORBA Solution Set Version 1:1";
Part 7: "Basic Configuration Management IRP CMIP Solution Set Version 1:1";
Part 8: "Name Convention for Managed Objects".

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

Configuration Management (CM), in general, provides the operator with the ability to assure correct and effective operation of the 3G-network as it evolves. CM actions have the objective to control and monitor the actual configuration on the NEs and NRs, and they may be initiated by the operator or functions in the OSs or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service. The CM actions are initiated either as a single action on a Network Element (NE) of the 3G-network or as part of a complex procedure involving actions on many NEs.

The Itf-N interface for Configuration Management is built up by a number of Integration Reference Points (IRPs) and a related Name Convention, which realise the functional capabilities over this interface. The basic structure of the IRPs is defined in 3GPP TS 32.101 [1] and 3GPP TS 32.102 [2]. For CM, a number of IRPs (and the Name Convention) are defined herein, used by this as well as other technical specifications for telecom management produced by 3GPP. All these documents are included in Parts 2 and onwards of the 3GPP TS 32.106.

This document constitutes 32.106 Part 6 - Basic Configuration Management IRP: CORBA Solution Set Version 1:1.

1 Scope

The purpose of this *Basic Configuration Management (CM) IRP: Information Service CORBA Solution Set* is to define the mapping of the IRP information model (see 3GPP TS 32.106-5 [4]) to the protocol specific details necessary for implementation of this IRP in a CORBA/IDL environment.

The present document does not describe any Network Resource Model (NRM) – this is described in 3GPP TS 32.106-5 [4]. Please note that 3GPP TS 32.106-5 [4] defines an *IRP Information Model*, which comprises both an IS and NRM definition.

2 References

The following documents contain provisions, which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] 3GPP TS 32.101: "3G Telecom Management principles and high level requirements".

[2] 3GPP TS 32.102: "3G Telecom Management architecture".

[3] 3GPP TS 32.106-1: “3G Configuration Management”.

[4] 3GPP TS 32.106-5: “Basic Configuration Management IRP: Information Model”.

[5] 3GPP TS 32.106-8: “Name Convention for Managed Objects”.

[6] OMG Notification Service, Version 1.0.

[7] OMG CORBA services: Common Object Services Specification, Update: November 22, 1996.

[8] The Common Object Request Broker: Architecture and Specification (for specification of valid version, see [1]).

[9] 3GPP TS 32.106-3: “Notification IRP: CORBA Solution Set, Version 1:1”.

[10] 3GPP TS 32.111-3: “Alarm IRP: CORBA Solution Set, Version 1:1”.

3 Definitions and abbreviations

3.1 Definitions

For terms and definitions please refer to 3GPP TS 32.101 [1], 3GPP TS 32.102 [2], 3GPP TS 32.106-1 [3] and 3GPP TS 32.106-5 [4].

3.2 Abbreviations

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

CORBA	Common Object Request Broker Architecture
DN	Distinguished Name

IS	Information Service
IDL	Interface Definition Language (OMG)
IRP	Integration Reference Point
MO	Managed Object
MOC	Managed Object Class
NRM	Network Resource Model
OMG	Object Management Group
SS	Solution Set

4 IRP Solution Set version

The version of this CORBA Solution Set (SS) is 1:1, where the first "1" means that it corresponds to the Information Model version 1, and the second "1" means that it is the first CORBA Solution Set corresponding to Information Model version 1.

5 Architectural features

The overall architectural feature of Basic Configuration Management IRP is specified in 3GPP TS 32.106-5 [4]. This clause specifies features that are specific to the CORBA SS.

5.1 Notifications

Notifications are sent according to the Notification IRP: CORBA SS (see 3GPP TS 32.106-3 [9]).

The contents of the Basic CM IRP notifications are defined in the present document.

5.2 Filter language

The filter language used in the SS is the Extended Trader Constraint Language (see OMG Notification Service [6]). IRPAgents may throw a FilterComplexityLimit exception when a given filter is too complex. However, for 3GPP Release 99 an "empty filter" shall be used i.e. a filter that satisfies all MOs of a scoped search (this does not affect the filter for notifications as defined in the Notification IRP – see 3GPP TS 32.106-3 [9]).

5.3 Syntax for Distinguished Names and Versions

The format of a Distinguished Name is defined in 3GPP TS 32.106-8 [5].

The Version of this IRP is represented as a string. The value of this version is defined by a constant in Annex A.

6 Mapping

6.1 General mappings

The IS parameter name managedObjectInstance is mapped into DN.

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.

If a reference attribute is changed, an AttributeValueChange notification is emitted.

6.2 Operation and Notification mapping

The IS part of Basic CM IRP: IM (see 3GPP TS 32.106-5 [4]) defines semantics of operation and notification visible across the Basic Configuration Management IRP. Table 1 indicates mapping of these operations and notifications to their equivalents defined in this SS.

Table 1: Mapping from IS Notification/Operation to SS equivalents

IS Operation/ notification (3GPP TS 32.106-5 [4])	SS Method	Qualifier

`getMoAttributes`	`BasicCmIrpOperations::find_managed_objects`
	`Iterator::get_next_elements`
	`Iterator::destroy`
		M

`getContainment`	`BasicCmIrpOperations::find_managed_objects`
	`Iterator::get_next_elements`
	`Iterator::destroy`
		O

`getBasicCmIRPVersion`	`get_basicCm_IRP_version`	M
`notifyObjectCreation` (to convey of a new Managed Object created)	`See Notification IRP: CORBA SS [9]`	O
`notifyObjectDeletion` (to convey of a new Managed Object deleted)	`See Notification IRP: CORBA SS [9]`	O
`notifyAttributeValueChange` (to convey of a change of one or several attributes of a Managed Object)	`See Notification IRP: CORBA SS [9]`	O

6.3 Operation parameter mapping

The IS part of Basic CM IRP: IM (see 3GPP TS 32.106-5 [4]) defines semantics of parameters carried in operations across the Basic Configuration Management IRP. Tables 2, 3 and 4 indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

The SS operation `find_managed_objects` is equivalent to the IS operation `getMoAttributes` when called with `ResultContents` set to `NAMES_AND_ATTRIBUTES`. Iterating the `Iterator` is used to fetch the result.

