Technical Specification Group Services and System Aspects
Meeting #17, Biarritz, France, 9-12 September 2002

TSGS#17(02)0483

Source: SA5 (Telecom Management)

Title: Rel-5 CRs 32.600/1/2/4 (Basic Configuration Management Integration Reference Point (IRP)) - upgrade to Rel-5

Document for: Approval

Agenda Item: 7.5.3

Doc-1st-	Spec	CR	Rev	Phase	Subject	Cat	Version	Doc-	Workite	Relation
SP-020483	32.600	001	-	Rel-5	**Add Kernel CM, Revise Basic (adding Active CM) and Bulk CM**	B	4.0.0	S5-026678	OAM-NIM	**Grandparent CR**
SP-020483	32.601	001	-	Rel-5	**Add Active CM and Update Basic CM requirements**	B	4.0.0	S5-026679	OAM-NIM	**Parent CR**
SP-020483	32.602	002	-	Rel-5	**Add Active CM and new methodology, Remove CM Notifications (moved to Kernel CM - 32.66x)**	B	4.1.0	S5-026680	OAM-NIM	**Child CR**
SP-020483	32.603	006	-	Rel-5	**Add Active Basic CM feature - CORBA Solution Set**	B	4.3.1	S5-026653	OAM-NIM	**Grandchild CR**

3GPP TSG-SA5 Meeting #30 **S5-026678**
Tampere, FINLAND, 19 - 23 August 2002

CR-Form-v4

CHANGE REQUEST

⌘ **32.600** **CR** **001** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network **X** Core Network **X**

Title:	⌘	Add Kernel CM, Revise Basic (adding Active CM) and Bulk CM
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**B**
Release:	⌘	Rel-5

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 *(GSM Phase 2)*
R96 *(Release 1996)*
R97 *(Release 1997)*
R98 *(Release 1998)*
R99 *(Release 1999)*
REL-4 *(Release 4)*
REL-5 *(Release 5)*

Reason for change:	⌘	Create Release 5 of this IRP Requirements Specification from 32600-400.
Summary of change:	⌘	1) Update for R5 2) Add Kernel CM Overview 3) Revise Overview of Basic (adding Active CM) and Bulk CM
Consequences if not approved:	⌘	

Clauses affected:	⌘	Introduction, Clauses 1 and 7		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications **X** O&M Specifications	⌘	 32.601, 32.602, 32.603
Other comments:	⌘	S5-026678 Rel-5 CR 32600 Add new Rel-5 functionality (Kernel CM, Active CM) - Parent CR S5-026679 Rel-5 CR 32601 Basic CM IRP Requirements - Child CR S5-026680 Rel-5 CR 32602 Basic CM IRP IS - Grandchild CR S5-026653 Rel-5 CR 32603 Basic CM IRP IS - Grandchild CR		

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 Network Elements (NEs) and Network Resources (NRs), and they may be initiated by the operator or by functions in the Operations Systems (OSs) or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service (QOS). The CM actions are initiated either as single actions on single NEs of the 3G network, or as part of a complex procedure involving actions on many resources/objects in one or several NEs.

Clauses 4 to 6 give an introduction and description of the main concepts of CM, which are not mandatory for compliance with this specification. Clause 7 contains the specific definitions for the standardised interface Itf-N, which are necessary to follow for compliance.

Clause 4 provides a brief background of CM, while Clause 5 explains CM services available to the operator. Clause 6 breaks these services down into individual CM functions, which support the defined services. Clause 7 defines the Itf-N (see 3GPP TS 32.102 [2]) to be used for 3G CM.

~~Due to the growing number of specifications to model new services and Resource Models for Configuration Management (CM), as well as the expected growth in size of each of them from 3GPP Release 4 onwards, a new structure of the specifications is already needed in Release 4. This structure is needed for several reasons, but mainly to enable more independent development and release for each part, as well as a simpler document identification and version handling. Another benefit would be that it becomes easier for bodies outside 3GPP, such as the ITU-T, to refer to telecom management specifications from 3GPP. The new structure of the specifications does not lose any information or functionality supported by the Release 1999. The restructuring also includes defining new IRPs for the Network Resource Model (NRM) parts of R99 Basic CM IRP (Generic, Core Network and UTRAN NRM). These IRPs are named "Network Resources IRP".~~

~~Further, the Notification IRP (in Release 1999: 32.106-1 to -4) and the Name convention for Managed Objects (in Release 1999: 32.106-8) have been moved to a separate number series used for specifications common between several management areas (e.g. CM, FM, PM).~~

~~Finally, in addition to the restructuring mentioned above, the need to define some new functionality and IRPs for CM compared to Release 1999, has also been identified. Firstly, a new Bulk CM IRP, and secondly an a GERAN Network Resources IRP, have been created. Thirdly, the Generic, UTRAN and GERAN Network Resources IRPs have been extended with support for GSM-UMTS Inter-system handover (ISH), and the 32.600 (Concept and High-level Requirements) has been modified to cover the high-level Bulk CM and ISH requirements.~~

Table: Mapping between Release '99 and the new specification numbering scheme

R99 Old no.	Old (R99) specification title	Rel-4 New no.	New (Rel-4) specification title
32.106-1	3G Configuration Management: Concept and Requirements	32.600	**3G Configuration Management: Concept and High-level Requirements**
32.106-1	<Notification IRP requirements from 32.106-1 and 32.106-2>	32.301	**Notification IRP**: Requirements
32.106-2	Notification IRP: IS	32.302	Notification IRP: Information Service
32.106-3	Notification IRP: CORBA SS	32.303	Notification IRP: CORBA SS
32.106-4	Notification IRP: CMIP SS	32.304	Notification IRP: CMIP SS
32.106-8	Name convention for Managed Objects	32.300	**Name Convention for Managed Objects**
32.106-1	<Basic CM IRP IS requirements from 32.106-1 and 32.106-5>	32.601	**Basic CM IRP**: Requirements
32.106-5	Basic CM IRP IM (Intro & IS part)	32.602	Basic CM IRP: Information Service
32.106-6	Basic CM IRP CORBA SS (IS related part)	32.603	Basic CM IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (IS related part)	32.604	Basic CM IRP: CMIP SS
32.106-8	Name convention for Managed Objects	32.300	**Name Convention for Managed Objects**
-	-	32.611	**Bulk CM IRP**: Requirements
-	-	32.612	Bulk CM IRP: Information Service
-	-	32.613	Bulk CM IRP: CORBA SS
-	-	32.614	Bulk CM IRP: CMIP SS
		32.615	Bulk CM IRP: XML file format definition
32.106-1	<Basic CM IRP Generic NRM requirements from 32.106-1 and 32.106-5>	32.621	**Generic Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (Generic NRM part)	32.622	Generic Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (Generic NRM related part)	32.623	Generic Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (Generic NRM related part)	32.624	Generic Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP CN NRM requirements from 32.106-1 and 32.106-5>	32.631	**Core Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (CN NRM part)	32.632	Core Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (CN NRM related part)	32.633	Core Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (CN NRM related part)	32.634	Core Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP UTRAN NRM requirements from 32.106-1 and 32.106-5>	32.641	**UTRAN Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (UTRAN NRM part)	32.642	UTRAN Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (UTRAN NRM related part)	32.643	UTRAN Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (UTRAN NRM related part)	32.644	UTRAN Network Resources IRP: CMIP SS
		32.651	**GERAN Network Resources IRP**: Requirements
		32.652	GERAN Network Resources IRP: NRM
		32.653	GERAN Network Resources IRP: CORBA SS
		32.654	GERAN Network Resources IRP: CMIP SS

7 Itf-N Interface

7.1 CM principles

The Itf-N (see ref. 3GPP TS 32.102 [2]) is an object oriented interface, i.e. all resources of the 3G network (functional and physical resources) whose management is standardised by the present document are represented as Managed Object Instances (MOI) of a Network Resource Model (NRM).

The NRM shall be highly simplified for the purpose of the NM, based on the assumption that all of the detailed CM actions, including fault correction after one or more alarms, are performed by an Element Manager (EM), which knows the vendor-specific NRM and configuration.

The NRM identifies the basic Network Resources (NRs) to the level of detail required by FM and PM at the Network Management (NM) level. In addition to NR identification, the NRM also supports the alarm surveillance part of FM by defining which alarms can be notified by which Managed Object Classes (MOCs).

The definition of the Network Resource Model (NRM) for the Itf-N (connecting the NM with a "subordinate entity", which may be an EM or a NE) is described in 3GPP TS 32.622 [3] and other NRM IRPs listed in the Introduction clause, which define the Generic Network Resource Model and other specific NRMs applicable to UMTS management, such as the UTRAN NRM.

This clause describes the specific functional requirements related to CM of Network Resources (NRs) on the Itf-N There are two types of CM functions:

- *Passive* CM (configuration overview), which mainly provides to the NM current information about the current configuration changes by means of notifications, and allows a retrieval and synchronisation of configuration related data on NM request.

  The forwarding of these notifications over the Itf-N is controlled by means of configuring adequate filtering mechanisms within the subordinate entities. The Itf-N also provides the means for storage ("logging") and later retrieval of desired information within the subordinate entities.

- *Active* CM, which offers to the NM operator a real capability to change the current network configuration.

There are also at least two approaches to CM - Basic CM and Bulk CM:

Basic CM is characterised by

- The use of singular operations to retrieve (configuration parameters) over *Itf-N* from single NEs, or a collection of NEs. (The passive aspect of Basic CM.)

- The use of singular operations to activate configuration parameters in EM/NEs over Itf-N. (The active aspect of Basic CM.)

Bulk CM is characterised by

- Bulk (file-oriented) data retrieval (configuration parameters) over Itf-N from single NEs, a collection of NEs or the whole network. (The passive aspect of Bulk CM.)

- Bulk (file-oriented) data download of configuration parameters to EM/NEs over Itf-N. (An active aspect of Bulk CM.)

- The network-wide activation of those parameters through a single operation. (An active aspect of Bulk CM.)

- The ability to fallback to a previous stable configuration through a single operation. (An active aspect of Bulk CM.)

7.2 Overview of IRPs related to CM

The Itf-N for CM 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 a Name Convention) are defined, used by this as well as other specifications For Telecom Management (TM) produced by 3GPP. All these IRPs are defined in separate 3GPP specifications~~, and listed in the Introduction clause~~.

7.3 ~~Basic~~ Kernel CM

The Kernel CM IRP provides the essential and common CM functions. A CM implementation will include the Kernel CM IRP and either one or both of the Basic CM IRP and the Bulk CM IRP. The Kernel CM IRP is specified in TS 32.661 through 32.664.

~~7.3.1 Passive CM~~

~~7.3.1.1 Real-time forwarding of CM-related event reports~~

The principal, but not the only, function of the Kernel CM IRP is to provide real-time forwarding of CM related event reports. During normal operation the NM is continuously informed by the managed subordinate entities about all network configuration changes, in accordance with the Network Resource Model (NRM) applied on the Itf-N. For this purpose the following CM-related event reports with regard to the ITU-T Recommendation X.721 [4], ITU-T Recommendation X.730 [5] and ITU-T Recommendation X.731 [6] are forwarded to the NM:

- Object creation;
- Object deletion;
- Attribute value change.

The real-time forwarding of these event reports occurs via appropriate filtering mechanisms ("discriminators" on CMIP interfaces, "subscription" on CORBA interfaces) located in the subordinate entity in accordance with ITU-T Recommendation X.734 [7] or OMG event/notification service. These filters may be controlled (i.e. created, modified and eventually deleted) locally in the subordinate entities or remotely by the NM (via the Itf-N) in order to ensure that only the event reports which fulfil pre-defined criteria can reach the superior NM. In a multiple manager environment each NM may have its own filtering mechanism within every subordinate entity, which is able to generate CM-related notifications.

It should be possible to pack multiple notifications together for sending to NM. This provides more efficient use of data communication resources. In order to pack multiple notifications, an EM/NE configurable parameter defines the maximum number of notifications to be packed together. Additionally an EM/NE configurable parameter defines the maximum time delay before the notifications have to be sent.

7.4 Basic CM

The Basic CM IRP provides a single operation style of CM as described for Basic CM in Clause 7.1. The Basic CM IRP is specified in TS 32.601 through 32.604

~~7.3.1.2~~7.4.1 Passive CM – Retrieval/synchronisation of CM-related information ~~on NM request~~

As long as the network is in operation and fault free, the update of the CM-related information on NM level is continuously ensured by the real-time forwarding of concerned reports as described in subclause 7.3.1. In case of faults (either on the NM or in a subordinate entity or on the communication link) it is possible that some CM-related event reports are lost. Therefore the CM-related information on the NM may become non-aligned with the real configuration of the network (depending on the strategy of the NM where to store network configuration information). In this case a synchronisation process may be necessary to align the CM-related information of the NM with the configuration information of the subordinate entities.

The retrieval or synchronisation ("alignment") of network configuration information between the NM and one or more of its subordinate entities can be triggered at any time by the NM.

There are two different alternatives for this synchronisation:

- via a read command with appropriate filtering;
- as an ordered sequence of CM-related event reports.

~~7.3.2~~7.4.2 Active CM

The optional Active CM functions of the Basic CM IRP provide the ability to modify Network Resources. There are three active CM functions:

- Create a Managed Object instance
- Delete a Managed Object instance(s)
- Modify the attributes of a Managed Object instance(s)

~~In the present document it is assumed that active CM is a task that can be performed only by the Element Managers (EMs) and/or local maintenance terminal actions. Thus it is outside the scope of the present document.~~

~~7.4~~7.5 Bulk CM

~~Itf-N shall provide~~The Bulk CM IRP provides efficient mechanisms to upload current CM data from the IRP Agent and download new CM data to the IRP Agent and to activate the new CM data. Bulk CM provides both active and passive CM functions as described in Clause 7.1. The Bulk CM IRP is specified in TS 32.611 through 32.615

~~It shall be possible to~~Bulk CM can transfer a CM file containing, for example, radio network parameters from the NM to the IRP Agent using a standardised file format and transfer mechanism. The IRP Agent shall also be capable of making the necessary configuration changes in its managed NEs, using the parameters and information contained in the transferred CM file.

~~The detailed requirements for Bulk CM are contained in 3GPP TS 32.602-1.~~

7.6 Common CM and NRM IRP Requirements

The following requirements apply to and are referenced by all CM and NRM IRPs:

- Any implementation of a CM IRP must also include the implementation of the Kernel CM IRP in order to claim 3GPP conformance.
- For each Managed Object Class (MOC) specified in an NRM IRP, the specification shall
  - Specify whether instances of the MOC may be created over the ~~i~~Itf-N.
  - Specify whether instances of the MOC may be deleted over the ~~i~~Itf-N.
  - Specify whether instances of the MOC may be modified over the ~~i~~Itf-N by indicating whether at least one attribute of the instance may be modified (i.e., is Read/Write).
  - Specify default values for the attributes of the MOC. When default values are specified for a MOC, each attribute may be given a single value (of the same type as the attribute), may be specified as vendor-specific (in which case no specific value is specified) or may be specified as having "no default". These default values may be used when an instance of the MOC is created or when attribute values are modified. How default attribute values are specifically to be used shall be specified in the Basic and Bulk CM IRP Information Specifications. The default values are applicable to both Basic and Bulk CM as well as to CM operations initiated by the agent itself.

