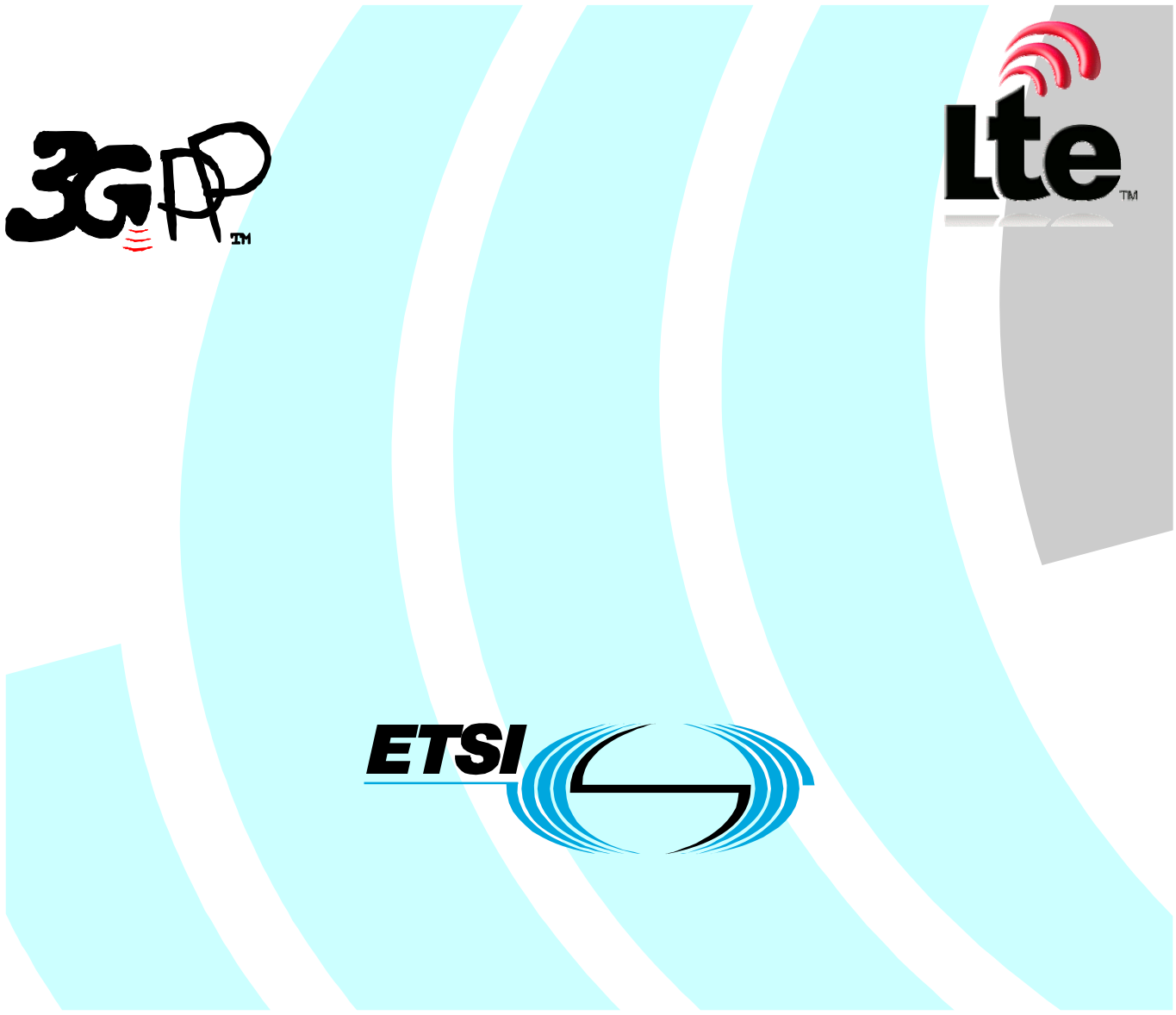
ETSI TS 132 603 V8.0.0 (2009-01)

Technical Specification

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Telecommunication management;
Configuration Management (CM);
Basic CM Integration Reference Point (IRP):
Common Object Request Broker Architecture (CORBA)
Solution Set (SS)
(3GPP TS 32.603 version 8.0.0 Release 8)