Table 2: Mapping from IS `getMoAttributes` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`baseObjectInstance`	`in DN baseObject`	M
`scope`	`in searchControl (SearchControl.scope and SearchControl.level)`	M
`filter`	`in searchControl (SearchControl.filter)`	M
`attributeListIn`	`in requestedAttributes`	M

`managedObjectClass`
`managedObjectInstance`
`attributeListOut`
	`parameter fetchedElements in the get_next_elements in the Iterator interface.`	M

`status`	`exception UndefinedMOException,`
	`exception IllegalDNFormatException,`
	`exception UndefinedScopeException,`
	`exception IllegalScopeTypeException,`
	`exception IllegalScopeLevelException,`
	`exception IllegalFilterFormatException,`
	`exception FilterComplexityLimit`
		M

The SS operation `find_managed_objects` is equivalent to the IS operation `getContainment` when called with `ResultContents` set to `NAMES`. Iterating the `Iterator` is used to fetch the result.

Table 3: Mapping from IS `getContainment` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`baseObjectInstance`	`in DN baseObject`	M
`scope`	`in searchControl (SearchControl.scope and SearchControl.level)`	O
Not specified in IS	`in searchControl (SearchControl.filter)`	M
`containment`	`parameter fetchedElements in the get_next_elements in the Iterator interface.`	M
`status`	`exception UndefinedMOException, exception IllegalDNFormatException, exception UndefinedScopeException, exception IllegalScopeTypeException, exception IllegalScopeLevelException, exception IllegalFilterFormatException, exception FilterComplexityLimit`	M

Table 4: Mapping from IS `getBasicCmIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`versionNumberList`	Return value of type: `CommonIRPConstDefs::VersionNumberSet`	M
`status`	`- (No failure conditions identified)`	M

6.4 Notification attribute mapping

The IS part of Basic CM IRP: IM (see 3GPP TS 32.106-5 [4]) identifies and defines the semantics of attributes for `notifyObjectCreation`, `notifyObjectDeletion` and `notifyAttributeValueChange` for use for its IRP. Table 5 shows the mapping of the IS notifications to SS equivalents.

Table 5: Mapping from IS notifications to SS equivalents

IS notifications in 3GPP TS 32.106-5 [4]	SS notifications	Qualifier
`NotifyObjectCreation`	`push_structured_event`	O
`NotifyObjectDeletion`	`push_structured_event`	O
`NotifyAttributeValue Change`	`push_structured_event`	O

The IS part of Basic CM IRP: IM (see 3GPP TS 32.106-5 [4]) also qualifies the attributes. Tables 6, 7, 8 and 9 show the mapping of these IS attributes to SS equivalents.

Table 6: Mapping from IS `Notification Header` attributes to SS equivalent

IS Attribute of `Notification Header` in 3GPP TS 32.106-5 [4]	SS Attribute	Qualifier
managedObjectClass	NotificationDefs::NotificationCommon::MANAGED_OBJECTCLASS	M
managedObjectInstance	NotificationDefs::NotificationCommon::MANAGED_OBJECT_INSTANCE	M
notificationId	NotificationDefs::NotificationCommon::NOTIFICATION_ID	O
eventTime	NotificationDefs::NotificationCommon::EVENT_TIME	M
systemDN	NotificationDefs::NotificationCommon::SYSTEM_DN	O
eventType	header.fixed_header.event_type.type_name	M
extendedEventType	header.fixed_header.event_name - (always contains an empty string)	M

Table 7: Mapping from IS `notifyObjectCreation` attributes to SS equivalent OBJECT_CREATION

IS Attribute of `notifyObjectCreation` in 3GPP TS 32.106-5 [4]	SS Attribute	Qualifier
notificationHeader	See Table 6	M
correlatedNotifications	NotificationDefs::MOCreation::CORRELATED_NOTIFICATIONS	O
additionalText	NotificationDefs::MOCreation::ADDITIONAL_TEXT	O
sourceIndicator	NotificationDefs::MOCreation::SOURCE_INDICATOR	O
attributeList	remainder_of_body	O

Table 8: Mapping from IS `notifyObjectDeletion` attributes to SS equivalent OBJECT_DELETION

IS Attribute of `notifyObjectDeletion` in 3GPP TS 32.106-5 [4]	SS Attribute	Qualifier
notificationHeader	See Table 6	M
correlatedNotifications	NotificationDefs::MODeletion::CORRELATED_NOTIFICATIONS	O
additionalText	NotificationDefs::MODeletion::ADDITIONAL_ TEXT	O
sourceIndicator	NotificationDefs::MODeletion::SOURCE_INDICATOR	O
attributeList	remainder_of_body (a field of the StructuredEvent)	O

Table 9: Mapping from IS `notifyAttributeValueChange` attributes to SS equivalent ATTRIBUTE_VALUE_CHANGE

IS Attribute of `notifyAttributeValueChange` in 3GPP TS 32.106-5 [4]	SS Attribute	Qualifier
notificationHeader	See Table 6	M
correlatedNotifications	NotificationDefs::AttributeValueChange::CORRELATED_NOTIFICATIONS	O
additionalText	NotificationDefs::AttributeValueChange::ADDITIONAL_TEXT	M
sourceIndicator	NotificationDefs::AttributeValueChange::SOURCE_INDICATOR	O

IS Attribute of notifyAttributeValueChange in 3GPP TS 32.106-5 [4]	SS Attribute	Qualifier
attributeValueChangeDefinition	remainder_of_body	M

6.5 Network Resource Model (NRM) mapping

6.5.1 Generic NRM Managed Object Class (MOC) mapping

This Solution Set supports reference attributes for relations other than containment relations between objects. Reference attributes are therefore introduced in each MOC where needed.