3GPP TSG-SA5 (Telecom Management) **S5-026679**
Meeting #30, Tampere, FINLAND, 19 - 23 August 2002

CR-Form-v4

CHANGE REQUEST

⌘ **32.601** **CR** **001** ⌘ ev **-** ⌘ Current version: **4.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network **X** Core Network **X**

Title:	⌘	Add Active CM and Update Basic CM requirements
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**B**
Release:	⌘	Rel-5

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
REL-4	*(Release 4)*
REL-5	*(Release 5)*

Reason for change:	⌘	Create Release 5 of this IRP Requirements Specification from 32601-400.
Summary of change:	⌘	1) Update R4 Basic CM requirements for new R5 version of Basic CM 2) Add Active CM requirements
Consequences if not approved:	⌘	

Clauses affected:	⌘	Introduction, Clauses 1, 2, and 4		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications **X** O&M Specifications	⌘	32.602, 32.603
Other comments:	⌘	S5-026678 Rel-5 CR 32600 Add new Rel-5 functionality (Kernel CM, Active CM) - Parent CR S5-026679 Rel-5 CR 32601 Basic CM IRP Requirements - Child CR S5-026680 Rel-5 CR 32602 Basic CM IRP IS - Grandchild CR S5-026653 Rel-5 CR 32603 Basic CM IRP IS - Grandchild CR		

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 Network Elements (NEs) and Network Resources (NRs), and they may be initiated by the operator or by functions in the Operations Systems (OSs) or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service (QOS). The CM actions are initiated either as single actions on single NEs of the 3G network, or as part of a complex procedure involving actions on many resources/objects in one or several NEs.

~~Due to the growing number of specifications to model new services and Resource Models for Configuration Management (CM), as well as the expected growth in size of each of them from 3GPP Release 4 onwards, a new structure of the specifications is already needed in Release 4. This structure is needed for several reasons, but mainly to enable more independent development and release for each part, as well as a simpler document identification and version handling. Another benefit would be that it becomes easier for bodies outside 3GPP, such as the ITU-T, to refer to telecom management specifications from 3GPP. The new structure of the specifications does not lose any information or functionality supported by the Release 1999. The restructuring also includes defining new IRPs for the Network Resource Models (Generic, Core Network and UTRAN NRM).~~

~~Finally, the Name convention for Managed Objects (in Release 1999: 32.106-8) has been moved to a separate number series used for specifications common between several management areas (e.g. CM, FM, PM).~~

~~The following table shows an overview of the mapping between the old Release 1999 and new Release 4 CM specification structure.~~

~~Table: Mapping between Release '99 and the new Rel-4 specifications~~

~~R99 Old no.~~	~~Old (R99) specification title~~	~~Rel-4 New no.~~	~~New (Rel-4) specification title~~
~~32.106-1~~	~~3G Configuration Management: Concept and Requirements~~	~~32.600~~	~~**3G Configuration Management: Concept and High-level Requirements**~~
~~32.106-1~~	~~<Notification IRP requirements from 32.106-1 and 32.106-2>~~	~~32.301~~	~~**Notification IRP**: Requirements~~
~~32.106-2~~	~~Notification IRP: IS~~	~~32.302~~	~~Notification IRP: Information Service~~
~~32.106-3~~	~~Notification IRP: CORBA SS~~	~~32.303~~	~~Notification IRP: CORBA SS~~
~~32.106-4~~	~~Notification IRP: CMIP SS~~	~~32.304~~	~~Notification IRP: CMIP SS~~
~~32.106-8~~	~~Name convention for Managed Objects~~	~~32.300~~	~~**Name Convention for Managed Objects**~~
~~32.106-1~~	~~<Basic CM IRP IS requirements from 32.106-1 and 32.106-5>~~	~~32.601~~	~~**Basic CM IRP**: Requirements~~
~~32.106-5~~	~~Basic CM IRP IM (Intro & IS part)~~	~~32.602~~	~~Basic CM IRP: Information Service~~
~~32.106-6~~	~~Basic CM IRP CORBA SS (IS related part)~~	~~32.603~~	~~Basic CM IRP: CORBA SS~~
~~32.106-7~~	~~Basic CM IRP CMIP SS (IS related part)~~	~~32.604~~	~~Basic CM IRP: CMIP SS~~
~~32.106-1~~	~~<Basic CM IRP Generic NRM requirements from 32.106-1 and 32.106-5>~~	~~32.621~~	~~**Generic Network Resources IRP**: Requirements~~
~~32.106-5~~	~~Basic CM IRP IM (Generic NRM part)~~	~~32.622~~	~~Generic Network Resources IRP: NRM~~
~~32.106-6~~	~~Basic CM IRP CORBA SS (Generic NRM related part)~~	~~32.623~~	~~Generic Network Resources IRP: CORBA SS~~
~~32.106-7~~	~~Basic CM IRP CMIP SS (Generic NRM related part)~~	~~32.624~~	~~Generic Network Resources IRP: CMIP SS~~
~~32.106-1~~	~~<Basic CM IRP CN NRM requirements from 32.106-1 and 32.106-5>~~	~~32.631~~	~~**Core Network Resources IRP**: Requirements~~
~~32.106-5~~	~~Basic CM IRP IM (CN NRM part)~~	~~32.632~~	~~Core Network Resources IRP: NRM~~
~~32.106-6~~	~~Basic CM IRP CORBA SS (CN NRM related part)~~	~~32.633~~	~~Core Network Resources IRP: CORBA SS~~
~~32.106-7~~	~~Basic CM IRP CMIP SS (CN NRM related part)~~	~~32.634~~	~~Core Network Resources IRP: CMIP SS~~
~~32.106-1~~	~~<Basic CM IRP UTRAN NRM requirements from 32.106-1 and 32.106-5>~~	~~32.641~~	~~**UTRAN Network Resources IRP**: Requirements~~
~~32.106-5~~	~~Basic CM IRP IM (UTRAN NRM part)~~	~~32.642~~	~~UTRAN Network Resources IRP: NRM~~
~~32.106-6~~	~~Basic CM IRP CORBA SS (UTRAN NRM related part)~~	~~32.643~~	~~UTRAN Network Resources IRP: CORBA SS~~
~~32.106-7~~	~~Basic CM IRP CMIP SS (UTRAN NRM related part)~~	~~32.644~~	~~UTRAN Network Resources IRP: CMIP SS~~

4 Requirements

4.1 General Requirements

This requirements specification defines requirements for the IS for this IRP. As such, capabilities specified here as being required in the IS are not necessarily required in the product implementation. That which is required in the product implementation will be specified in the IS itself.

The following general and high-level requirements shall apply for the present IRP:

A. IRP-related requirements in 3GPP TS 32.101: "3G Telecom Management principles and high level requirements" [1].

B. IRP-related requirements in 3GPP TS 32.102: "3G Telecom Management architecture" [2].

C. IRP-related requirements in 3GPP TS 32.600: "3G Configuration Management: Concept and High-level Requirements" [3].

In addition to the above, the following more specific requirements shall apply:

1. The IS defined by this IRP shall enable an NM to operate on (access) any of the NRMs defined in [4], [5], [6] and [7].

2. The IS defined by this IRP shall as far as possible be independent of any specific definitions of MOCs, attributes etc. in the NRMs referred to in item 1.

3. The IS specified by this IRP shall assume that when this IRP is implemented that the Kernel CM IRP is also implemented.

4.2 Passive CM Requirements

The IS defined by this IRP shall include the following operations that may be invoked by the IRP Manager to retrieve management information from the MIB maintained by the IRPAgent:

- An operation to retrieve the value of attributes from one or more managed object instances.
- An operation to retrieve the containment relationships between the managed object instances of a containment tree of managed objects.
- An operation to retrieve the Basic CM IRP versions that are supported by the IRPAgent.
- An operation to cancel a previously initiated operation if it has not completed. This operation shall, as a minimum, be able to cancel the operation that retrieves attributes. It may be specified to cancel any operation.

4.3 Active CM Requirements

Active CM requirements are specified as additions to Passive CM requirements and not intended to be implemented without implementation of Passive CM.

The IS defined by this IRP shall include the following operations that may be invoked by the IRP Manager to communicate management information to the IRPAgent specifying changes to be made to the MIB maintained by that IRPAgent:

- An operation to create an instance of a managed object.
- An operation to delete one or more instances of managed objects.
- An operation to modify one or more attributes of one or more instances of managed objects.

TS 32.600 [3] specifies the information that must provided in the NRM specifications on a per managed object basis to support these Active CM operations.

3GPP TSG-SA5 (Telecom Management) **S5-026680**
Meeting #30, Tampere, FINLAND, 19 - 23 August 2002

CR-Form-v4

CHANGE REQUEST

⌘ **32.602** **CR** **002** ⌘ ev **-** ⌘ Current version: **4.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network **X** Core Network **X**

Title:	⌘	Add Active CM and new methodology, Remove CM Notifications (moved to Kernel CM - 32.66x)
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**B**
Release:	⌘	Rel-5

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 *(GSM Phase 2)*
R96 *(Release 1996)*
R97 *(Release 1997)*
R98 *(Release 1998)*
R99 *(Release 1999)*
REL-4 *(Release 4)*
REL-5 *(Release 5)*

Reason for change:	⌘	Create Release 5 of this IRP Informaion Service Specification from 32602-410
Summary of change:	⌘	1) Remove CM Notifications (moved to Kernel CM) 2) Add Active CM 3) Update to new methodology
Consequences if not approved:	⌘	
Clauses affected:	⌘	Introduction, Clauses 1, 2, 6, and 7
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	S5-026678 Rel-5 CR 32600 Add new Rel-5 functionality (Kernel CM, Active CM) - Parent CR S5-026679 Rel-5 CR 32601 Basic CM IRP Requirements - Child CR S5-026680 Rel-5 CR 32602 Basic CM IRP IS - Grandchild CR S5-026653 Rel-5 CR 32603 Basic CM IRP IS - Grandchild CR

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 Network Elements (NEs) and Network Resources (NRs), and they may be initiated by the operator or by functions in the Operations Systems (OSs) or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service (QOS). The CM actions are initiated either as single actions on single NEs of the 3G network, or as part of a complex procedure involving actions on many resources/objects in one or several NEs.

Due to the growing number of specifications to model new services and Resource Models for Configuration Management (CM), as well as the expected growth in size of each of them from 3GPP Release 4 onwards, a new structure of the specifications is already needed in Release 4. This structure is needed for several reasons, but mainly to enable more independent development and release for each part, as well as a simpler document identification and version handling. Another benefit would be that it becomes easier for bodies outside 3GPP, such as the ITU-T, to refer to telecom management specifications from 3GPP. The new structure of the specifications does not lose any information or functionality supported by the Release 1999. The restructuring also includes defining new IRPs for the Network Resource Models (Generic, Core Network and UTRAN NRM).

Finally, the Name convention for Managed Objects (in Release 1999: 32.106-8) has been moved to a separate number series used for specifications common between several management areas (e.g. CM, FM, PM).

The following table shows an overview of the mapping between the old Release 1999 and new Release 4 CM specification structure.

Table: Mapping between Release '99 and the new Rel-4 specifications

R99 Old no.	Old (R99) specification title	Rel-4 New no.	New (Rel-4) specification title
32.106-1	3G Configuration Management: Concept and Requirements	32.600	**3G Configuration Management: Concept and High-level Requirements**
32.106-1	<Notification IRP requirements from 32.106-1 and 32.106-2>	32.301	**Notification IRP**: Requirements
32.106-2	Notification IRP: IS	32.302	Notification IRP: Information Service
32.106-3	Notification IRP: CORBA SS	32.303	Notification IRP: CORBA SS
32.106-4	Notification IRP: CMIP SS	32.304	Notification IRP: CMIP SS
32.106-8	Name convention for Managed Objects	32.300	**Name Convention for Managed Objects**
32.106-1	<Basic CM IRP IS requirements from 32.106-1 and 32.106-5>	32.601	**Basic CM IRP**: Requirements
32.106-5	Basic CM IRP IM (Intro & IS part)	32.602	Basic CM IRP: Information Service
32.106-6	Basic CM IRP CORBA SS (IS related part)	32.603	Basic CM IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (IS related part)	32.604	Basic CM IRP: CMIP SS
32.106-1	<Basic CM IRP Generic NRM requirements from 32.106-1 and 32.106-5>	32.621	**Generic Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (Generic NRM part)	32.622	Generic Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (Generic NRM related part)	32.623	Generic Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (Generic NRM related part)	32.624	Generic Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP CN NRM requirements from 32.106-1 and 32.106-5>	32.631	**Core Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (CN NRM part)	32.632	Core Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (CN NRM related part)	32.633	Core Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (CN NRM related part)	32.634	Core Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP UTRAN NRM requirements from 32.106-1 and 32.106-5>	32.641	**UTRAN Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (UTRAN NRM part)	32.642	UTRAN Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (UTRAN NRM related part)	32.643	UTRAN Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (UTRAN NRM related part)	32.644	UTRAN Network Resources IRP: CMIP SS

1 Scope

The present document defines a component of an Integration Reference Point (IRP) through which an 'IRPAgent' (typically an Element Manager or Network Element) can communicate basic Configuration Management related information to one or several 'IRPManagers' (typically Network Managers).

~~This version of the IRP is mainly intended for "passive management" of high-level network configuration and status information as required by a Network Manager.~~

~~The Configuration Management (CM) area is very large. The intention is to split the specification of the related interfaces in several IRPs – as described in the Introduction clause above. An important aspect of such a split is that the Network Resource Models (NRMs) defined in different IRPs containing NRMs are consistent, and that NRMs supported by an IRPAgent implementation can be accessed as one coherent model through one IRP Information Service. The Basic CM IRP: IS defined herein provides one such Information Service.~~

~~Thus, to summarize, the Basic CM IRP: IS defined in the present document has the following main purpose: to define an interface for retrieval of Configuration Management information.~~

The function of this Basic CM IRP Information Service is to define an interface for the retrieval and modification of Configuration Management Information.

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 32.302: "Telecommunication Management; Configuration Management; Part 2: Notification Integration Reference Point; Information Service".

[4] 3GPP TS 32.~~112~~312: "Generic IRP Management: Information Service".

[5] Void

[6] Void

[7] ITU-T Recommendation X.710 (~~1991~~1997): "Common Management Information Service Definition for CCITT Applications".

[8] ITU-T Recommendation X.721 (02/92): "Information Technology - Open Systems Interconnection – Structure of Management Information: Definition of Management Information".

[9] ITU-T Recommendation X.730 (01/92): "Information Technology - Open Systems Interconnection – Systems Management: Object Management Function".

[10] ITU-T Recommendation X.733 (02/92): "Information Technology - Open Systems Interconnection - Alarm Reporting Function".

[11] Void

[12] Void

[13] 3GPP TS 32.300: "Name Convention for Managed Objects".

[14] 3GPP TS 32.600: "3G Configuration Management: Concepts and requirements".