Reference
RTS/TSGS-0532603v800

Keywords
GSM, LTE, UMTS

ETSI

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

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

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

Intellectual Property Rights

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

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

Foreword

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

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

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

Contents

Foreword

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

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

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

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

32.601: "Configuration Management (CM); Basic CM Integration Reference Point (IRP); Requirements"

32.602: "Configuration Management (CM); Basic CM Integration Reference Point (IRP); Information Service (IS)"

32.603: "Configuration Management (CM); Basic CM Integration Reference Point (IRP); Common Object Request Broker Architecture (CORBA) Solution Set (SS)"

32.607: "Configuration Management (CM); Basic CM Integration Reference Point (IRP); SOAP Solution Set (SS)"

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.

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.

This Solution Set specification is related to 3G TS 32.602 V7.0.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: "Telecommunication management; Principles and high level requirements".

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

[3] 3GPP TS 32.600: "Telecommunication management; Configuration Management (CM); Concept and high-level requirements".

[4] 3GPP TS 32.602: "Telecommunication management; Configuration Management (CM); Basic CM Integration Reference Point (IRP) Information Service (IS)".

[5] 3GPP TS 32.300: "Telecommunication management; Configuration Management (CM); 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: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP); Common Object Request Broker Architecture (CORBA) Solution Set (SS)".

[10] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[11] 3GPP TS 32.663: "Telecommunication management; Configuration Management (CM); Kernel CM Integration Reference Point (IRP); Common Object Request Broker Architecture (CORBA) Solution Set (SS)".

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: [10].

This string (or sequence of strings, if more than one version is supported) is returned in getBasicCmIRPVersion method.

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 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]).

5.2 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 TS 32.663 [11]) is emitted.

6.2 Operation mapping

The Basic CM IRP: IS (see 3GPP TS 32.602 [4]) defines semantics of operation visible across the Basic Configuration Management IRP. Table 1 indicates mapping of these operations to their equivalents defined in this SS.

Table 1: Mapping from IS Operation to SS equivalents

IS Operation (3GPP TS 32.602 [4])	SS Method	Qualifier
getMoAttributes	BasicCmIrpOperations::find_managed_objects BasicCmInformationIterator::next_basic_cm_informations	M
getContainment	BasicCmIrpOperations::find_managed_objects BasicCmInformationIterator::next_basic_cm_informations	O
getIRPVersion (see note)	get_basic_cm_irp_version	M
cancelOperation	BasicCmInformationIterator::destroy	O
createMo	BasicCmIrpOperations::create_managed_object	O
deleteMo	BasicCmIrpOperations::delete_managed_objects DeleteResultIterator::next_basic_cm_informations DeleteResultIterator::next_delete_errors	O
setMoAttributes	BasicCmIrpOperations::modify_managed_objects ModifyResultIterator::next_basic_cm_informations ModifyResultIterator::next_modification_errors	O
getOperationProfile (see note)	get_basic_cm_irp_operation_profile	O
getNotificationProfile (see note)	get_basic_cm_irp_notification_profile	O
NOTE: This operation is of IOC ManagedGenericIRP specified in [10]. The IOC BasicCmIRP of [4] inherits from it.		

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 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
invokeIdentifier	- (No equivalence)	-
invokeIdentifierOut	Return value of type BasicCmInformationIterator	M
baseObjectInstance	DN base_object	M
scope	SearchControl search_control (SearchControl.type and SearchControl.level)	M
filter	SearchControl search_control (SearchControl.filter)	M
attributeListIn	AttributeNameSet requested_attributes	M
managedObjectClass managedObjectInstance attributeListOut	Return value of type BasicCmInformationIterator - parameter out ResultSet fetched_elements of method next_basic_cm_informations	M
status	Exceptions: FindManagedObjects, ManagedGenericIRPSystem::InvalidParameter, UndefinedMOException, IllegalDNFormatException, UndefinedScopeException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalFilterFormatException, 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	- (No equivalence)	-
invokeIdentifierOut	Return value of type BasicCmInformationIterator	M
baseObjectInstance	DN base_object	M
scope	SearchControl search_control (SearchControl.type and SearchControl.level)	O
Not specified in IS	SearchControl search_control (SearchControl.filter)	M
containment	Return value of type BasicCmInformationIterator - parameter out ResultSet fetched_elements of method next_basic_cm_informations	M
status	Exceptions: FindManagedObjects, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ManagedGenericIRPSystem::ValueNotSupported, UndefinedMOException, IllegalDNFormatException, UndefinedScopeException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalFilterFormatException, FilterComplexityLimit	M