6.5.1.1 MOC G3SubNetwork

Table 10: Mapping from NRM MOC G3SubNetwork attributes to SS equivalent MOC G3SubNetwork attributes

NRM Attributes of MOC G3SubNetwork in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
g3SubNetworkId	g3SubNetworkId	string	Read-Only, M
dnPrefix	dnPrefix	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.1.2 MOC G3ManagedElement

Table 11: Mapping from NRM MOC G3ManagedElement attributes and association roles to SS equivalent MOC G3ManagedElement attributes

NRM Attributes/Association roles	SS Attributes	SS Type	Qualifier
g3ManagedElementId	g3ManagedElementId	string	Read-Only, M
dnPrefix	dnPrefix	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M
locationName	locationName	string	Read-Only, M
vendorName	vendorName	string	Read-Only, M
userDefinedState	userDefinedState	string	Read-Only, M
managedElementType	managedElementType	StringSet	Read-Only, M
managedBy	managedBy	BasicCmIRPSystem::AttributeTypes::MOReferenceSet	Read-Only, M

6.5.1.3 MOC MeContext

Table 12: Mapping from NRM MOC MeContext attributes to SS equivalent MOC MeContext attributes

NRM Attributes of MOC MeContext in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
meContextId	meContextId	string	Read-Only, M
dnPrefix	dnPrefix	string	Read-Only, M

6.5.1.4 MOC ManagementNode

Table 13: Mapping from NRM MOC ManagementNode attributes and association roles to SS equivalent MOC ManagementNode attributes

NRM Attributes/association roles of MOC ManagementNode in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
managementNodeId	managementNodeId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M
locationName	locationName	string	Read-Only, M
vendorName	vendorName	string	Read-Only, M
userDefinedState	userDefinedState	string	Read-Only, M
manages	manages	BasicCmIRPSystem::AttributeTypes::MOReferenceSet	Read-Only, M

6.5.1.5 MOC ManagedFunction

This Managed Object Class is provided for sub-classing only. Therefore no mapping for this class is provided in this document.

6.5.1.6 MOC IRPAgent

Table 14: Mapping from NRM MOC IRPAgent attributes to SS equivalent MOC IRPAgent attributes

NRM Attributes of MOC IRPAgent in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
irpAgentId	irpAgentId	string	Read-Only, M
systemDN	systemDN	string	Read-Only, M

6.5.1.7 MOC NotificationIRP

Table 15: Mapping from NRM MOC NotificationIRP attributes to SS equivalent MOC NotificationIRP attributes

NRM Attributes of MOC NotificationIRP in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
notificationIRPId	notificationIRPid	string	Read-Only, M
irpVersion	irpVersion	StringSet	Read-Only, M

6.5.1.8 MOC AlarmIRP

Table 16: Mapping from NRM MOC AlarmIRP attributes to SS equivalent MOC AlarmIRP attributes

NRM Attributes of MOC AlarmIRP in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
alarmIRPId	alarmIRPid	string	Read-Only, M
irpVersion	irpVersion	StringSet	Read-Only, M

6.5.1.9 MOC BasicCmIRP

Table 17: Mapping from NRM MOC BasicCmIRP attributes to SS equivalent MOC BasicCmIRP attributes

NRM Attributes of MOC BasicCmIRP in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
basicCmIRPId	basicCmIRPid	string	Read-Only, M
irpVersion	irpVersion	StringSet	Read-Only, M

6.5.2 UMTS NRM Managed Object Class (MOC) mapping

6.5.2.1 MOC RncFunction

Table 18: Mapping from NRM MOC RncFunction attributes to SS equivalent MOC RncFunction attributes

NRM Attributes of MOC RncFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
rncFunctionId	rncFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.2 MOC UtranCell

Table 19: Mapping from NRM MOC UtranCell attributes and associations to SS equivalent MOC UtranCell attributes

NRM Associations/Attributes of MOC UtranCell in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
utranCellId	utranCellId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M
AssociatedWith-1/ utranCell-IubLink	utranCellIubLink	BasicCmIRPSystem::AttributeTypes::MOReference	Read-Only, O (See Note 1)
AssociatedWith-2/ utranCell-NodeBFunction	utranCellNodeBFunction	BasicCmIRPSystem::AttributeTypes::MOReference	Read-Only, O (See Note 1)
NOTE: This association is optional, but under the condition that at least one of the associations AssociatedWith-1 and AssociatedWith-2 (between UtranCell, IubLink and NodeBFunction) shall be present.			

6.5.2.3 MOC NodeBFunction

Table 20: Mapping from NRM MOC NodeBFunction attributes and associations to SS equivalent MOC NodeBFunction attributes

NRM Associations/Attributes of MOC NodeBFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
nodeBFunctionId	nodeBFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M
ConnectedTo/ nodeBFunction-IubLink	nodeBFunctionIubLink	BasicCmIRPSystem::AttributeTypes::MOReference	Read-Only, M
AssociatedWith-2/ nodeBFunction-UtranCell	nodeBFunctionUtranCell	BasicCmIRPSystem::AttributeTypes::MOReferenceSet	Read-Only, O (See Note 1)
NOTE: This association is optional, but under the condition that at least one of the associations AssociatedWith-1 and AssociatedWith-2 (between UtranCell, IubLink and NodeBFunction) shall be present.			