4 System overview

4.1 System context

~~Figure 4.1 and Figure 4.2 identify system contexts of the subject IRP in terms of its implementation called IRPAgent and the user of the IRPAgent, called IRPManager. For a definition of IRPManager and IRPAgent, see 3GPP TS 32.102 [2].~~

~~The IRPAgent implements and supports the Basic CM IRP: IS. The IRPAgent can be an Element Manager (EM) or a mediator that interfaces one or more NEs (see Figure 4.1), or it can be a Network Element (NE) (see Figure 4.2). In the former case, the interfaces (represented by a thick dotted line) between the EM and the NEs are not subject of this IRP.~~

~~An IRPManager using this IRP shall choose one of the two System Contexts defined here, for each NE. For instance, if an EM is responsible for managing a number of NEs, the NM shall access this IRP through the EM and not directly to those NEs. For another IRP though, the System Context may be different.~~

Figure 4.1 and 4.2 identify system contexts of the IRP defined by the present specification in terms of its implementation called IRPAgent and the user of the IRPAgent, called IRPManager. For a definition of IRPManager and IRPAgent, see 3GPP TS 32.102 [2].

The IRPAgent implements and supports this IRP. The IRPAgent can reside in an Element Manager (EM) or a Network Element (NE) (see also [2] clause 8). In the former case, the interfaces (represented by a thick dotted line) between the EM and the NEs is not the subject of this IRP.

An IRPManager using this IRP shall choose one of the two System Contexts defined here, for each NE. For instance, if an EM is responsible for managing a number of NEs, the NM shall access this IRP through the EM and not directly to those NEs. For another IRP though, the System Context may be different.

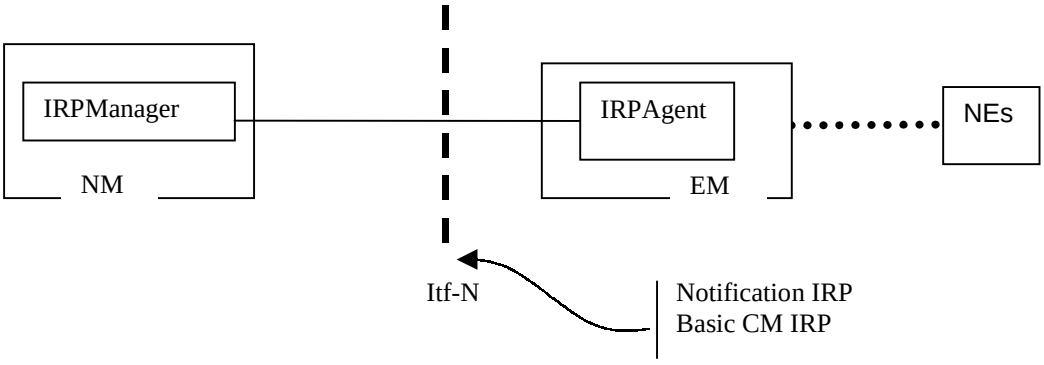


Figure 4.1: System Context A

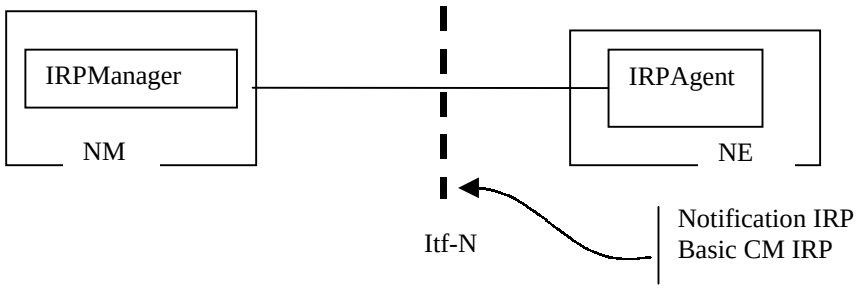


Figure 4.2: System Context B

4.2 Compliance rules

For general definitions of compliance rules related to qualifiers (Mandatory/Optional/Conditional) for *operations*, *notifications and parameters* (of operations and notifications) please refer to 3GPP TS 32.102 [2].

An IRPAgent that incorporates vendor-specific extensions shall support normal communication with a 3GPP SA5-compliant IRPManager with respect to all Mandatory and Optional managed object classes, attributes, associations, operations, parameters and notifications without requiring the IRPManager to have any knowledge of the extensions.

Given that

- rules for vendor-specific extensions remain to be fully specified, and
- many scenarios under which IRPManager and IRPAgent interwork may exist,

it is recognised that in Release 4/5 the IRPManager, even though it is not required to have knowledge of vendor-specific extensions, may be required to be implemented with an awareness that extensions can exist and behave accordingly.

5 Modelling approach

This clause identifies the modelling approach adopted and used in this IRP.

As described in 3GPP TS 32.101 [1], an IRP comprises the following components:

(1) an IRP Information Model that specifies the interface in a protocol neutral manner, defined as an Information Service and/or one or more Network Resource Models,

(2) a number of IRP Solution Sets that provide the actual realization of the operations and notifications defined in the IRP Information Model for each protocol environment.

The present document defines one such Information Service – the Basic CM IRP: IS.

The IRP Information Service is a specification of the *operations* and *notifications* that are visible over the IRP. These operations/notifications are generic in the sense that they do not specify the Managed Objects that are retrieved/manipulated/informed about over the interface, and thus this IS is independent of the NRM being managed.

5.1 IRP Information Service modelling approach

The IRP Information Service of the subject IRP specifies a number of protocol-independent operations and notifications that are needed by an IRPManager to retrieve CM information from an IRPAgent.

The operations and notifications of the IRP Information Service are mainly based on the principles of the Common Management Information Service (CMIS) defined in ITU-T X.710 [7] and ITU-T X.721 [8] (M-GET etc.). Note however, that the Information Service of the subject IRP is focused on the operations and notifications needed for basic CM purposes and thus only covers a subset of the operations/notifications defined in ITU-T X.710 [7]/ITU-T X.721 [8].

It is expected that most Solution Sets will implement the operations and notifications by mapping them to standard operations (and possibly standard notifications) that are applicable in the corresponding protocol environment.
A CMIP Solution Set should for instance map the operations to the more generic operations defined in CMIS, an SNMP Solution Set should map the operations to applicable SNMP operations, and a CORBA Solution Set should map the operations to applicable OMG/CORBA services.

~~6 IRP Information Service~~

~~This subclause specifies the *operations* and *notifications* that are visible over this IRP. These operations are generic in the sense that they do not specify the MOs that are retrieved/manipulated over the interface.~~

6.1 Interfaces

Figure 6.1 illustrates the operations and notifications defined as interfaces implemented and used by IRPAgent and IRPManager, described using UML notation (Interface in IRP Information Model is identical to concepts conveyed by stereotype <<interface>> of UML). Parameters and return status are not indicated.

Two interfaces are defined. One is called `BasicCmIRPOperations`. This interface defines operations implemented by IRPAgent and used (or called) by IRPManager. The other is called `BasicCmIRPNotifications`. This interface defines notifications implemented by IRPManager and used by IRPAgent.

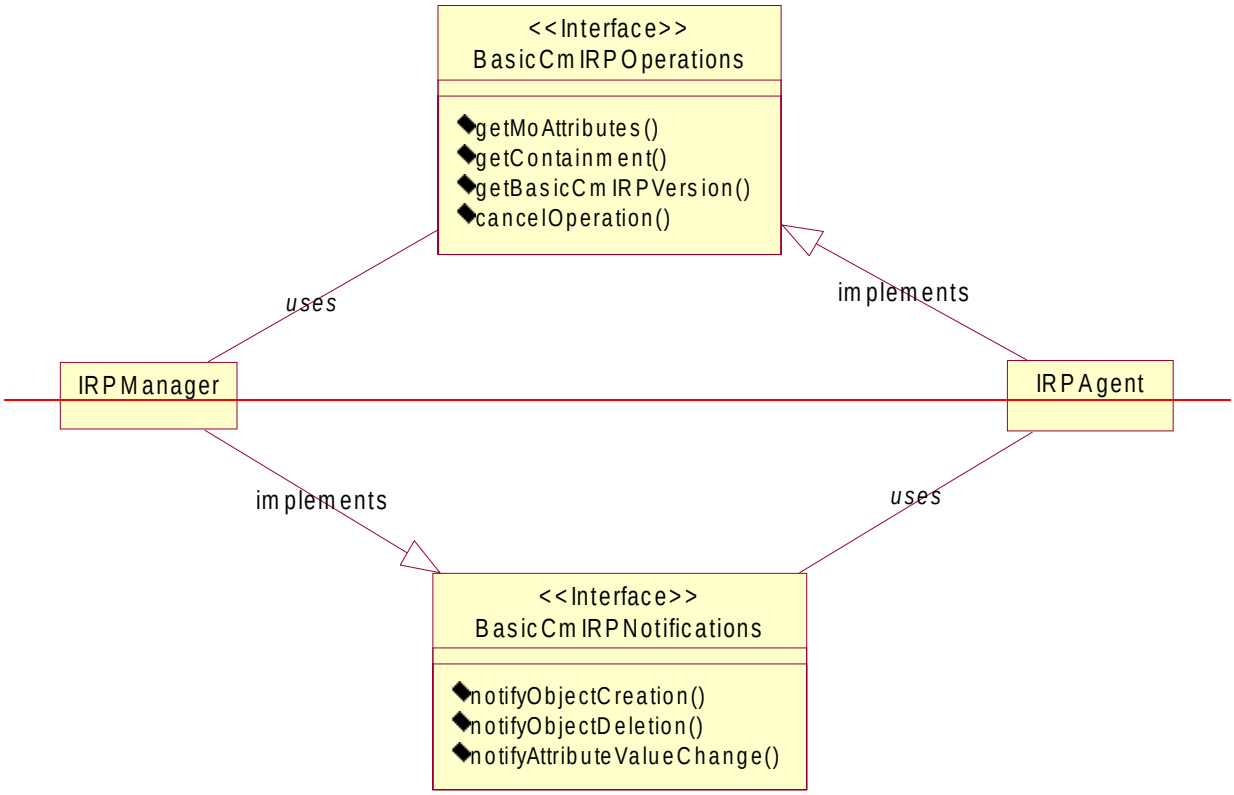


Figure 6.1: UML Interface Class Diagram

6.2 Operations

6.2.1 Operation `getMoAttributes` (M)

This operation is invoked by IRPManager to request the retrieval of management information (Managed Object attribute names and values) from the MIB maintained by IRPAgent. One or several Managed Objects may be retrieved based on the containment hierarchy. The operation corresponds to the M-GET service defined by CMIS (ITU-T X.710 [7]).

A Solution Set may choose to split this operation in several operations (e.g. operations to get "handlers" or "iterators" to Managed Objects fulfilling the scope/filter criteria and other operations to retrieve attribute names/values from these "handlers").

Table 1: Parameters of `getMoAttributes`

Name	Qualifier	Description
invokeIdentifierIn	Input, C	This parameter identifies the current invocation in both IRPManager and IRPAgent. This parameter can be used together with the 'cancelOperation' operation to cancel an on-going 'getMOAttributes' operation.
baseObjectInstance	Input, M	The MO where the search starts. This is a full Distinguished Name according to 3GPP TS 32.300 [13].
scope	Input, M	This parameter defines how many levels of the containment hierarchy to search (i.e. apply the filter defined below). The search starts from the MO given by the baseObjectInstance parameter. The levels of search that may be performed are: the base object alone (default); the n-th level subordinates of the base object; the base object and all of its subordinates down to and including the n-th level; the base object and all of its subordinates.
filter	Input, M	This parameter defines a filter test to be applied to the scoped Managed Object(s). If the filter is empty, all of the managed objects included by the scope are selected. The actual syntax and capabilities of the filter is Solution Set specific. However, each Solution Set should support a filter consisting of one or several assertions that may be grouped using the logical operators AND, OR and NOT. Each assertion is a logical expression of attribute existence, attribute value comparison ("equal to X, less than Y" etc.) and MO Class.
attributeListIn	Input, M	This parameter identifies the attributes to be returned by this operation. In the current version, only the semantics "Return all attributes" shall be supported. An empty list means "Return all attributes". For future releases the possibility to specify a list of attributes is expected.
invokeIdentifierOut	Output, M	This parameter identifies the current invocation in both IRPManager and IRPAgent. This parameter can be used together with the 'cancelOperation' operation to cancel an on-going 'getMOAttributes' operation.
managedObjectClass	Output, M	For each returned MO: The class of the MO.
managedObjectInstance	Output, M	For each returned MO: The name of the MO. This is a full Distinguished Name according to 3GPP TS 32.300 [13].
attributeListOut	Output, M	For each returned MO: A list of name/value pairs for the MO attributes.
status	Output, M	(a) Operation succeeded, or (b) Operation failed because of specified or unspecified reason.

6.2.2 Operation `getContainment` (O)

This (optional) operation is only intended for retrieval of the containment relations from the MIB.

The output parameter 'containment' of the operation shall contain a list of all Managed Object instances in the MIB maintained by IRPAgent (or a subset starting from a given base object) including containment information (naming tree).

The structure and format of the output parameter 'containment' are Solution Set dependent.

Table 2: Parameters of `getContainment`

Name	Qualifier	Description
invokeIdentifierIn	Input, C	This parameter identifies the current invocation in both IRPManager and IRPAgent. This parameter can be used together with the 'cancelOperation' operation to cancel an on-going 'getContainment' operation.
baseObject Instance	Input, M	The MO where the search starts. This is a full Distinguished Name according to 3GPP TS 32.300 [13].
scope	Input, O	This parameter gives a value N defining how many levels of the containment hierarchy from the baseObjectInstance to include in the result. The levels of inclusion that may be performed are: the base object alone (default); the n-th level subordinates of the base object; the base object and all of its subordinates down to and including the n-th level; the base object and all of its subordinates.
invokeIdentifierOut	Output, M	This parameter identifies the current invocation in both IRPManager and IRPAgent. This parameter can be used together with the 'cancelOperation' operation to cancel an on-going 'getContainment' operation.
containment	Output, M	A list of DN of all Managed Object instances that satisfy the scope.
status	Output, M	(a) Operation succeeded, or (b) Operation failed because of specified or unspecified reason.

6.2.3 Operation `getBasicCmIRPVersion` (M)

IRPManager wishes to find out the Basic CM IRP SS version(s) supported by IRPAgent. IRPAgent shall respond with a list of supported Basic CM IRP SS versions. Since the present document defines the first IRP version, implementation of IRPAgent in compliance to this version shall return with one version number in the list.

Table 3: Parameters of `getBasicCmIRPVersion`

Name	Qualifier	Description
versionNumberList	Output, M	It indicates one or more SS version numbers supported by the IRPAgent. The IRP document version number (sometimes called "IRPVersion" or "version number") string is used to identify which specification version(s) an implementation is conformant to. Each string in this set is derived using a rule described in the "Generic IRP" [4].
status	Output, M	(a) Operation succeeded in that versionNumberList contains valid result. (b) Operation failed. Output parameter versionNumberList may contain invalid result.

