ETSI TS 132 603 V4.0.0 (2001-06)

Technical Specification

Digital cellular telecommunications system (Phase 2+) (GSM);
Universal Mobile Telecommunications System (UMTS);
Telecommunication Management;
Configuration Management;
Basic configuration management IRP: CORBA solution set
(3GPP TS 32.603 version 4.0.0 Release 4)

Reference
DTS/TSGS-0532603Uv4

Keywords
GSM, UMTS

ETSI

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

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

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

Intellectual Property Rights

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

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

Foreword

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

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

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

Contents

Foreword

This Technical Specification 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).

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
32.106-1	<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 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, methods and notifications.

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] 3GPP TS 32.111-3: “Alarm IRP: CORBA Solution Set”.

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

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

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 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	
`getMoAttributes`	`BasicCmIrpOperations::find_managed_objects` `BasicCmInformationIterator::next_basicCmInformations`	
`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

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 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
invokeIdentifier	The iterator returned from the call (BasicCmInformationIterator) identifies the request.	M
baseObjectInstance	in DN baseObject	M
scope	in searchControl (SearchControl.scope and SearchControl.level)	M
filter	in searchControl (SearchControl.filter)	M
attributeListIn	in requestedAttributes	M
managedObjectClass managedObjectInstance attributeListOut	parameter fetchedElements in the next_basicCmInformations in the BasicCmInformationIterator interface.	M
status	exception UndefinedMOException, exception IllegalDNFormatException, exception UndefinedScopeException, exception IllegalScopeTypeException, exception IllegalScopeLevelException, exception IllegalFilterFormatException, exception FilterComplexityLimit	M

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

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

IS Operation parameter	SS Method parameter	Qualifier
invokeIdentifier	The iterator returned from the call (BasicCmInformationIterator) identifies the request.	M
baseObjectInstance	in DN baseObject	M
scope	in searchControl (SearchControl.scope and SearchControl.level)	O
Not specified in IS	in searchControl (SearchControl.filter)	M
containment	parameter fetchedElements in the next_basicCmInformations in the BasicCmInformationIterator interface.	M
status	exception UndefinedMOException, exception IllegalDNFormatException, exception UndefinedScopeException, exception IllegalScopeTypeException, exception IllegalScopeLevelException, exception IllegalFilterFormatException, exception FilterComplexityLimit	M

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

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

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)	M

6.4 Notification attribute mapping

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

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 `CommonIRPConstDefs::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.
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 "CommonIRPConstDefs.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";

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


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

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

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


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

```
      typedef sequence<string> StringSet;

   };

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

   exception UndefinedScopeException {
       string reason;
   };

   exception FilterComplexityLimit {
      string reason;
   };

   exception NextBasicCmInformations {
      string reason;
   };

   exception InvalidParameter {
      string parameter;
   };

   exception GetBasicCmIRPVersion {
      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
  {
     string name;
     any value;
  };

  typedef sequence<MOAttribute> MOAttributeSet;
```

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

typedef sequence<Result> ResultSet;


/**
 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,InvalidParameter);

    /**
    This method destroys the iterator.
    */

    void destroy ();

}; // end of BasicCmInformationIterator


typedef sequence<string> AttributeNameSet;
```

```
/**
  * 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.
     */
    CommonIRPConstDefs::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 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 (UndefinedMOException,
               IllegalDNFormatException,
               UndefinedScopeException,
               IllegalScopeTypeException,
               IllegalScopeLevelException,
```

```
                    IllegalFilterFormatException,
                    FilterComplexityLimit);


    };
};
#endif
```

Annex B (normative): CORBA IDL, Notification Definitions

```
#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
```

```
 */
interface NotificationCommon
{

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


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


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


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



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


   /**
    *  This constant defines the name of the
    *  source indicator  property, which is transported in the
    *  filterable_body_fields
    */
```

```
   const string SOURCE_INDICATOR = "SOURCE";

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


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



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

};


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

   const string EVENT_TYPE = ET_OBJECT_DELETION;


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

};



/**
 *  Constant definitions for the MO created notification
 */
interface MOCreation : NotificationCommon
{
```

```
         const string EVENT_TYPE = ET_OBJECT_CREATION;


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

      };



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

         const string EVENT_TYPE = ET_ATTRIBUTE_VALUE_CHANGE;

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


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


       };

};


#endif
```

Annex C (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Jun 2001	S_12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	2.0.0	4.0.0

History

Document history		
V4.0.0	July 2001	Publication