6.5.2.4 MOC IubLink

Table 21: Mapping from NRM MOC IubLink attributes and associations to SS equivalent MOC IubLink attributes

NRM Associations/Attributes of MOC IubLink in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
iubLinkId	iubLinkId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M
AssociatedWith-1/ iubLink-UtranCell	iubLinkUtranCell	BasicCmIRPSystem::AttributeTypes::MOReferenceSet	Read-Only, O (See Note 1)
ConnectedTo/ iubLink-NodeBFunction	iubLinkNodeBFunction	BasicCmIRPSystem::AttributeTypes::MOReference	Read-Only, M
NOTE: This association is optional, but under the condition that at least one of the associations AssociatedWith-1 and AssociatedWith-2 (between UtranCell, IubLink and NodeBFunction) shall be present.			

6.5.2.5 MOC MscFunction

Table 22: Mapping from NRM MOC MscFunction attributes to SS equivalent MOC MscFunction attributes

NRM Attributes of MOC MscFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
mscFunctionId	mscFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.6 MOC HlrFunction

Table 23: Mapping from NRM MOC HlrFunction attributes to SS equivalent MOC HlrFunction attributes

NRM Attributes of MOC HlrFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
hlrFunctionId	hlrFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.7 MOC VlrFunction

Table 24: Mapping from NRM MOC VlrFunction attributes to SS equivalent MOC VlrFunction attributes

NRM Attributes of MOC VlrFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
vlrFunctionId	vlrFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.8 MOC AucFunction

Table 25: Mapping from NRM MOC AucFunction attributes to SS equivalent MOC AucFunction attributes

NRM Attributes of MOC AucFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
aucFunctionId	aucFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.9 MOC EirFunction

Table 26: Mapping from NRM MOC EirFunction attributes to SS equivalent MOC EirFunction attributes

NRM Attributes of MOC EirFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
eirFunctionId	eirFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.10 MOC SmsIwmscFunction

Table 27: Mapping from NRM MOC SmsIwmscFunction attributes to SS equivalent MOC SmsIwmscFunction attributes

NRM Attributes of MOC SmsIwmscFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
smsIwmscFunctionId	smsIwmscFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.11 MOC SmsGmscFunction

Table 28: Mapping from NRM MOC SmsGmscFunction attributes to SS equivalent MOC SmsGmscFunction attributes

NRM Attributes of MOC SmsGmscFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
smsGmscFunctionId	smsGmscFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.12 MOC SgsnFunction

Table 29: Mapping from NRM MOC SgsnFunction attributes to SS equivalent MOC SgsnFunction attributes

NRM Attributes of MOC SgsnFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
sgsnFunctionId	sgsnFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.13 MOC GgsnFunction

Table 30: Mapping from NRM MOC GgsnFunction attributes to SS equivalent MOC GgsnFunction attributes

NRM Attributes of MOC GgsnFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
ggsnFunctionId	ggsnFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.14 MOC BgFunction

Table 31: Mapping from NRM MOC BgFunction attributes to SS equivalent MOC BgFunction attributes

NRM Attributes of MOC BgFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
bgFunctionId	bgFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

6.5.2.15 MOC GmscFunction

Table 32: Mapping from NRM MOC GmscFunction attributes to SS equivalent MOC GmscFunction attributes

NRM Attributes of MOC GmscFunction in 3GPP TS 32.106-5 [4]	SS Attributes	SS Type	Qualifier
gmscFunctionId	gmscFunctionId	string	Read-Only, M
userLabel	userLabel	string	Read-Only, M

7 Use of OMG Structured Event

In CORBA SS, OMG defined `StructuredEvent` (see OMG Notification Service [6]) is used to carry notification. This clause identifies the OMG defined `StructuredEvent` attributes that carry the attributes of parameters defined in 3GPP TS 32.106-5 [4].

The composition of OMG Structured Event, as defined in OMG Notification Service [6], is:

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

Table 33 lists all OMG Structured Event attributes in its leftmost column. The second column identifies the SS attributes, if any, that shall be carried there.

Attributes that are denoted as "optional" may be absent from the OMG Structured Event. As an example, if the optional `additionalText` attribute is not used for a particular notification, then the IRPAgent may exclude `additionalText` from the filterable body fields for that particular notification. Individual notifications from the same IRPAgent may include or exclude the same optional attribute.

Table 33: Use of OMG Structured Event

OMG CORBA Structured Event attribute	Comment
`domain_name`	It contains the version of the supported SS version, This version is defined by constant VERSION, see `Annex A (normative): CORBA IDL, Access Protocol`

`type_name`	It shall indicate one of the following ITU-T defined semantics:
	`OBJECT_CREATION`
	`OBJECT_DELETION`
	`ATTRIBUTE_VALUE_CHANGE`
	
	It is a string. It is assumed that the types are defined in `Annex B (normative): CORBA IDL, Notification Definitions`

`event_name`	Not used. Shall be set to an empty string.
`variable Header`	Not used
`filterable_ body_fields`	Are used to transport most of the notification information. Each property transported contains a name and a value, where all names are defined in: Annex B (normative): CORBA IDL, Notification Definitions and in subclause 6.4 Notification attribute mapping
`remainder_of_body`	Is used to transport attribute information, see: Annex B (normative): CORBA IDL, Notification Definitions and subclause 6.4 Notification attribute mapping

8 Rules for management information model extensions

This clause discusses how the models and IDL definitions provided in 3GPP TS 32.111-3 [10], 3GPP TS 32.106-3 [9] and 3GPP TS 32.106-6 (the present document) can be extended for a particular implementation and still remain compliant with 3GPP SA5's specifications.