6.2.4 Operation `cancelOperation` (O)

IRPManager invokes this operation to cancel an on-going Basic CM IRP operation it issued before. Presently the Basic CM IRP operations that can be cancelled by invoking 'cancelOperation' are 'getMOAttributes' and 'getContainment'.

Table 4: Parameters of `cancelOperation`

Name	Qualifier	Description
invokeIdentifier	Input, M	This parameter identifies an on-going Basic CM IRP operation to be cancelled.
status	Output, M	(a) Operation succeeded. (b) Operation failed because of specified or unspecified reason.

6.3 Notifications

6.3.1 General

Operations that IRPManager uses to manage subscription to receive notifications are specified in Notification IRP IS 3GPP TS 32.302 [3]. Notification IRP IS [3] does not define any specific notification but instead defines information that is commonly found in notifications defined by other IRPs. This information is called notificationHeader.

Thus, the commonly carried attributes in each notification are collectively called `notificationHeader` in the present document. The attribute names and their qualifiers are listed in Table 4.

Table 4: Notification Header

Attributes defined in 3GPP TS 32.302 [3]	Comment	Qualifier for use in this IS
managedObjectClass	(mapped to objectClass in [3])	M
managedObjectInstance	(mapped to objectInstance in [3])	M
notificationId		O
eventTime		M
systemDN		C
eventType	(mapped to notificationType in [3] – see Annex A)	M

The following subclauses define specific notifications relevant for Basic CM IRP.

6.3.2 Notification `notifyObjectCreation` (O)

IRPAgent notifies the subscribed IRPManager that a new Managed Object has been created and that the new object satisfies the filter constraint expressed in IRPManager's `subscribe` operation (see 3GPP TS 32.302 [3]). This notification is based on the `objectCreation` notification type specified in ITU-T X.721 [8] and ITU-T X.730 [9] (difference compared to these specifications are indicated in the description below).

Table 5: Parameters for `notifyObjectCreation`

Name	Qualifier	Description
notificationHeader	Input, M	See Table 4: Notification Header.
correlatedNotifications	Input, O	A set of notifications that are correlated to the subject notification. Defined in ITU-T X.733 [10].
additionalText	Input, O	It can contain further information on the creation of the MO.
sourceIndicator	Input, O	This parameter, when present, indicates the source of the operation that led to the generation of this notification. It can have one of the following values: resource operation: The notification was generated in response to an internal operation of the resource; management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; unknown: It is not possible to determine the source of the operation.
attributeList	Input, O	The attributes (name/value pairs) of the created MO.

6.3.3 Notification `notifyObjectDeletion` (O)

IRPAgent notifies the subscribed IRPManager of a deleted Managed Object. The IRPAgent invokes this notification because the subject notification satisfies the filter constraint expressed in the IRPManager `subscribe` operation (see 3GPP TS 32.302 [3]). This notification is based on the `objectDeletion` notification type specified in ITU-T X.721 [8] and ITU-T X.730 [9] (difference compared to these specifications are indicated in the description below).

Note that when a Managed Object is deleted, all subordinate Managed Objects (i.e. the complete sub-tree of the MIB) are also deleted. Furthermore, all associations where the Managed Object participates are deleted.

Table 6: Parameters for `notifyObjectDeletion`

Name	Qualifier	Description
notificationHeader	Input, M	See Table 4: Notification Header.
correlatedNotifications	Input, O	A set of notifications that are correlated to the subject notification. Defined in ITU-T X.733 [10]..
additionalText	Input, O	It can contain further information on the deleted MO.
sourceIndicator	Input, O	This parameter, when present, indicates the source of the operation that led to the generation of this notification type. It can have one of the following values: resource operation: The notification was generated in response to an internal operation of the resource; management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; unknown: It is not possible to determine the source of the operation.
attributeList	Input, O	The attributes (name/value pairs) of the deleted MO.

6.3.4 Notification `notifyAttributeValueChange` (O)

IRPAgent notifies the subscribed IRPManager of a change of one or several attributes of a Managed Object in the NRM. The IRPAgent invokes this notification because the subject notification satisfies the filter constraint expressed in the IRPManager `subscribe` operation (see 3GPP TS 32.302 [3]). This notification is based on the `attributeValueChange` notification type specified in ITU-T X.721 [8] and ITU-T X.730 [9] (difference compared to these specifications are indicated in table 7).

Table 7: Parameters for `notifyAttributeValueChange`

Name	Qualifier	Description
notificationHeader	Input, M	See Table 4: Notification Header.
correlatedNotifications	Input, O	A set of notifications that are correlated to the subject notification. Defined in ITU-T X.733 [10].
additionalText	Input, O	It can contain further information on the attribute change of the MO.
sourceIndicator	Input, O	This parameter, when present, indicates the source of the operation that led to the generation of this notification type. It can have one of the following values: resource operation: The notification was generated in response to an internal operation of the resource; management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; unknown: It is not possible to determine the source of the operation.
attributeValueChange Definition	Input, M	The changed attributes (name/value pairs) of the MO (with both new and, optionally, old values).

6. Information Object Classes

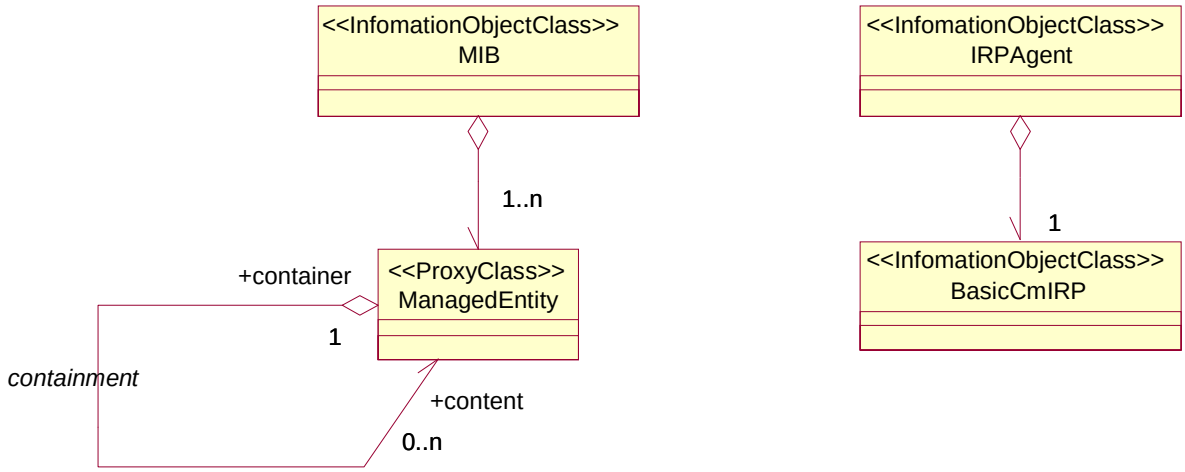
6.1 Imported information entities and local labels

Label reference	Local label
32.622, information object class, `Top`	`Top`
32.622, information object class, `IRPAgent`	`IRPAgent`
32.622, information object class, `GenericIRP`	`GenericIRP`
32.312, information object class, `ManagedGenericIRP`	`ManagedGenericIRP`

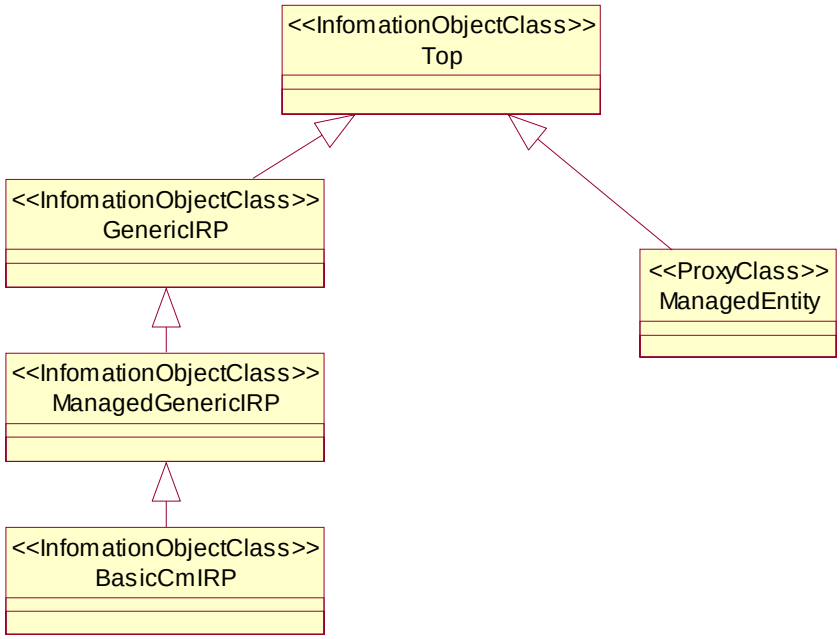
6.2 Class diagram

This sub-clause introduces the set of information object classes (IOCs) that encapsulate information within the `IRPAgent`. The intent is to identify the information required for the BasicCmIRP Agent implementation of its operations and notification emission. This sub-clause provides the overview of all support object classes in UML. Subsequent sub-clauses provide more detailed specification of various aspects of these support object classes.

6.2.1 Attributes and relationships



6.2.2 Inheritance



6.3 Information Object Class Definitions

6.3.1 BasicCmIRP

6.3.1.1 Definition

BasicCmIRP is the representation of the basic configuration management capabilities specified by this specification. This IOC inherits from ManagedGenericIRP IOC specified in TS 32.312 [4].

6.3.2 ManagedEntity

6.3.2.1 Definition

The IOC ManagedEntity represents the role that can be played by an instance of an IOC defined in Network Resources Models, e.g. Generic Network Resource Model, Core Network Resource Model, UTRAN Network Resource Model or GERAN Network Resource Models. ManagedEntity is used in the specification of Basic CM IRP operations to represent an instance of an IOC defined in these Network Resource Models.

6.4 Information relationship definitions

6.4.1 containment (M)

6.4.1.1 Definition

This represents the relationship containment as defined in ITU-T Rec X.720 [].

6.4.1.2 Role

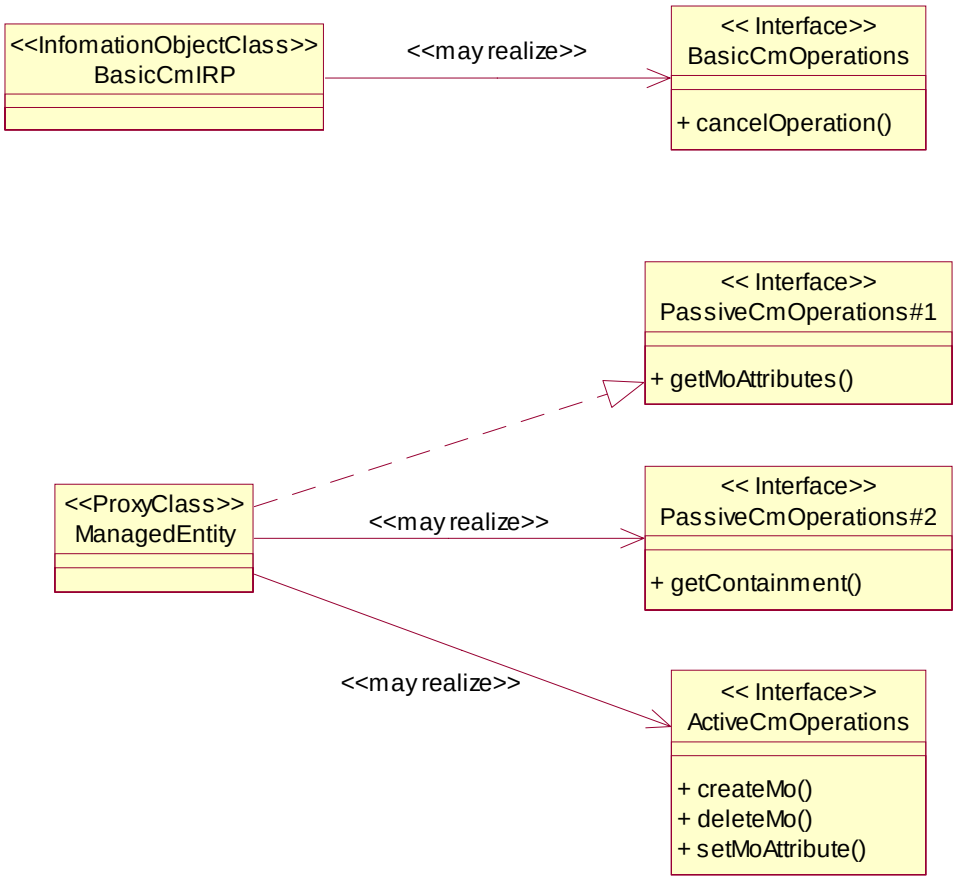
Name	Definition
container	It represents the capability, for an instance of a ManagedEntity, to contain other objects.
content	It represents the capability, for an instance of a ManagedEntity, to be contained in another object.

6.4.1.3 Constraint

Name	Definition
inv_noSelfContainment	No instance of the IOC ManagedEntity can play both roles container and content in the same instance of the relationship containment.

7 Interface Definition

7.1 Class diagram



7.2 Generic rules

Rule 1: Each operation with at least one input parameter supports a pre-condition valid_input_parameter which indicates that all input parameters shall be valid with regards to their information type. Additionally, each such operation supports an exception operation_failed_invalid_input_parameter which is raised when pre-condition valid_input_parameter is false. The exception has the same entry and exit state.

Rule 2: Each operation with at least one optional input parameter supports a set of pre-conditions supported_optional_input_parameter_xxx where "xxx" is the name of the optional input parameter and the pre-condition indicates that the operation supports the named optional input parameter. Additionally, each such operation supports an exception operation_failed_unsupported_optional_input_parameter_xxx which is raised when (a) the pre-condition supported_optional_input_parameter_xxx is false and (b) the named optional input parameter is carrying information. The exception has the same entry and exit state.

Rule 3: Each operation shall support a generic exception operation_failed_internal_problem that is raised when an internal problem occurs and that the operation cannot be completed. The exception has the same entry and exit state.

7.3 Interface PassiveCmIRPOperations#1

7.3.1 getMoAttributes (M)

7.3.1.1 Definition

This operation is invoked by IRPManager to request the retrieval of management information (Managed Object attribute names and values) from the MIB maintained by IRPAgent. One or several Managed Objects may be retrieved - based on the containment hierarchy. This operation provides functionality that is similar to that provided by the M-GET service defined by CMIS (ITU-T X.710 [7]).
A Solution Set may choose to split this operation in several operations (e.g. operations to get "handlers" or "iterators" to Managed Objects fulfilling the scope/filter criteria and other operations to retrieve attribute names/values from these "handlers").