Table 4: Mapping from IS getIRPVersion 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	Exceptions: ManagedGenericIRPSystem::OperationNotSupported, DestroyException	M

Table 6: Mapping from IS createMo parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
managedObjectClass managedObjectInstance	DN object_name	M
referenceObjectInstance	DN reference_object	O
attributeListIn attributeListOut	MoAttributeSet attributes	M
status	AttributeErrorSeq attribute_errors Exceptions: CreateManagedObject, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, UndefinedMOException, IllegalDNFormatException, DuplicateMO, CreateNotAllowed, ObjectClassMismatch, NoSuchObjectClass, ParentObjectDoesNotExist	M

Table 7: Mapping from IS deleteMo parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
baseObjectInstance	DN base_object	M
scope	SearchControl search_control (SearchControl.type and SearchControl.level)	M
filter	SearchControl search_control (SearchControl.filter)	M
deletionList	Return value of type DeleteResultIterator - parameter out ResultSet fetched_elements of method next_basic_cm_informations	M
status	Return value of type DeleteResultIterator - parameter out DeleteErrorSeq fetched_delete_errors of method next_delete_errors 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	DN base_object	M
scope	SearchControl search_control (SearchControl.type and SearchControl.level)	M
filter	SearchControl search_control (SearchControl.filter)	M
modificationList	AttributeModificationSet modifications	M
modificationListOut	Return value of type ModifyResultIterator - parameter out ResultSet fetched_elements of method next_basic_cm_informations	M
status	Return value of type ModifyResultIterator - parameter out ModifyAttributeErrorsSeq fetched_modify_errors of method next_modify_errors Exceptions: ModifyManagedObjects, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter, UndefinedMoException, IllegalDNFormatException, UndefinedScopeException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalFilterFormatException, FilterComplexityLimit	M

Table 9: Mapping from IS getOperationProfile parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
irpVersion	ManagedGenericIRPConstDefs::VersionNumber basic_cm_irp_version	M
operationNameProfile, operationParameterProfile	Return value of type ManagedGenericIRPConstDefs::MethodList	M
status	Exceptions: GetBasicCmIRPOperationProfile, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table 10: Mapping from IS getNotificationProfile parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
irpVersion	ManagedGenericIRPConstDefs::VersionNumber basic_cm_irp_version	M
notificationNameProfile, notificationParameterProfile	Return value of type ManagedGenericIRPConstDefs::MethodList	M
status	Exceptions: GetBasicCmIRPNotificationProfile, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

7 Void

Annex A (normative): CORBA IDL, Access Protocol

A.1 IDL specification (file name "BasicCMIRPConstDefs.idl")

```
//File: BasicCMIRPConstDefs.idl

#ifndef _BASIC_CM_IRP_CONST_DEFS_IDL_
#define _BASIC_CM_IRP_CONST_DEFS_IDL_

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

/* ## Module: BasicCMIRPConstDefs
This module contains commonly used definitions for BasicCMIRP.
================================================================
*/
module BasicCMIRPConstDefs
{
   /**
    * 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".
    */
   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;

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

   /**
    * 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;
   Filter filter_;
   ResultContents contents;
};

/**
 * Represents an attribute: "name" is the attribute name
 * and "value" is the attribute value.
 */
struct MOAttribute
{
   MOAttributeName name;
   MOAttributeValue 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_name" field),
 * - the category of the error ("error" field),
 * - optionally, additional details on the error ("reason" field).
 */
struct DeleteError
{
   DN object_name;
   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_name" field),
 * - a sequence containing the attribute errors ("errors" field).
 */
struct ModifyAttributeErrors
{
   DN object_name;
   AttributeErrorSeq errors;
};

typedef sequence <ModifyAttributeErrors> ModifyAttributeErrorsSeq;

typedef sequence <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 provided 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;

};

#endif // _BASIC_CM_IRP_CONST_DEFS_IDL_
```