8.1 Allowed extensions

Vendor-specific MOCs, other than those specified in clause 6.5, may be supported. The vendor-specific MOCs may support new types of attributes. The 3GPP SA5-specified notifications may be issued referring to the vendor-specific MOCs and vendor-specific attributes. New MOCs shall be distinguishable from 3GPP SA5 MOCs by name. 3GPP SA5-specified and vendor-specific attributes may be used in vendor-specific MOCs. Vendor-specific attribute names shall be distinguishable from existing attribute names.

Basic CM IRP MOCs may be subclassed. Subclassed MOCs shall maintain the specified behaviour of the 3GPP SA5's 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 3GPP SA5 MOCs without subclassing.

When subclassing, the 3GPP SA5-specified containment rules and their specified cardinality shall still be followed. As an example, `ManagementNode` (or its subclasses) shall be contained under `G3SubNetwork` (or its subclasses). Also, in R99, there may only be 0 or 1 `ManagementNode` (or its subclasses) contained under `G3SubNetwork` (or its subclasses).

Managed Object Instances may be instantiated as CORBA objects. This requires that the MOCs be represented in IDL. 3GPP SA5's 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 (3GPP TS 32.106-6).

Extension rules related to notifications (Notification categories, Event Types, Extended Event Types etc.) are for further study in 3GPP's Releases 4/5.

8.2 Extensions not allowed

The IDL specifications in 3GPP TS 32.111-3 [10], 3GPP TS 32.106-3 [9] and 3GPP TS 32.106-6 (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 3GPP TS 32.111-3 [10], 3GPP TS 32.106-3 [9] and 3GPP TS 32.106-6 (the present document) may not be subclassed or extended. New interfaces may be defined with vendor-specific methods.