7.3.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
invokeIdentifierIn	C	A unique identifier that is Solution Set dependent.	This parameter identifies the current invocation. This parameter is used in the 'cancelOperation' operation to cancel an on-going 'getMOAttributes' operation.
baseObjectInstance	M	DistinguishedName	The MO instance that is to be used as the starting point for the selection of managed objects to which the filter (when supplied) is to be applied. ~~The MO where the search starts.~~ This is a full Distinguished Name according to 3GPP TS 32.300 [13].
scope	M	SEQUENCE < ENUM { BASE_OBJECT_ONLY, NTH_LEVEL_SUBORDINATES, BASE_NTH_LEVEL, BASE_ALL}, theLevel> Note: theLevel contains valid information if NTH_LEVEL_SUBORDINATES or BASE_NTH_LEVEL is used. ~~Integer>~~	This parameter defines how many levels of the containment hierarchy to select for the filter defined below. The selection starts from the MO given by the baseObjectInstance parameter. The levels of selection that may be performed are: ~~This parameter defines how many levels of the containment hierarchy to search (i.e. apply the filter defined below). The search starts from the MO given by the baseObjectInstance parameter. The levels of search that may be performed are:~~ BASE OBJECT ONLY: the base object alone (default); NTH LEVEL SUBORDINATES: the n-th level subordinates of the base object; BASE NTH LEVEL: the base object and all of its subordinates down to and including the n-th level; BASE ALL: the base object and all of its subordinates.
filter	M	See Comment	This parameter defines a filter test to be applied to the scoped Managed Object(s). If the filter is empty, all of the managed objects included by the scope are selected. The actual syntax and capabilities of the filter is

			Solution Set specific. However, each Solution Set should support a filter consisting of one or several assertions that may be grouped using the logical operators AND, OR and NOT. Each assertion is a logical expression of attribute existence, attribute value comparison ("equal to X, less than Y" etc.) and MO Class.
attributeListIn	M	LIST OF AttributeName	This parameter identifies the attributes to be returned by this operation. In R99, only the semantics "Return all attributes" shall be supported. An empty list means "Return all attributes". For future releases the possibility to specify a list of attributes is expected.

7.3.1.3 Output Parameters

Name	Qualifier	Matching Information	Comment
invokeIdentifierOut	M (Note)	invokeIdentifierIn from the input parameters of this operation	This parameter identifies the current invocation in both IRPManager and IRPAgent. This parameter can be used together with the 'cancelOperation' operation to cancel an on-going 'getMOAttributes' operation.
managedObjectClass	M	ManagedEntity.objectClass	For each returned MO: The class of the MO.
managedObjectInstance	M	ManagedEntity.distinguishedName	For each returned MO: The name of the MO. This is a full Distinguished Name according to 3GPP TS 32.~~106-8~~300[13].
attributeListOut	M	LIST OF SEQUENCE<name OF ManagedEntity.anAttribute, value OF ManagedEntity.anAttribute>	For each returned MO: A list of name/value pairs for MO.
status	M	ENUM (OperationSucceeded, OperationFailed~~,~~)	An operation may fail because of a specified or unspecified reason.

Note: This parameter is meaningful only if the IRPAgent supports the cancelOperation.

7.3.1.4 Pre-condition

baseObjectExists

Assertion Name	Definition
baseObjectExists	The ManagedEntity instance specified by the baseObjectInstance parameter exists.

7.3.1.5 Post-condition

None specific.

7.3.1.6 Exceptions

Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.
duplicateInvocation	**Condition:** The invoke identifier specified was allocated to another operation **Returned Information:** The output parameter status. **Exit state:** Entry state. **Note:** This exception is conditional and applies only to Solution Sets where it is meaningful.
resourceLimitation	**Condition:** Operation not performed due to resource limitation. **Returned Information:** The output parameter status. **Exit state:** Entry state.
operationCancelled	**Condition:** Operation cancelled by cancelOperation operation. **Returned Information:** The output parameter status. **Exit state:** Entry state.
complexityLimitation	**Condition:** Operation not performed because a parameter was too complex. **Returned Information:** The output parameter status. **Exit state:** Entry state.

7.4 Interface PassiveCmIRPOperations#2

7.4.1 getContainment (O)

7.4.1.1 Definition

This (optional) operation is only intended for retrieval of the containment relations from the MIB.

7.4.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
invokeIdentifierIn	~~M~~C	A unique identifier that is Solution Set dependent.	This parameter identifies the current invocation in both IRPManager and IRPAgent. This parameter can be used together with the 'cancelOperation' operation to cancel an on-going 'getContainment' operation.
baseObjectInstance	M	DistinguishedName	The MO instance that is to be used as the starting point for the selection of managed objects to which the filter (when supplied) is to be applied. ~~The MO where the search starts.~~ This is a full Distinguished Name according to 3GPP TS 32.106-8 [13].
scope	O	See corresponding parameter in getMOAttributes.~~SEQUENCE < ENUM { BASE_OBJECT_ONLY, NTH_LEVEL_SUBORDINATES, BASE_NTH_LEVEL, BASE_ALL}, Integer>~~	See corresponding parameter in getMOAttributes.~~The levels of inclusion that may be performed are:~~ ~~the base object alone (default);~~ ~~the n-th level subordinates of the base object;~~ ~~the base object and all of its subordinates down to and including the n-th level;~~ • ~~the base object and all of its subordinates.~~

7.4.1.3 Output Parameters

The output parameter 'containment' of the operation shall contain a list of all Managed Object instances in the MIB maintained by IRPAgent (or a subset starting from a given base object) including containment information (naming tree).

The structure and format of the output parameter 'containment' are Solution Set dependent.

Name	Qualifier	Matching Information	Comment
containment	M	LIST OF ManagedEntity.distinguishedName	A list of DN of all Managed Object instances that satisfy the scope.
invokeIdentifierOut	M (Note)	invokeIdentifierIn from the input parameters of this operation	This parameter identifies the current invocation. . This parameter is used in 'cancelOperation' operation to cancel an on-going 'getContainment' operation.
status	M	ENUM (OperationSucceeded, OperationFailed)	An operation may fail because of a specified or unspecified reason.

Note: This parameter is meaningful only if the IRPAgent supports the cancelOperation.

7.4.1.4 Pre-condition

baseObjectExists

Assertion Name	Definition
baseObjectExists	The ManagedEntity instance specified by the baseObjectInstance parameter exists.

7.4.1.5 Post-condition

None specific

7.4.1.6 Exceptions

Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.
duplicateInvocation	**Condition:** The invoke identifier specified was allocated to another operation **Returned Information:** The output parameter status. **Exit state:** Entry state. **Note:** This exception is conditional and applies only to Solution Sets where it is meaningful.
resourceLimitation	**Condition:** Operation not performed due to resource limitation. **Returned Information:** The output parameter status. **Exit state:** Entry state.
operationCancelled	**Condition:** Operation cancelled by cancelOperation operation. **Returned Information:** The output parameter status. **Exit state:** Entry state.
complexityLimitation	**Condition:** Operation not performed because a parameter was too complex. **Returned Information:** The output parameter status. **Exit state:** Entry state.

7.5 Interface BasicCmIRPOperations

7.5.1 cancelOperation (~~M~~O)

7.5.1.1 Definition

IRPManager invokes this operation to cancel an on-going Basic CM IRP operation it issued before. Presently the Basic CM IRP operations that can be cancelled by invoking 'cancelOperation' are 'getMOAttributes' and 'getContainment'.

7.5.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
~~nvokeIdentifier~~invokeIdentifierIn	M	A unique identifier that is Solution Set dependent.	This parameter identifies an on-going Basic CM IRP operation to be cancelled.

7.5.1.3 Output Parameters

Name	Qualifier	Matching Information	Comment
status	M	ENUM (OperationSucceeded, OperationFailed)	An operation may fail because of a specified or unspecified reason.

7.5.1.4 Pre-condition

operationExits

Assertion Name	Definition
operationExits	The operation identified by the invokeIdentifierIn is ongoing.

7.5.1.5 Post-condition

operationCancelled

Assertion Name	Definition
operationCancelled	The operation identified by the invokeIdentifierIn is cancelled.

7.5.1.6 Exceptions

Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.

7.6 Interface ActiveCmIRPOperations

7.6.1 createMo (O)

7.6.1.1 Definition

This operation is invoked by IRPManager to request the IRPAgent to create a Managed Object instance in the MIB maintained by the IRPAgent. This operation will create only one Managed Object instance. This operation provides functionality that is similar to that provided by the M-CREATE service defined by CMIS (ITU-T X.710 [7]).

7.6.1.2 Input Parameters

Name	Qualifier	Information Type	Comment
managedObjectClass	M	ObjectClassIdentifier	This parameter specifies the class of the new managed object instance.
managedObjectInstance	M	DistinguishedName	This parameter specifies the instance of the managed object that is to be created and registered. This is a full Distinguished Name according to 3GPP TS 32.300 [13].
referenceObjectInstance	O	Solution Set dependant	This parameter may have a null value. When this parameter is supplied, it must specify an existing instance of a managed object, called the reference object, of the same class as the new object to be created. Attribute values associated with the reference object instance become the default values for those not specified by the attributeListIn parameter.
attributeListIn	M	LIST OF SEQUENCE< attribute name, attribute value>	This parameter may have a null value. When this parameter is supplied, it contains a list of name/value pairs specifying attribute identifiers and their values to be assigned to the new managed object. These values override the values for the corresponding attributes derived from either the reference object (if the referenceObjectInstance parameter is supplied) or the default value set specified in the definition of the managed object's class.

7.6.1.3 Output Parameters

Name	Qualifier	Matching Information	Comment
attributeListOut	M	LIST OF SEQUENCE< name OF ManagedEntity.anAttribute, value OF ManagedEntity.anAttribute >~~of ManageEntity.managedAttribute, ManagedEntity.managedAttribute>~~	This list of name/value pairs contains the attributes of the new managed object and the actual value assigned to each.
status	M	ENUM (OperationSucceeded, OperationFailed)	An operation may fail because of a specified or unspecified reason.

7.6.1.4 Pre-condition

managedEntityDoesNotExist

Assertion Name	Definition
managedEntityDoesNot Exist	The ManagedEntity instance is not being created with the same Distinguished Name as another already existing Managed Object instance.

7.6.1.5 Post-condition

managedEntityCreated

Assertion Name	Definition
managedEntityCreated	The ManagedEntity instance of the specified object class has been created with the specified Distinguished Name.

7.6.1.6 Exceptions

Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.
objectClassSpecificationMi ssmatched	**Condition:** The object class named by ObjectClassIdentifier input parameter does not match the object class of the managed object specified by a non-null referenceObjectInstance input parameter. **Returned Information:** The output parameter status. **Exit state:** Entry state.
InvalidObjectInstance	**Condition:** The object instance name specified implied a violation of the naming rules; **Returned Information:** The output parameter status. **Exit state:** Entry state.
createNotAllowed	**Condition:** The object to be created may not be created over the Itf-N. **Returned Information:** The output parameter status. **Exit state:** Entry state.
noSuchObjectClass	**Condition:** The class of the specified managed object is not recognized or. **Returned Information:** The output parameter status. **Exit state:** Entry state.
classInstanceConflict	**Condition:** The specified managed object instance may not be created as member of the specified class. **Returned Information:** The output parameter status. **Exit state:** Entry state.
noSuchAttribute	**Condition:** A specified attribute is not recognized or is not valid for specified object class. **Returned Information:** The output parameter status. **Exit state:** Entry state.
invalidAttributeValue	**Condition:** Value specified for an attribute is not valid for that attribute. **Returned Information:** The output parameter status. **Exit state:** Entry state.
missingAttributeValue	**Condition:** One or more required attribute values were not supplied and default values are not available. **Returned Information:** The output parameter status. **Exit state:** Entry state.

7.6.2 deleteMo (O)

7.6.2.1 Definition

This operation is invoked by IRPManager to request the deletion of one or more Managed Object instances from the MIB maintained by IRPAgent. This operation provides functionality that is similar to that provided by the M-DELETE service defined by CMIS (ITU-T X.710 [7]).

7.6.2.2 Input Parameters

Name	Qualifier	Information Type	Comment
baseObjectInst	M	DistinguishedNa	The MO instance that is to be used as the starting point for the

ance		me	selection of managed objects to which the filter (when supplied) is to be applied. This is a full Distinguished Name according to 3GPP TS 32.300 [13].
scope	M	See corresponding parameter in getMOAttributes.~~SEQUENCE < ENUM { BASE_OBJECT_ONLY, NTH_LEVEL_SUBORDINATES, BASE_NTH_LEVEL, BASE_ALL}, Integer>~~	See corresponding parameter in getMOAttributes.~~This parameter defines how many levels of the containment hierarchy to the filter defined below. The starts from the MO given by the baseObjectInstance parameter. The levels of that may be performed are:~~ ~~the base object alone (default);~~ ~~the n-th level subordinates of the base object;~~ ~~the base object and all of its subordinates down to and including the n-th level;~~ ~~the base object and all of its subordinates.~~
filter	M	See comment	See corresponding parameter in getMOAttributes.~~This parameter defines a filter test to be applied to the scoped Managed Object(s). If the filter is empty, all of the managed objects included by the scope are selected.~~ ~~The actual syntax and capabilities of the filter is Solution Set specific. However, each Solution Set should support a filter consisting of one or several assertions that may be grouped using the logical operators AND, OR and NOT. Each assertion is a logical expression of attribute existence, attribute value comparison ("equal to X, less than Y" etc.) and MO Class.~~

7.6.2.3 Output Parameters

Name	Qualifier	Matching Information	Comment
deletionList	M	LIST OF SEQUENCE< ManagedEntity.distinguishedName, ManagedEntity.objectClass >	If the base object alone is specified, then this parameter is optional; otherwise it contains a list of managedObjectInstance/managedObjectClass pairs identifying the managed objects deleted.
status	M	ENUM (OperationSucceeded, OperationFailed, OperationPartiallySucceeded)	An operation may fail because of a specified or unspecified reason. The operation is partially successful if some, but not all, objects selected to be deleted are actually deleted.

In lieu of a synchronization parameter, best effort synchronization will apply; that is, all managed objects selected for this operation will perform the operation if possible regardless of whether some managed objects fail to perform it.

7.6.2.4 Pre-condition

baseObjectExists AND allChildrenOfObjectsToBeDeletedSpecifiedForDeletion

Assertion Name	Definition
baseObjectExists	The ManagedEntity instance specified by the baseObjectInstance parameter exists.
allChildrenOfObjectsToBeDeletedSpecifiedForDeletion	For any ManagedEntity instance specified for deletion, all of its dependant ManagedEntity instances must be specified for deletion.

7.6.2.5 Post-condition

selectedObjectsDeleted OR someSelectedObjectsDeleted

Assertion Name	Definition
selectedObjectsDeleted	All of the ManagedEntity instances selected for deletion are deleted.
someSelectedObjectsDeleted	Some but not all of the selected ManagedEntity instance were deleted and for any of the ManagedEntity instances deleted all of the child ManagedEntity instances of that ManagedEntity instance is deleted.