Annex A (normative): CORBA IDL, Access Protocol

```
#ifndef BasicCmIRPSystem_idl
#define BasicCmIRPSystem_idl

#pragma prefix "3gppsa5.org"

#include "CommonIRPConstDefs.idl"


module BasicCmIRPSystem
{

   /**
    *  This constant defines the version of this IRP.
    */
   const string VERSION = "1c1";

   /**
    * The format of Distinguished Name (DN) is specified in "Name Conventions
    * for Managed Objects revision B".
    */
   typedef string DN;


   /**
    *  This module adds datatype definitions for types
    *  used in the NRM which are not basic datatypes defined
    *  already in CORBA.
    */
   module AttributeTypes
   {

      /**
       * An MO reference referres to an MO instance.
       * "otherMO" contains the  distinguished name of the referred MO.
       * A conceptual "null" reference (meaning no MO is referenced)
       * is represented as an empty string ("").
       *
       */
      struct MOReference
      {
         DN otherMO;
      };

      /**
       * MOReferenceSet represents a set of MO references.
       * This type is used to hold 0..n MO references.
       * A referred MO is not allowed to be repeated (therefore
       * it is denoted as a "Set")
       */
      typedef sequence<MOReference> MOReferenceSet;


      /**
       *  A set of strings.
       */
```

```
      typedef sequence<string> StringSet;

   };



    exception IllegalFilterFormatException {
       string reason;
    };
    exception IllegalDNFormatException {
       string reason;
    };
    exception IllegalScopeTypeException {
       string reason;
    };
    exception IllegalScopeLevelException {
       string reason;
    };
    exception UndefinedMOException {
       string reason;
    };

   exception UndefinedScopeException {
       string reason;
   };

   exception FilterComplexityLimit {
      string reason;
   };


   /**
    *
    * In R99 the only allowed filter value is "TRUE" i.e. a filter that
    * matches everything.
    */
   typedef string FilterType;

    /**
     * ResultContents is used to tell how much information to get back
     * from the find_managed_objects operation.
     *
     * NAMES: Used to get only Distinguished Name
     *        for MOs.
     *        The name contains both the MO class
     *        and the names of all superior objects in the naming
     *        tree.
     *
     * NAMES_AND_ATTRIBUTES: Used to get both NAMES plus
     *    MO attributes (all or selected).
     */
    enum ResultContents
    {
       NAMES,
       NAMES_AND_ATTRIBUTES
    };

    /**
     * ScopeType defines the kind of scope to use in a search
     * together with SearchControl.level, in a SearchControl value.
     *
```

```
   * SearchControl.level is always >= 0. If a level is bigger than the
   * depth of the tree there will be no exceptions thrown.
   * BASE_ONLY: level ignored, just return the base object.
   * BASE_NTH_LEVEL: return all subordinate objects that are on "level"
   * distance from the base object, where 0 is the base object.
   * BASE_SUBTREE: return the base object and all of its subordinates
   * down to and including the nth level.
   * BASE_ALL: level ignored, return the base object and all of it's
   * subordinates.
   */
  enum ScopeType
  {
     BASE_ONLY,
     BASE_NTH_LEVEL,
     BASE_SUBTREE,
     BASE_ALL
  };

  /**
   * SearchControl controls the find_managed_object search,
   * and contains:
   * the type of scope ("type" field),
   * the level of scope ("level" field), level 0 means the "baseObject",
   *    level 1 means baseobject including its sub-ordinates etc..
   * the filter ("filter" field),
   * the result type ("contents" field).
   * The type, level and contents fields are all mandatory.
   * The filter field contains the filter expression.
   *  The string "TRUE" indicates "no filter",
   * i.e. a filter that matches everything.
   */
  struct SearchControl
  {
     ScopeType type;
     unsigned long level;
     FilterType filter;
     ResultContents contents;
  };

  /**
   * Represents an attribute: "name" is the attribute name
   * and "value" is the attribute value in form of a CORBA Any.
   * The allowed attribute value types are defined in the
   * AttributeTypes module.
   */
 struct MOAttribute
 {
    string name;
    any value;
 };

 typedef sequence<MOAttribute> MOAttributeSet;



  struct Result
  {
     DN mo;
     MOAttributeSet attributes;
  };
```

```
typedef sequence<Result> ResultSet;


/**
 * Iterator interface
 *
 *
 */
interface Iterator
{
   exception IllegalCountException {
      string reason;
   };


   /**
    * Gets data from an Iterator.
    * This method returns between 1 and "how_many" elements
    * The IRPAgent may return less than "how_many" items even if
    * there are more items to send. "how_many" shall be non-zero.
    * Return TRUE if there are more elements to return.
    * Return FALSE if thereare no more elements to be returned.
    * Note that the IRPAgent may both provide the last items in the
    * Basic CM operation results and also indicate FALSE for completion.
    *
    * If FALSE is returned, the IRPAgent will automatically destroy the
    * iterator. Otherwise, it is the IRPManager's responsibility to
    * destroy the iterator.
    *
    *
    * @parm howMany how many elements to return in the "fetchedElements" out
    * parameter.
    * @parm fetchedElements the elements.
    * @returns A boolean indicating if any elements are returned.
    * "fetchedElements" is empty when the Iterator is empty.
    *
    * @raises IllegalCountException "howMany" has a value < 0.
    */
   boolean get_next_elements(in unsigned short howMany,
                             out ResultSet fetchedElements)
      raises (IllegalCountException);


   /**
    * Destroys an Iterator. This method shall be used if
    * the iterator is to be removed before all elements
    * are iterated.
    *
    */
   void destroy();


};


typedef sequence<string> AttributeNameSet;


/**
 * The BasicCmIrpOperations interface.
```

```
 * Supports a number of Resource Model versions.
 */
interface BasicCmIrpOperations
{

   /**
    * Get the version of the interface and all supported resource
    * model versions.
    *
    * @returns all supported versions.
    */
   CommonIRPConstDefs::VersionNumberSet get_basicCm_IRP_version();


   /**
    * Performs a containment search, using a SearchControl to
    * control the search and the returned results.
    *
    * All MOs in the scope constitute a set that the filter works on.
    * The result Iterator contains all matched MOs,
    * with the amount of detail specified in the SearchControl.
    * For the special case when no managed objects are matched in
    * find_managed_objects, the Iterator will be returned. Executing
    * the get_next_elements in the Iterator will return FALSE for
    * completion.
    *
    * @parm baseObject The start MO in the containment tree.
    * @parm searchControl the SearchControl to use.
    * @parm requestedAttributes defines which attributes to get.
    *   If this parameter is empty (""),  all attributes shall
    *   be returned. Note: In R99 this is the only supported semantics.
    *   Note that this argument is only
    *   relevant if ResultContents in the search control is
    *   specifed to NAMES_AND_ATTRIBUTES.
    *
    *
    * @raises UndefinedMOException The MO does not exist.
    * @raises IllegalDNFormatException The dn syntax string is
    * malformed.
    * @raises IllegalScopeTypeException The ScopeType in scope contains
    * an illegal value.
    * @raises IllegalScopeLevelException The scope level is negative
    * (<0).
    * @raises IllegalFilterFormatException The filter string is
    * malformed.
    * @raises FilterComplexityLimit if the filter syntax is correct,
    *   but the filter is too complex to be processed by the IRP agent.
    * @see SearchControl
    * @see Iterator
    */
   Iterator find_managed_objects(in DN baseObject,
                                 in SearchControl searchControl,
                                 in AttributeNameSet requestedAttributes)
      raises (UndefinedMOException,
              IllegalDNFormatException,
              UndefinedScopeException,
              IllegalScopeTypeException,
              IllegalScopeLevelException,
              IllegalFilterFormatException,
              FilterComplexityLimit);
```

```
    };
};
#endif
```

Annex B (normative): CORBA IDL, Notification Definitions

```
#ifndef NotificationDefs_idl
#define NotificationDefs_idl

#pragma prefix "3gppsa5.org"

#include <TimeBase.idl>          // CORBA Time Service
#include <NotificationIRPConstDefs.idl>

module BasicCmIRPSystem
{

   module NotificationDefs
   {


      /**
       * Definition of ITU-T defined semantics.
       * These constants are used in the type_name
       * (header.fixed_header.event_type.type_name)
       * field to denote the notification type
       * Note all values are unique among themselves.  Other IRP documents
       * cannot use the same values.
       */
      const string ET_OBJECT_CREATION = "x6";

      const string ET_OBJECT_DELETION = "x7";

      const string ET_ATTRIBUTE_VALUE_CHANGE = "x8";



      /**
       * Information about one attribute
       * - name defines the name of the attribute
       * - value defines the value of the attribute
       *
       */
      struct MOAttribute
      {
         string name;
         any value;
      };


      /**
       * A set of attribute names and values
       */
      typedef sequence<MOAttribute> MOAttributeSet;


      /**
       *  This interface defines fields that are common for all
       *  notification types.
       *  All constants in the scope of this interface will be
       *  visible in the interfaces that inherits this.
       *  For instance constant
```

```
 *  NotificationCommon::MANAGED_OBJECT_CLASS
 *  can be addressed by MODeletion::MANAGED_OBJECT_CLASS
 */
interface NotificationCommon
{

   /**
    * This constant defines a field in the filterable
    * information in a StructuredEvent.
    * This string is mapped to the name part of a
    * Property in the event and the value part will
    * carry the MO class name represented
    * as a string.
    */
   const string MANAGED_OBJECT_CLASS =
     NotificationIRPConstDefs::NV_MANAGED_OBJECT_CLASS;


   /**
    * This constant defines a field in the filterable
    * information in a StructuredEvent.
    * This string is mapped to the name part of a
    * Property in the event and the value part will
    * carry the MO distinguished name represented
    * as a string.
    */
   const string MANAGED_OBJECT_INSTANCE =
     NotificationIRPConstDefs::NV_MANAGED_OBJECT_INSTANCE;


   /**
    *  This constant defines the name of the notification
    *  ID property, which is transported in the
    *  filterable_body_fields
    */
   const string NOTIFICATION_ID =
     NotificationIRPConstDefs::NV_NOTIFICATION_ID;


   /**
    *  This constant defines the name of the
    *  event time property, which is transported in the
    *  filterable_body_fields.
    *  The data type for the value of this property
    *  is defined by datatype CommonIRPConstDefs::IRPTime
    */
   const string EVENT_TIME =
     NotificationIRPConstDefs::NV_EVENT_TIME;



   /**
    *  This constant defines the name of the
    *  system name property, which is transported in the
    *  filterable_body_fields
    */
   const string SYSTEM_DN =
     NotificationIRPConstDefs::NV_SYSTEM_DN;


   /**
```

```
     *  This constant defines the name of the
     *  source indicator  property, which is transported in the
     *  filterable_body_fields
     */
    const string SOURCE_INDICATOR = "SOURCE";

    /**
     * Valid values for the SOURCE_INDICATOR
     * property
     */
    const string RESOURCE_OPERATION =  "RESOURCE OPERATION";
    const string MANAGEMENT_OPERATION = "MANAGEMENT OPERATION";
    const string UNKNOWN_OPERATION = "UNKNOWN";




    /**
     *  This constant defines the name of the
     *  additional text property,
     *  which is transported in the filterable_body
     *  fields.
     *  The data type for the value of this property
     *  is a string.
     */
    const string ADDITIONAL_TEXT =
      NotificationIRPConstDefs::NV_ADDITIONAL_TEXT;



    /**
     *  This constant defines the name of the
     *  correlated notifications property,
     *  which is transported in the
     *  filterable_body_fields
     *  The value part of the property is defined
     *  in the NotificationIRP;
     *    NotificationIRPConstDefs::CorrelatedNotificationSetType
     */
    const string CORRELATED_NOTIFICATIONS =
      NotificationIRPConstDefs::NV_CORRELATED_NOTIFICATIONS;

};


/**
 *  Constant definitions for the MO deleted notification
 */
interface MODeletion : NotificationCommon
{

    const string EVENT_TYPE = ET_OBJECT_DELETION;


    /**
     * This information mapped into the remainder_of_body
     * in the StructuredEvent
     */
    typedef MOAttributeSet AttributeValues;

};
```

```
      /**
       *  Constant definitions for the MO created notification
       */
      interface MOCreation : NotificationCommon
      {
         const string EVENT_TYPE = ET_OBJECT_CREATION;


         /**
          * This information mapped into the remainder_of_body
          * in the StructuredEvent
          */
         typedef MOAttributeSet InitialAttributeValues;

      };



      /**
       *  Constant definitions for the Attribute Value Change
       *  notification
       */
      interface AttributeValueChange : NotificationCommon
      {

         const string EVENT_TYPE = ET_ATTRIBUTE_VALUE_CHANGE;

         /**
          * Information about modidified attributes for
          * one MO instance.
          * - name defines the name of the attribute
          * - newValue defines the new value of the attribute
          * - oldValue defines the previous value of the attribute
          *    The value is optional, which means that it may contain
          *    an empty any (null inserted in the any).
          *
          */
         struct ModifiedAttribute
         {
            string name;
            any newValue;
            any oldValue;
         };


         /**
          * This information mapped into the remainder_of_body
          * in the StructuredEvent.
          */
         typedef sequence<ModifiedAttribute> ModifiedAttributeSet;



       };

    };
};
```

```
#endif
```