7.6.2.6 Exceptions

Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.
invalidObjectInstance	**Condition:** The object instance name specified implied a violation of the naming rules; **Returned Information:** The output parameter status. **Exit state:** Entry state.
deleteNotAllowed	**Condition:** Some of the object instances to be deleted may not be deleted over the Itf-N. **Returned Information:** The output parameter status. **Exit state:** Entry state.
resourceLimitation	**Condition:** Operation not performed due to resource limitation. **Returned Information:** The output parameter status. **Exit state:** Entry state.
complexityLimitation	**Condition:** Operation not performed because a parameter was too complex. **Returned Information:** The output parameter status. **Exit state:** Entry state.

7.6.3 setMoAttributes (O)

7.6.3.1 Definition

This operation is invoked by IRPManager to request the modification of management information (Managed Object attribute values) in the MIB maintained by IRPAgent. Attributes of one or several Managed Objects may be modified - based on the containment hierarchy. This operation provides functionality that is similar to that provided by the M-SET service defined by CMIS (ITU-T X.710 [7]).

7.6.3.2 Input Parameters

Name	Qualifier	Information Type	Comment
baseObjectInstance	M	DistinguishedName	The MO instance that is to be used as the starting point for the selection of managed objects to which the filter (when supplied) is to be applied. This is a full Distinguished Name according to 3GPP TS 32.300 [13].
scope	M	See corresponding parameter in getMOAttributes. ~~SEQUENCE < ENUM { BASE_OBJECT_ONLY, NTH_LEVEL_SUBORDINATES, BASE_NTH_LEVEL, BASE_ALL}, Integer>~~	See corresponding parameter in getMOAttributes.~~This parameter defines how many levels of the containment hierarchy to search (i.e. apply the filter defined below). The search starts from the MO given by the baseObjectInstance parameter. The levels of search that may be performed are:~~ ~~the base object alone (default);~~ ~~the n-th level subordinates of the base object;~~ ~~the base object and all of its subordinates down to and including the n-th level;~~

			~~the base object and all of its subordinates.~~
filter	M	See comment	See corresponding parameter in getMOAttributes.~~This parameter defines a filter test to be applied to the scoped Managed Object(s). If the filter is empty, all of the managed objects included by the scope are selected.~~ ~~The actual syntax and capabilities of the filter is Solution Set specific. However, each Solution Set should support a filter consisting of one or several assertions that may be grouped using the logical operators AND, OR and NOT. Each assertion is a logical expression of attribute existence, attribute value comparison ("equal to X, less than Y" etc.) and MO Class.~~
modificationList	M	LIST OF SEQUENCE <attribute identifier, [attribute values], ENUM(replace, add values, remove values, set to default)> See Comment for when attribute values are require and when they are optional.	This parameter contains a set of attribute modification specifications, each of which contains: 1. attribute identifier: the identifier of the attribute whose value(s) is(are) to be modified. 2. attribute value: the value(s) to be used in the modification of the attribute. The use of this parameter is defined by the modify operator. This parameter is optional when the set to default modify operator is specified and if supplied, shall be ignored. 3. modify operator: the way in which the attribute values(s) (if supplied) is(are) to be applied to the attribute. The possible operators are: a. replace: the attribute value(s) specified shall be used to replace the current values(s) of the attribute; b. add values: the attribute values(s) specified shall be added to the current value(s) of the of the attribute. This operator shall only be applied to a set-valued attribute and shall perform a set union (in the mathematical sense) between the current values(s) of the attribute and the attribute value(s) specified. Value(s) specified in the attribute value parameter which is(are) already in the current values of the attribute shall not cause an error to be returned. c. remove values: the attribute value(s) specified shall be removed from the current values(s) of the attribute. This operator shall only be applied to a set-valued attribute and shall perform a set difference (in the mathematical sense) between the current value(s) of the attribute and the attribute values(s) specified. Value(s) specified in the attribute value parameter which is(are) not in the current value(s) of the attribute shall not cause an error to be returned; d. set to default: when this operator is applied to a single-valued attribute, the value of the attribute shall be set to its default value. When this operator is applied to a set–valued attribute, the value(s) of the attribute shall be set to their default value(s) and only as many values as defined by the

			default shall be assigned. If there is no default value defined, an error shall be returned. Note: Set is used here in the mathematical sense so that a set-valued attribute is an unordered set of unique values. The modify operator is optional, and if it is not specified, the replace operator shall be assumed. The modificationList parameter contains a single set of attribute modification specifications and this same set is applied to each managed object instance to be modified.

7.6.3.3 Output Parameters

Name	Qualifier	Matching Information	Comment
modificationListOut	M	LIST OF SEQUENCE< ManagedEntity.distinguishedName, ManagedEntity.objectClass, LIST OF SEQUENCE<name OF ManagedEntity.anAttribute, value OF ManagedEntity.anAttribute>>	This parameter will provide for each managed object instance the full Distinguished Name of the managed object instance, the managedObjectClass, and a list of name/value pairs with the values of all the attributes of the modified managed object instance after modification. The form of this information is solution set dependant and may be provided in one or many data structures.
status	M	ENUM (OperationSucceeded, OperationFailed, OperationPartiallySucceeded)	An operation may fail because of a specified or unspecified reason and no attributes have been updated. The operation is only successful if all specified attributes of all selected objects are actually modified. Otherwise, the operation is partially successful. ~~An operation may fail because of a specified or unspecified reason. The operation is partially successful if some attributes of some objects are modified. The operation is only successful if all specified attributes of all selected objects are actually modified.~~

In lieu of a synchronization parameter, best effort synchronization will apply; that is, all managed objects selected for this operation will perform the operation if possible regardless of whether some managed objects fail to perform it.

7.6.3.4 Pre-condition

baseObjectExists

Assertion Name	Definition
baseObjectExists	The ManagedEntity instance specified by the baseObjectInstance parameter exists.

7.6.3.5 Post-condition

selectedObjectsModified OR someSelectedObjectsModified

Assertion Name	Definition
selectedObjectsModified	All of the attributes of all of the ManagedEntity instances selected for modification are modified as specified.
someSelectedObjectsModified	Some attributes of some of the selected ManagedEntity instances were modified but not all attributes of all selected ManagedEntity instances.

7.6.3.6 Exceptions

Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned Information:** The output parameter status. **Exit state:** Entry state.
modifyNotAllowed	**Condition:** The object to be modified may not be modified over the Itf-N. **Returned Information:** The output parameter status. **Exit state:** Entry state.
noSuchObjectClass	**Condition:** The class of the specified managed object is not recognized or. **Returned Information:** The output parameter status. **Exit state:** Entry state.
classInstanceConflict	**Condition:** The specified managed object instance may not be created as member of the specified class. **Returned Information:** The output parameter status. **Exit state:** Entry state.
noSuchAttribute	**Condition:** A specified attribute is not recognized or is not valid for specified object class. **Returned Information:** The output parameter status. **Exit state:** Entry state.
invalidAttributeValue	**Condition:** Value specified for an attribute is not valid for that attribute. **Returned Information:** The output parameter status. **Exit state:** Entry state.
missingAttributeValue	**Condition:** One or more required attribute values were not supplied and default values are not available. **Returned Information:** The output parameter status. **Exit state:** Entry state.
resourceLimitation	**Condition:** Operation not performed due to resource limitation. **Returned Information:** The output parameter status. **Exit state:** Entry state.
complexityLimitation	**Condition:** Operation not performed because a parameter was too complex. **Returned Information:** The output parameter status. **Exit state:** Entry state.

Annex A (normative): Notification/Event Types

Notification IRP: Information Service [3] defines an attribute called `notificationType` that shall be present in all notifications. This document defines an attribute called `eventType` that shall be present in all CM notifications defined herein. The mapping of this `eventType` to the `notificationType` is that they are semantically equal for the CM notifications. Thus, the event types described below (also the same as in Release 99) shall be mapped to the `notificationType` of the notification header.

This annex lists and explains Event Types used by Basic CM IRP and then lists the Event Types valid for each notification in this IRP.

Encoding of `eventType` is Solution Set dependent. For example, the value of `eventType` may be encoded as an Object Identifier in the CMIP SS and as a numeric string in the CORBA SS.

The tables below may be extended in the future.

Table A.1: Event Types

Event Types	Explanation
Object creation	A notification of this type indicates that a new managed object instance has been created (as defined in ITU-T X.721 [8] and ITU-T X.730 [9]).
Object deletion	A notification of this type indicates that a managed object instance has been deleted (as defined in ITU-T X.721 [8] and ITU-T X.730 [9]).
Attribute value change	A notification of this type indicates that the value(s) of one or more attributes have changed (as defined in ITU-T X.721 [8] and ITU-T X.730 [9]).

Table A.2: Event types applicable to each Notification

Notification	Event Type
notifyObjectCreation	Object creation
notifyObjectDeletion	Object deletion
notifyAttributeValueChange	Attribute value change

CR-Form-v7

CHANGE REQUEST

⌘ **32.603** **CR** **006** ⌘ **rev** **-** ⌘ Current version: **4.3.1** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network **X** Core Network **X**

Title:	⌘	Add Active Basic CM feature - CORBA Solution Set
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**B**
Release:	⌘	Rel-5

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Introduction of active Basic CM feature in Rel-5 version of Basic CM IRP
Summary of change:	⌘	• Removal of CM notifications (transferred to Rel-5 new Kernel CM IRP) • Addition of active Basic CM feature • Addition of reference to the related IS TS • Editorial modifications
Consequences if not approved:	⌘	
Clauses affected:	⌘	Introduction, Scope, References, 4, 5.1 (void), 6.1, 6.2, 6.3, 6.4 (void), 7 (void), 8.1, annex A, annex B (void)

	Y	N		
Other specs affected: ⌘		X	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	

Other comments: ⌘

S5-026678	Rel-5 CR 32600 Add new Rel-5 functionality (Kernel CM, Active CM) - Parent CR
S5-026679	Rel-5 CR 32601 Basic CM IRP Requirements - Child CR
S5-026680	Rel-5 CR 32602 Basic CM IRP IS - Grandchild CR
S5-026653	Rel-5 CR 32603 Basic CM IRP IS - Grandchild CR

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

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

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service (QOS). The CM actions are initiated either as single actions on single NEs of the 3G network, or as part of a complex procedure involving actions on many resources/objects in one or several NEs.

~~Due to the growing number of specifications to model new services and Resource Models for Configuration Management (CM), as well as the expected growth in size of each of them from 3GPP Release 4 onwards, a new structure of the specifications is already needed in Release 4. This structure is needed for several reasons, but mainly to enable more independent development and release for each part, as well as a simpler document identification and version handling. Another benefit would be that it becomes easier for bodies outside 3GPP, such as the ITU-T, to refer to telecom management specifications from 3GPP. The new structure of the specifications does not lose any information or functionality supported by the Release 1999.~~

~~In addition to the restructuring, the need to define some new IRPs for CM, compared to Release 1999, has also been identified. Firstly, a new IRP for the Bulk CM, and secondly, one for each of the NRM parts (Generic, Core Network, UTRAN and GERAN NRM).~~

~~Finally, the Notification IRP (in Release 1999: 32.106-1 to -4) and the Name convention for Managed Objects (in Release 1999: 32.106-8) have been moved to a separate number series used for specifications common between several management areas (e.g. CM, FM, PM).~~

1 Scope

The purpose of this *Basic Configuration Management (CM) IRP: CORBA Solution Set* is to define the mapping of the Basic CM IRP: IS (see 3GPP TS 32.602 [4]) to the protocol specific details necessary for implementation of this IRP in a CORBA/IDL environment.

This document defines NRM independent data types~~,~~ and methods ~~and notifications~~.

This Solution Set specification is related to 3G TS 32.602 V5.0.X.

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: "3G Telecom Management principles and high level requirements".

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

[3] 3GPP TS 32.600: “3G Configuration Management: Concept and High-level Requirements”.

[4] 3GPP TS 32.602: “Basic Configuration Management IRP: Information Service”.

[5] 3GPP TS 32.300: “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 32.303: “Notification IRP: CORBA Solution Set”.

[10] Void.~~3GPP TS 32.111-3: “Alarm IRP: CORBA Solution Set”.~~

[11] 3GPP TS 32.312: "Generic IRP Management: Information Service".

[12] 3GPP TS 32.663: "Kernel CM IRP: CORBA Solution Set".

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 32.600 [3] and 3GPP TS 32.602 [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 document version number string

The IRP document version number (sometimes called “IRPVersion” or “SS version number”) string is used to identify this specification. The string is derived using a rule described in 3GPP TS 32.312: "Generic IRP Management: Information Service" [11]. ~~The value of this string is defined by a constant in Annex A.~~

This string (or sequence of strings, if more than one version is supported) is returned in `getBasicCmIRPVersion` method ~~and is carried in the first field of the notification header of all notifications related to this IRP~~.

5 Architectural features

The overall architectural feature of Basic Configuration Management IRP is specified in 3GPP TS 32.602 [4]. This clause specifies features that are specific to the CORBA SS.

5.1 Notifications

Void.

~~Notifications are sent according to the Notification IRP: CORBA SS (see 3GPP TS 32.303 [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 32.303 [9]).

5.3 Syntax for Distinguished Names and Versions

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

The version of this IRP is represented as a string (see also clause 4).

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 (see [12]) is emitted.

6.2 Operation ~~and Notification~~ mapping

The Basic CM IRP: IM (see 3GPP TS 32.602 [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 32.602 [4])	SS Method	Qualifier
getMoAttributes	BasicCmIrpOperations::find_managed_objects BasicCmInformationIterator::next_basicCmInformations	M
getContainment	BasicCmIrpOperations::find_managed_objects BasicCmInformationIterator::next_basicCmInformations	O
getBasicCmIRPVersion	get_basicCm_IRP_version	M
cancelOperation	BasicCmInformationIterator::destroy	O
~~notifyObjectCreation (to convey of a new Managed Object created)~~	~~See Notification IRP: CORBA SS [9]~~	~~O~~
~~notifyObjectDeletion (to convey of a 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~~
createMo	BasicCmIrpOperations::create_managed_object	O
deleteMo	BasicCmIrpOperations::delete_managed_objects	O
setMoAttributes	BasicCmIrpOperations::modify_managed_objects	O

6.3 Operation parameter mapping

The Basic CM IRP: IS (see 3GPP TS 32.602 [4]) defines semantics of parameters carried in operations across the Basic Configuration Management IRP. Tables 2~~, 3 and 4~~ through 8 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 `BasicCmInformationIterator` is used to fetch the result.

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

IS Operation parameter	SS Method parameter	Qualifier
~~invokeIdentifierIn~~invokeIdentifier	- (~~no~~ No equivalence)	-
invokeIdentifierOut	~~The iterator returned from the call (~~Return value of type BasicCmInformationIterator~~) identifies the request.~~	M
baseObjectInstance	in DN baseObject	M
~~Scope~~scope	in SearchControl searchControl (SearchControl.~~scope~~type and SearchControl.level)	M
~~Filter~~filter	in SearchControl searchControl (SearchControl.filter)	M
attributeListIn	in AttributeNameSet requestedAttributes	M
managedObjectClass managedObjectInstance attributeListOut	Return value of type BasicCmInformationIterator - parameter out ResultSet fetchedElements ~~in the~~ of method next_basicCmInformations ~~in the BasicCmInformationIterator interface.~~	M

IS Operation parameter	SS Method parameter	Qualifier
~~Status~~status	Exceptions: FindManagedObjects, ManagedGenericIRPSystem::InvalidParameter, ~~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 `BasicCmInformationIterator` is used to fetch the result.