Annex C (normative): CORBA IDL, NRM Definitions

```
#ifndef ModelDefs_idl
#define ModelDefs_idl

#pragma prefix "3gppsa5.org"


module BasicCmIRPSystem
{

   /**
    * This module defines constants for each MO class name and
    * the attribute names for each defined MO class.
    */
   module NRMDefinitions
   {

      /**
       *  Definitions for MO class G3SubNetwork
       */
      interface G3SubNetwork
      {
         const string CLASS = "G3SubNetwork";

         // Attribute Names
         //
         const string g3SubNetworkId = "g3SubNetworkId";
         const string dnPrefix = "dnPrefix";
         const string userLabel = "userLabel";
      };

      /**
       *  Definitions for MO class G3ManagedElement
       */
      interface G3ManagedElement
      {
         const string CLASS = "G3ManagedElement";

         // Attribute Names
         //
         const string g3ManagedElementId = "g3ManagedElementId";
         const string dnPrefix = "dnPrefix";
         const string managedElementType = "managedElementType";
         const string userLabel = "userLabel";
         const string vendorName = "vendorName";
         const string userDefinedState ="userDefinedState";
         const string locationName ="locationName";

         const string managedBy = "managedBy";

      };


      /**
       *  Definitions for MO class MeContext
       */
      interface MeContext
      {
```

```
   const string CLASS = "MeContext";

   // Attribute Names
   //
   const string meContextId = "meContextId";
   const string dnPrefix = "dnPrefix";
};

/**
 *  Definitions for MO class ManagementNode
 */
interface ManagementNode
{
   const string CLASS = "ManagementNode";

   // Attribute Names
   //
   const string managementNodeId = "managementNodeId";
   const string userLabel = "userLabel";
   const string vendorName = "vendorName";
   const string userDefinedState = "userDefinedState";
   const string locationName = "locationName";
   const string manages = "manages";

};


/**
 *  Definitions for abstract MO class ManagedFunction
 *
 */
interface ManagedFunction
{
   const string CLASS = "ManagedFunction";

   // Attribute Names
   //
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class RncFunction
 */
interface IRPAgent
{
   const string CLASS = "IRPAgent";

   // Attribute Names
   //
   const string irpAgentId = "irpAgentId";
   const string systemDN = "systemDN";
};



/**
 *  Definitions for MO class NotificationIRP
 */
interface NotificationIRP
{
```

```
   const string CLASS = "NotificationIRP";

   // Attribute Names
   //
   const string notificationIRPId = "notificationIRPId";
   const string irpVersion = "irpVersion";
};


/**
 *  Definitions for MO class AlarmIRP
 */
interface AlarmIRP
{
   const string CLASS = "AlarmIRP";

   // Attribute Names
   //
   const string alarmIRPId = "alarmIRPId";
   const string irpVersion = "irpVersion";
};




/**
 *  Definitions for MO class BasicCmIRP
 */
interface BasicCmIRP
{
   const string CLASS = "BasicCmIRP";

   // Attribute Names
   //
   const string basicCmIRPId = "basicCmIRPId";
   const string irpVersion = "irpVersion";
};


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

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


/**
 *  Definitions for MO class UtranCell
 */
interface UtranCell
{
   const string CLASS = "UtranCell";

   // Attribute Names
```

```
   //
   const string utranCellId = "utranCellId";
   const string userLabel = "userLabel";
   const string utranCellNodeBFunction = "utranCellNodeBFunction";
   const string utranCellIubLink = "utranCellIubLink";
};



/**
 *  Definitions for MO class NodeBFunction
 */
interface NodeBFunction
{
   const string CLASS = "NodeBFunction";

   // Attribute Names
   //
   const string nodeBFunctionId = "nodeBFunctionId";
   const string userLabel = "userLabel";
   const string nodeBFunctionIubLink = "nodeBFunctionIubLink";
   const string nodeBFunctionUtranCell = "nodeBFunctionUtranCell";
};


/**
 *  Definitions for MO class IubLink
 */
interface IubLink
{
   const string CLASS = "IubLink";

   // Attribute Names
   //
   const string iubLinkId = "iubLinkId";
   const string userLabel = "userLabel";
   const string iubLinkNodeBFunction = "iubLinkNodeBFunction";
   const string iubLinkUtranCell = "iubLinkUtranCell";

};


/**
 *  Definitions for MO class MscFunction
 */
interface MscFunction
{
   const string CLASS = "MscFunction";

   // Attribute Names
   //
   const string mscFunctionId = "mscFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class HlrFunction
 */
interface HlrFunction
{
```

```
   const string CLASS = "HlrFunction";

   // Attribute Names
   //
   const string hlrFunctionId = "hlrFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class VlrFunction
 */
interface VlrFunction
{
   const string CLASS = "VlrFunction";

   // Attribute Names
   //
   const string vlrFunctionId = "vlrFunctionId";
   const string userLabel = "userLabel";
};



/**
 *  Definitions for MO class AucFunction
 */
interface AucFunction
{
   const string CLASS = "AucFunction";

   // Attribute Names
   //
   const string aucFunctionId = "aucFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class EirFunction
 */
interface EirFunction
{
   const string CLASS = "EirFunction";

   // Attribute Names
   //
   const string eirFunctionId = "eirFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class SmsIwmscFunction
 */
interface SmsIwmscFunction
{
   const string CLASS = "SmsIwmscFunction";

   // Attribute Names
   //
```

```
   const string smsIwmscFunctionId = "smsIwmscFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class SmsGmscFunction
 */
interface SmsGmscFunction
{
   const string CLASS = "SmsGmscFunction";

   // Attribute Names
   //
   const string smsGmscFunctionId = "smsGmscFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class SgsnFunction
 */
interface SgsnFunction
{
   const string CLASS = "SgsnFunction";

   // Attribute Names
   //
   const string sgsnFunctionId = "sgsnFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class GgsnFunction
 */
interface GgsnFunction
{
   const string CLASS = "GgsnFunction";

   // Attribute Names
   //
   const string ggsnFunctionId = "ggsnFunctionId";
   const string userLabel = "userLabel";
};


/**
 *  Definitions for MO class BgFunction
 */
interface BgFunction
{
   const string CLASS = "BgFunction";

   // Attribute Names
   //
   const string bgFunctionId = "bgFunctionId";
   const string userLabel = "userLabel";
};
```

```
      /**
       *  Definitions for MO class GmscFunction
       */
      interface GmscFunction
      {
         const string CLASS = "GmscFunction";

         // Attribute Names
         //
         const string gmscFunctionId = "gmscFunctionId";
         const string userLabel = "userLabel";
      };



   };

};

#endif
```

Annex D (informative): Change history

Change history					
TSG SA#	Version	CR	Tdoc SA	New Version	Subject/Comment
S_10	1.0.0	-	SP-000514	3.0.0	Approved at TSG SA #10 and placed under Change Control

History

Document history		
V3.0.0	December 2000	Publication