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

IS Operation parameter	SS Method parameter	Qualifier
~~invokeIdentifierIn~~invokeIdentifier	- (~~no~~ No equivalence)	-
invokeIdentifierOut	~~The iterator returned from the call (~~Return value of type BasicCmInformationIterator~~) identifies the request.~~	M
baseObjectInstance	in DN baseObject	M
~~Scope~~scope	in SearchControl searchControl (SearchControl.~~scope~~type and SearchControl.level)	O
~~Not specified in IS~~Not specified in IS	in SearchControl searchControl (SearchControl.filter)	M
~~Containment~~containment	Return value of type BasicCmInformationIterator - parameter out ResultSet fetchedElements ~~in the~~ of method next_basicCmInformations ~~in the BasicCmInformationIterator interface.~~	M
~~Status~~status	Exceptions: FindManagedObjects, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ~~exception~~ ManagedGenericIRPSystem::ValueNotSupported, ~~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
versionNumberSet	Return value of type ManagedGenericIRPConstDefs::VersionNumberSet	M
status	Exceptions: GetBasicCmIRPVersion	M

Table 5: Mapping from IS `cancelOperation` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
invokeIdentifier	- (Not applicable, the BasicCmInformationIterator instance identifies the ongoing operation)~~.~~	M
status	~~- (No failure conditions identified)~~Exceptions: DestroyException	M

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

IS Operation parameter	SS Method parameter	Qualifier
managedObjectClass managedObjectInstance	in DN objectName	M
referenceObjectInstance	in DN referenceObject	O
attributeListIn attributeListOut	inout MoAttributeSet attributes	M
status	out AttributeErrorSeq attributeErrors Exceptions: CreateManagedObject, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, UndefinedMOException, IllegalDNFormatException, DuplicateMO, CreateNotAllowed, ObjectClassMismatch, NoSuchObjectClass	M

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

IS Operation parameter	SS Method parameter	Qualifier
baseObjectInstance	in DN baseObject	M
scope	in SearchControl searchControl (SearchControl.type and SearchControl.level)	M
filter	in SearchControl searchControl (SearchControl.filter)	M
deletionList	Return value of type DeleteResultIterator - parameter out ResultSet fetchedElements of method next_basicCmInformations	M
status	Return value of type DeleteResultIterator - parameter out DeleteErrorSeq fetchedDeleteErrors of method next_deleteErrors Exceptions: DeleteManagedObjects, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter, UndefinedMoException, IllegalDNFormatException, UndefinedScopeException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalFilterFormatException, FilterComplexityLimit	M

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

IS Operation parameter	SS Method parameter	Qualifier
baseObjectInstance	in DN baseObject	M
scope	in SearchControl searchControl (SearchControl.type and SearchControl.level)	M
filter	in SearchControl searchControl (SearchControl.filter)	M
modificationList	in AttributeModificationSet modifications	M
modificationListOut	Return value of type ModifyResultIterator - parameter out ResultSet fetchedElements of method next_basicCmInformations	M

IS Operation parameter	SS Method parameter	Qualifier
status	Return value of type ModifyResultIterator - parameter out ModifyAttributeErrorsSeq fetchedModifyErrors of method next_modifyErrors Exceptions: ModifyManagedObjects, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter, UndefinedMoException, IllegalDNFormatException, UndefinedScopeException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalFilterFormatException, FilterComplexityLimit	M

6.4 Notification attribute mapping

Void.

~~The Basic CM IRP: IS (see 3GPP TS 32.602 [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 6: Mapping from IS notifications to SS equivalents~~

~~IS notifications in 3GPP TS 32.602 [4]~~	~~SS notifications~~	~~Qualifier~~
~~NotifyObjectCreation~~	~~push_structured_event~~	~~O~~
~~NotifyObjectDeletion~~	~~push_structured_event~~	~~O~~
~~NotifyAttributeValue Change~~	~~push_structured_event~~	~~O~~

~~The Basic CM IRP: IS (see 3GPP TS 32.602 [4]) also qualifies the attributes. Tables 6, 7, 8 and 9 show the mapping of these IS attributes to SS equivalents.~~

~~Table 7: Mapping from IS `Notification Header` attributes to SS equivalent~~

~~IS Attribute of `Notification Header` in 3GPP TS 32.602 [4]~~	~~SS Attribute~~	~~Qualifier~~
~~managedObjectClass~~	~~BasicCmNotifDefs::NotificationCommon::MANAGED_OBJECTCLASS~~	~~M~~
~~managedObjectInstance~~	~~BasicCmNotifDefs::NotificationCommon::MANAGED_OBJECT_INSTANCE~~	~~M~~
~~notificationId~~	~~BasicCmNotifDefs::NotificationCommon::NOTIFICATION_ID~~	~~O~~
~~eventTime~~	~~BasicCmNotifDefs::NotificationCommon::EVENT_TIME~~	~~M~~
~~systemDN~~	~~BasicCmNotifDefs::NotificationCommon::SYSTEM_DN~~	~~O~~
~~eventType~~	~~header.fixed_header.event_type.type_name~~	~~M~~

~~Table 8: Mapping from IS `notifyObjectCreation` attributes to SS equivalent OBJECT_CREATION~~

~~IS Attribute of `notifyObjectCreation` in 3GPP TS 32.602 [4]~~	~~SS Attribute~~	~~Qualifier~~
~~notificationHeader~~	~~See Table 7~~	~~M~~
~~correlatedNotifications~~	~~BasicCmNotifDefs::MOCreation::CORRELATED_NOTIFICATIONS~~	~~O~~
~~additionalText~~	~~BasicCmNotifDefs::MOCreation::ADDITIONAL_TEXT~~	~~O~~
~~sourceIndicator~~	~~BasicCmNotifDefs::MOCreation::SOURCE_INDICATOR~~	~~O~~
~~attributeList~~	~~remainder_of_body~~	~~O~~

Table 9: Mapping from IS `notifyObjectDeletion` attributes to SS equivalent OBJECT_DELETION

IS Attribute of `notifyObjectDeletion` in 3GPP TS 32.602 [4]	SS Attribute	Qualifier
notificationHeader	See Table 7	M
correlatedNotifications	BasicCmNotifDefs::MODeletion::CORRELATED_NOTIFICATIONS	O
additionalText	BasicCmNotifDefs::MODeletion::ADDITIONAL_TEXT	O
sourceIndicator	BasicCmNotifDefs::MODeletion::SOURCE_INDICATOR	O
attributeList	remainder_of_body (a field of the StructuredEvent)	O

Table 10: Mapping from IS `notifyAttributeValueChange` attributes to SS equivalent ATTRIBUTE_VALUE_CHANGE

IS Attribute of `notifyAttributeValueChange` in 3GPP TS 32.602 [4]	SS Attribute	Qualifier
notificationHeader	See Table 7	M
correlatedNotifications	BasicCmNotifDefs::AttributeValueChange::CORRELATED_NOTIFICATIONS	O
additionalText	BasicCmNotifDefs::AttributeValueChange::ADDITIONAL_TEXT	M
sourceIndicator	BasicCmNotifDefs::AttributeValueChange::SOURCE_INDICATOR	O
attributeValueChangeDefinition	remainder_of_body	M

7 Use of OMG Structured Event

Void.

In CORBA SS, OMG defined `StructuredEvent` (see OMG Notification Service [6]) is used to carry notifications. This clause identifies the OMG defined `StructuredEvent` attributes that carry the attributes of notifications defined in 3GPP TS 32.602 [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 11: Use of OMG Structured Event

SS Attribute	OMG CORBA Structured Event attribute	Comment
There is no corresponding SS attribute	domain_name	It contains the supported SS document version (see clause 4). This version is defined by the string constant BasicCmIRPSystem::VERSION defined in this specification.
Event Type	type_name	It is an attribute of notificationHeader. It shall indicate one of the following ITU-T defined semantics: Object Creation, Object Deletion and Attribute Value Change. It is a string. Its value is either defined by BasicCmNotifDefs::MOCreation::EVENT_TYPE, BasicCmNotifDefs::MODeletion::EVENT_TYPE or BasicCmNotifDefs::AttributeValueChange::EVENT_TYPE
-	event_name	Shall be set to an empty string
There is no corresponding SS attribute	variable Header	
Managed Object Class, Managed Object Instance	One NV pair of filterable_ body_fields	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. They are attributes of notificationHeader. Name of NV pair is a string, BasicCmNotifDefs::<interface>::MANAGED_OBJECT_INSTANCE where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a string. This string conveys the semantics of both the Managed Object Class and the Managed Object Instance. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.303 [9]).
Notification Id	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is a string, BasicCmNotifDefs::<interface>::NOTIFICATION_ID where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a long. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.303 [9]).
Event Time	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is a string, BasicCmNotifDefs::<interface>::EVENT_TIME where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a ManagedGenericIRPConstDefs::IRPTime defined in 3GPP TS 32.303 [9]. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.303 [9]).
System DN	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is a string, BasicCmNotifDefs::<interface>::SYSTEM_DN where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a string. See corresponding table in Notification IRP: CORBA SS [9].
Correlated Notifications	One NV pair of filterable_ body_fields	It is an attribute of the Object Creation, Object Deletion and Attribute Value Change notifications. Name of NV pair is a string, BasicCmNotifDefs::<interface>::CORRELATED_NOTIFICATIONS where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a NotificationIRPConstDefs::CorrelatedNotificationSetType defined in 3GPP TS 32.303 [9].
Additional Text	One NV pair of filterable_ body_fields	It is an attribute of the Object Creation, Object Deletion and Attribute Value Change notifications. Name of NV pair is a string, BasicCmNotifDefs::<interface>::ADDITIONAL_TEXT where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a string.
Source Indicator	One NV pair of filterable_ body_fields	It is an attribute of the Object Creation, Object Deletion and Attribute Value Change notifications. Name of NV pair is a string, BasicCmNotifDefs::<interface>::SOURCE_INDICATOR where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a string with values of either BasicCmNotifDefs::<interface>::RESOURCE_OPERATION, BasicCmNotifDefs::<interface>::MANAGEMENT_OPERATION or BasicCmNotifDefs::<interface>::UNKNOWN_OPERATION where <interface> is either MODeletion, MOCreation or AttributeValueChange.

~~SS Attribute~~	~~OMG CORBA Structured Event attribute~~	~~Comment~~
~~There is no corresponding SS attribute~~		~~Is used to transport attribute information. For Object Creation notification, this is defined by BasicCmNotifDefs::MOCreation::InitialAttributeValues. For Object Deletion notification, this is defined by BasicCmNotifDefs::MODeletion::AttributeValues. For Attribute Value Change notification, this is defined by BasicCmNotifDefs::AttributeValueChange::ModifiedAttributeSet. The name component of InitialAttributeValues, AttributeValues and ModifiedAttributeSet will be set to attribute names defined in BasicCmNRMDefs.~~

8 Rules for NRM extensions

This clause discusses how the models and IDL definitions provided in 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 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.

NRM 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 NRM 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 `SubNetwork` (or its subclasses). Also, in Rel-4, there may only be 0 or 1 `ManagementNode` (or its subclasses) contained under `SubNetwork` (or its subclasses).

Managed Object Instances may be instantiated as CORBA objects. This requires that the MOCs be represented in IDL. 3GPP SA5's NRM 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.~~

8.2 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.

Annex A (normative): CORBA IDL, Access Protocol

```
#ifndef BasicCmIRPSystem_idl
#define BasicCmIRPSystem_idl

#include "ManagedGenericIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"

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

module BasicCmIRPSystem
{

   /**
    *  This constant defines the version of this IRP.
    */
   const string VERSION = "32.601-3 V4.0";

   /**
    * Defines the name of a Managed Object Class
    */
   typedef string MOClass;


   /**
    * The format of Distinguished Name (DN) is specified in 3GPP TS 32.300
    * "Name Conventions
    * for Managed Objects revision B".
    */
   typedef string DN;


   /**
    * Defines the name of an attribute of a Managed Object
    */
   typedef string MOAttributeName;


   /**
    * Defines the value of an attribute of a Managed Object in form of a CORBA
    * Any. Apart from basic datatypes already defined in CORBA, the allowed
    * attribute value types are defined in the AttributeTypes module.
    */
   typedef any MOAttributeValue;


   /**
    *  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;
   };

   exception NextBasicCmInformations {
      string reason;
   };

   exception InvalidParameter {
      string parameter;
   };

   exception GetBasicCmIRPVersion {
      string reason;
   };

   exception DuplicateMO {};

   exception CreateNotAllowed {};

   exception ObjectClassMismatch {};

   exception NoSuchObjectClass {
      MOClass objectClass;
   };
```

```
    /**
     * System otherwise fails to complete the operation.  System can provide
     * reason to qualify the exception. The semantics carried in reason
     * is outside the scope of this IRP.
     */
    exception NextBasicCmInformations { string reason; };
    exception NextDeleteErrors { string reason; };
    exception NextModifyErrors { string reason; };
    exception DestroyException { string reason; };
    exception GetBasicCmIRPVersion { string reason; };
    exception FindManagedObjects { string reason; };
    exception CreateManagedObject { string reason; };
    exception DeleteManagedObjects { string reason; };
    exception ModifyManagedObjects { string reason; };


    /**
     *
     * In this version 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
   {
      stringMOAttributeName name;
      anyMOAttributeValue value;
   };

   typedef sequence<MOAttribute> MOAttributeSet;



    struct Result
    {
       DN mo;
       MOAttributeSet attributes;
    };

    typedef sequence<Result> ResultSet;


    /**
     * AttributeErrorCategory defines the categories of errors, related to
     * attributes, that can occur during creation or modification of MOs.
     *
     * NO_SUCH_ATTRIBUTE: The specified attribute does not exist.
     * INVALID_ATTRIBUTE_VALUE: The specified attribute value is not valid.
     * MISSING_ATTRIBUTE_VALUE: An attribute value is required but none was
     *   provided and no default value is defined for the attribute.
     * INVALID_MODIFY_OPERATOR: The specified modify operator is not valid
     *   (e.g. operator ADD_VALUES applied to a non multi-valued attribute
     *   or operator SET_TO_DEFAULT applied where no default value is defined).
     * MODIFY_NOT_ALLOWED: The modification of the attribute is not allowed.
     * MODIFY_FAILED: The modification failed because of an unspecified reason.
     */
    enum AttributeErrorCategory
```

```
    {
        NO_SUCH_ATTRIBUTE,
        INVALID_ATTRIBUTE_VALUE,
        MISSING_ATTRIBUTE_VALUE,
        INVALID_MODIFY_OPERATOR,
        MODIFY_NOT_ALLOWED,
        MODIFY_FAILED
    };


    /**
     * DeleteErrorCategory defines the categories of errors that can occur
     * during deletion of MOs.
     *
     * SUBORDINATE_OBJECT: The MO cannot be deleted due to subordinate MOs.
     * DELETE_NOT_ALLOWED: The deletion of the MO is not allowed.
     * DELETE_FAILED: The deletion failed because of an unspecified reason.
     */
    enum DeleteErrorCategory
    {
        SUBORDINATE_OBJECT,
        DELETE_NOT_ALLOWED,
        DELETE_FAILED
    };


    /**
     * AttributeError represents an error, related to an attribute, that occured
     * during creation or modification of MOs.
     * It contains:
     * - the name of the indicted attribute ("name" field),
     * - the category of the error ("error" field),
     * - optionally, the indicted attribute value ("value" field),
     * - optionally, additional details on the error ("reason" field).
     */
    struct AttributeError
    {
        MOAttributeName name;
        AttributeErrorCategory error;
        MOAttributeValue value;
        string reason;
    };

    typedef sequence<AttributeError> AttributeErrorSeq;


    /**
     * DeleteError represents an error that occured during deletion of MOs.
     * It contains:
     * - the distinguished name of the indicted MO ("object" field),
     * - the category of the error ("error" field),
     * - optionally, additional details on the error ("reason" field).
     */
    struct DeleteError
    {
        DN object;
        DeleteErrorCategory error;
        string reason;
    };

    typedef sequence<DeleteError> DeleteErrorSeq;
```

```
    /**
     * ModifyAttributeErrors represents errors that occured during
     * modification of attributes of a MO.
     * It contains:
     * - the distinguished name of the indicted MO ("object" field),
     * - a sequence containing the attribute errors ("errors" field).
     */
    struct ModifyAttributeErrors
    {
       DN object;
       AttributeErrorSeq errors;
    };

    typedef sequence<ModifyAttributeErrors> ModifyAttributeErrorsSeq;


    /**
     The BasicCmInformationIterator is used to iterate through a snapshot of
      Managed Object Information when IRPManager invokes find_managed_objects.
      IRPManager uses it to pace the return of Managed Object Information.

     IRPAgent controls the life-cycle of the iterator. However, a destroy
      operation is provided to handle the case where IRPManager wants to stop
      the iteration procedure before reaching the last iteration.
     */
    interface BasicCmInformationIterator
    {

       /**
        This method returns between 1 and "how_many" Managed Object information.
         The IRPAgent may return less than "how_many" items even if there are
         more items to return. "how_many" must be non-zero. Return TRUE if there
         may be more Managed Object information to return. Return FALSE if there
         are no more Managed Object information to be returned.

         If FALSE is returned, the IRPAgent will automatically destroy the
          iterator.

         @parm how_many 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 BasicCmInformationIterator is
          empty.
        */

        boolean next_basicCmInformations (
          in unsigned short how_many,
          out ResultSet fetchedElements
        )
        raises (NextBasicCmInformations,
                ManagedGenericIRPSystem::InvalidParameter);

        /**
        This method destroys the iterator.
        */

        void destroy ()
        raises (DestroyException);

    }; // end of BasicCmInformationIterator
```

```
    /**
     The DeleteResultIterator is used to iterate through the list of deleted MOs
      when IRPManager invokes method "delete_managed_objects".
      IRPManager uses it to pace the return of Managed Object Information.

     IRPAgent controls the life-cycle of the iterator. However, a destroy
      operation is provided to handle the case where IRPManager wants to stop
      the iteration procedure before reaching the last iteration.
     */
    interface DeleteResultIterator : BasicCmInformationIterator
    {

       /**
        Inherited method "next_basicCmInformations" has the same behaviour as
         for interface BasicCmInformationIterator, except that:
         - The Managed Object information returned in parameter
           "fetchedElements" contains only the DNs of the deleted MOs
           (no attributes are returned).
         - If FALSE is returned, the IRPAgent will not automatically destroy the
           iterator.
        */

       /**
        This method returns between 0 and "how_many" deletion errors. The
         IRPAgent may return less than "how_many" items even if there are more
         items to return. "how_many" must be non-zero. Return TRUE if there are
         more deletion errors to return. Return FALSE if there are no more
         deletion errors to be returned.

         If FALSE is returned and last call to inherited method
         "next_basicCmInformations" also returned FALSE (i.e. no more Managed
         Object information to be returned), the IRPAgent will automatically
         destroy the iterator.

         @parm how_many: how many deletion errors to return in the
          "fetchedDeleteErrors" out parameter.
         @parm fetchedDeleteErrors: the deletion errors.
         @returns: a boolean indicating if any deletion errors are returned.
        */

        boolean next_deleteErrors (
          in unsigned short how_many,
          out DeleteErrorSeq fetchedDeleteErrors
        )
        raises (NextDeleteErrors,
                ManagedGenericIRPSystem::InvalidParameter);

    }; // end of DeleteResultIterator


    /**
     The ModifyResultIterator is used to iterate through the list of modified
      MOs when IRPManager invokes method "modify_managed_objects".
      IRPManager uses it to pace the return of Managed Object Information.

     IRPAgent controls the life-cycle of the iterator. However, a destroy
      operation is provided to handle the case where IRPManager wants to stop
      the iteration procedure before reaching the last iteration.
     */
    interface ModifyResultIterator : BasicCmInformationIterator
    {

       /**
```

```
        Inherited method "next_basicCmInformations" has the same behaviour as
         for interface BasicCmInformationIterator, except that:
         - The Managed Object information returned in parameter
           "fetchedElements" contains DNs and attributes of the modified MOs.
         - If FALSE is returned, the IRPAgent will not automatically destroy the
           iterator.
        */

       /**
        This method returns between 0 and "how_many" modification errors. The
         IRPAgent may return less than "how_many" items even if there are more
         items to return. "how_many" must be non-zero. Return TRUE if there are
         more modification errors to return. Return FALSE if there are no more
         modification errors to be returned.

         If FALSE is returned and last call to inherited method
         "next_basicCmInformations" also returned FALSE (i.e. no more Managed
         Object information to be returned), the IRPAgent will automatically
         destroy the iterator.

         @parm how_many: how many modification errors to return in the
          "fetchedModifyErrors" out parameter.
         @parm fetchedModifyErrors: the modification errors.
         @returns: a boolean indicating if any modification errors are returned.
        */

        boolean next_modificationErrors (
          in unsigned short how_many,
          out ModifyAttributeErrorsSeq fetchedModifyErrors
        )
        raises (NextModifyErrors,
                ManagedGenericIRPSystem::InvalidParameter);

    }; // end of ModifyResultIterator


   typedef sequence<string MOAttributeName> AttributeNameSet;


    /**
     * ModifyOperator defines the way in which an attribute value is to be
     * applied to an attribute in a modification of MO attributes.
     *
     * REPLACE: replace the current value with the provide value
     * ADD_VALUES: for a multi-valued attribute, add the provided values to the
     *   current list of values
     * REMOVE_VALUES: for a multi-valued attribute, remove the provided values
     *   from the current list of values
     * SET_TO_DEFAULT: set the attribute to its default value
     */
    enum ModifyOperator
    {
        REPLACE,
        ADD_VALUES,
        REMOVE_VALUES,
        SET_TO_DEFAULT
    };


    /**
     * AttributeModification defines an attribute value and the way it is to
     * be applied to an attribute in a modification of MO attributes.
     * It contains:
```

```
     * - the name of the attribute to modify ("name" field),
     * - the value to apply to this attribute ("value" field),
     * - the way the attribute value is to be applied to the attribute
     *   ("operator" field).
    struct AttributeModification
    {
       MOAttributeName name;
       MOAttributeValue value;
       ModifyOperator operator;
    };

    typedef sequence<AttributeModification> AttributeModificationSet;
```

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

    /**
     * Get the version(s) of the interface
     *
     * @raises GetBasicCmIRPVersion when the system for some reason
     *   can not return the supported versions.
     * @returns all supported versions.
     */
    ManagedGenericIRPConstDefs::VersionNumberSet get_basicCm_IRP_version()
       raises (GetBasicCmIRPVersion);


    /**
     * 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 BasicCmInformationIterator 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 BasicCmInformationIterator will be returned.
     * Executing the next_basicCmInformations in the
     * BasicCmInformationIterator 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. In this version 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 ManagedGenericIRPSystem::ValueNotSupported if a valid but
     * unsupported parameter value is passed. E.g. the contents
     * field in the searchcontrol parameter contains the value NAMES and
     * the optional getContainment IS operation is not supported.
```

```
         * @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 BasicCmInformationIterator
         */
        BasicCmInformationIterator find_managed_objects(in DN baseObject,
                                        in SearchControl searchControl,
                                        in AttributeNameSet requestedAttributes)
           raises (FindManagedObjects,
                   ManagedGenericIRPSystem::ParameterNotSupported,
                   ManagedGenericIRPSystem::InvalidParameter,
                   ManagedGenericIRPSystem::ValueNotSupported,
                   UndefinedMOException,
                   IllegalDNFormatException,
                   UndefinedScopeException,
                   IllegalScopeTypeException,
                   IllegalScopeLevelException,
                   IllegalFilterFormatException,
                   FilterComplexityLimit);

        /**
         * Performs the creation of a MO instance in the MIB maintained
         * by the IRPAgent.
         *
         * @parm objectName: the distinguished name of the MO to create.
         * @parm referenceObject: the distinguished name of a reference MO.
         * @parm attributes: in input, initial attribute values for the MO to
         *   create; in output, actual attribute values of the created MO.
         * @parm attributeErrors: errors, related to attributes, that caused the
         *   creation of the MO to fail.
         *
         * @raises ManagedGenericIRPSystem::OperationNotSupported: The operation
         *   is not supported.
         * @raises ManagedGenericIRPSystem::ParameterNotSupported: An optional
         *   parameter is not supported.
         * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid
         *   parameter value has been provided.
         * @raises UndefinedMOException: The MO does not exist.
         * @raises IllegalDNFormatException: The DN syntax string is malformed.
         * @raises DuplicateMO: A MO already exist with the same DN as the one
         *   to create.
         * @raises CreateNotAllowed: The creation of the MO is not allowed.
         * @raises ObjectClassMismatch: The object class of the MO to create does
         *   not match with the object class of the provided reference MO.
         * @raises NoSuchObjectClass: The class of the object to create is not
         *   recognized.
         */
        void create_managed_object (
            in DN objectName,
            in DN referenceObject,
            inout MoAttributeSet attributes,
            out AttributeErrorSeq attributeErrors
        )
        raises (CreateManagedObject,
```

```
                ManagedGenericIRPSystem::OperationNotSupported,
                ManagedGenericIRPSystem::ParameterNotSupported,
                ManagedGenericIRPSystem::InvalidParameter,
                UndefinedMOException,
                IllegalDNFormatException,
                DuplicateMO,
                CreateNotAllowed,
                ObjectClassMismatch,
                NoSuchObjectClass);

        /**
         * Performs the deletion of one or more MO instances from the MIB
         * maintained by the IRPAgent, using a SearchControl to control the
         * instances to be deleted.
         *
         * All MOs in the scope constitute a set that the filter works on.
         * All matched MOs will be deleted by this operation.
         * The returned DeleteResultIterator is used to retrieve the DNs of the
         * MOs deleted and the errors that may have occurred preventing deletion
         * of some MOs.
         * For the special case when no managed objects are matched in
         * delete_managed_objects, the DeleteResultIterator will be returned.
         * Executing the next_basicCmInformations in the DeleteResultIterator
         * will return FALSE for completion.
         *
         * @parm baseObject: the start MO in the containment tree.
         * @parm searchControl: the SearchControl to use; field "contents" has no
         *   meaning here and shall be ignored.
          @returns: a DeleteResultIterator (see above).
         *
         * @raises ManagedGenericIRPSystem::OperationNotSupported: The operation
         *   is not supported.
         * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid
         *   parameter value has been provided.
         * @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: The filter syntax is correct,
         *   but the filter is too complex to be processed by the IRPAgent.
         */
        DeleteResultIterator delete_managed_objects (
           in DN baseObject,
           in SearchControl searchControl
        )
        raises (DeleteManagedObjects,
                ManagedGenericIRPSystem::OperationNotSupported,
                ManagedGenericIRPSystem::InvalidParameter,
                UndefinedMOException,
                IllegalDNFormatException,
                UndefinedScopeException,
                IllegalScopeTypeException,
                IllegalScopeLevelException,
                IllegalFilterFormatException,
                FilterComplexityLimit);

        /**
         * Performs the modification of MO attributes. One or more MOs attributes
         * may be modified according to a SearchControl.
         *
         * All MOs in the scope constitute a set that the filter works on.
```

```
        * All matched MOs will have their attributes modified by this operation.
        * The returned ModifyResultIterator is used to retrieve the DNs of the
        * modified MOs together with the values of the modified attributes, and
        * the errors that may have occurred preventing modification of some
        * attributes.
        * For the special case when no managed objects are matched in
        * modify_managed_objects, the ModifyResultIterator will be returned.
        * Executing the next_basicCmInformations in the ModifyResultIterator
        * will return FALSE for completion.
        *
        * @parm baseObject: the start MO in the containment tree.
        * @parm searchControl: the SearchControl to use; field "contents" has no
            meaning here and shall be ignored.
        * @parm modifications: the values for the attributes to modify and
            the way those values are to be applied to the attributes.
         @returns: a ModifyResultIterator (see above).
        *
        * @raises ManagedGenericIRPSystem::OperationNotSupported: The operation
        *   is not supported
        * @raises ManagedGenericIRPSystem::InvalidParameter: An invalid
        *   parameter value has been provided
        * @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: The filter syntax is correct,
        *   but the filter is too complex to be processed by the IRPAgent.
        */
       ModifyResultIterator modify_managed_objects (
          in DN baseObject,
          in SearchControl searchControl,
          in AttributeModificationSet modifications
       )
       raises (ModifyManagedObjects,
                ManagedGenericIRPSystem::OperationNotSupported,
                ManagedGenericIRPSystem::InvalidParameter,
                UndefinedMOException,
                IllegalDNFormatException,
                UndefinedScopeException,
                IllegalScopeTypeException,
                IllegalScopeLevelException,
                IllegalFilterFormatException,
                FilterComplexityLimit);

    };
};
#endif
```

Annex B (normative): CORBA IDL, Notification Definitions

Void.

```
#ifndef BasicCmNotifDefs_idl
#define BasicCmNotifDefs_idl
```

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

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

module BasicCmNotifDefs
{


      /**
       * 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
       */
      /*
      This block identifies attributes which are included as part of the Basic
      CM IRP. These attribute values should not clash with those defined for the
      attributes of notification header (see IDL of Notification IRP).
      */
      interface AttributeNameValue
      {
         const string SOURCE_INDICATOR = "SOURCE";
         const string ADDITIONAL_TEXT = "ADD_TEXT";
```

```
        const string CORRELATED_NOTIFICATIONS = "CORREL_NOTIFS";
    };

    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::AttributeNameValue::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::AttributeNameValue::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::AttributeNameValue::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::AttributeNameValue::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::AttributeNameValue::SYSTEM_DN;
```

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

        /**
         * 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 =
          BasicCmNotifDefs::AttributeNameValue::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 =
          BasicCmNotifDefs::AttributeNameValue::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 (informative): Change history

●●●