A.2 IDL specification (file name "BasicCMIRPSystem.idl")

```
//File: BasicCMIRPSystem.idl
#ifndef _BASIC_CM_IRP_SYSTEM_IDL_
#define _BASIC_CM_IRP_SYSTEM_IDL_

#include <ManagedGenericIRPConstDefs.idl>
#include <ManagedGenericIRPSystem.idl>
#include <BasicCMIRPConstDefs.idl>

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

module BasicCmIRPSystem
{
   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 DuplicateMO {};

   exception CreateNotAllowed {};

   exception ObjectClassMismatch {};

   exception NoSuchObjectClass {
      BasicCMIRPConstDefs::MOClass objectClass;
   };

   exception ParentObjectDoesNotExist {};

   /**
    * 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 GetBasicCmIRPOperationProfile { string reason; };
   exception GetBasicCmIRPNotificationProfile { string reason; };
   exception FindManagedObjects { string reason; };
   exception CreateManagedObject { string reason; };
   exception DeleteManagedObjects { string reason; };
   exception ModifyManagedObjects { string reason; };

   /**
   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 "fetched_elements" out
    parameter.
   @parm fetched_elements the elements.
   @returns A boolean indicating if any elements are returned.
    "fetched_elements" is empty when the BasicCmInformationIterator is
    empty.
   */

   boolean next_basic_cm_informations (
      in unsigned short how_many,
      out BasicCMIRPConstDefs::ResultSet fetched_elements
   )
   raises (
      NextBasicCmInformations,
      ManagedGenericIRPSystem::InvalidParameter,
      ManagedGenericIRPSystem::OperationNotSupported);

   /**
   This method destroys the iterator.
   */
   void destroy ()
   raises (
      DestroyException,
      ManagedGenericIRPSystem::OperationNotSupported);

}; // 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_basic_cm_informations" has the same behaviour as
   for interface BasicCmInformationIterator, except that:
   - The Managed Object information returned in parameter
     "fetched_elements" 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_basic_cm_informations" 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
    "fetched_delete_errors" out parameter.
   @parm fetched_delete_errors: the deletion errors.
   @returns: a boolean indicating if any deletion errors are returned.
```

```
      */

      boolean next_delete_errors (
         in unsigned short how_many,
         out BasicCMIRPConstDefs::DeleteErrorSeq fetched_delete_errors
      )
      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_basic_cm_informations" has the same behaviour as
      for interface BasicCmInformationIterator, except that:
      - The Managed Object information returned in parameter
        "fetched_elements" 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_basic_cm_informations" 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
       "fetched_modify_errors" out parameter.
      @parm fetched_modify_errors: the modification errors.
      @returns: a boolean indicating if any modification errors are returned.
      */

      boolean next_modification_errors (
         in unsigned short how_many,
         out BasicCMIRPConstDefs::ModifyAttributeErrorsSeq
            fetched_modify_errors
      )
      raises (
         NextModifyErrors,
         ManagedGenericIRPSystem::InvalidParameter);

   }; // end of ModifyResultIterator

   /**
     * 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_basic_cm_irp_version()
         raises (GetBasicCmIRPVersion);

      /**
```

```
 * Return the operation profile for a specific Basic CM IRP version.
 *
 * @raises GetBasicCmIRPOperationProfile when the system for some reason
 *   cannot return the supported operations and parameters.
 * @returns the list of all supported operations and their supported
 *   parameters for the specified version.
 */
ManagedGenericIRPConstDefs::MethodList get_basic_cm_irp_operation_profile
(
   in ManagedGenericIRPConstDefs::VersionNumber basic_cm_irp_version
)
raises (
   GetBasicCmIRPOperationProfile,
   ManagedGenericIRPSystem::OperationNotSupported,
   ManagedGenericIRPSystem::InvalidParameter);

/**
 * Return the notification profile for a specific Basic CM IRP version.
 *
 * @raises GetBasicCmIRPNotificationProfile when the system for some
 *   reason cannot return the supported notifications and parameters.
 * @returns the list of all supported notifications and their supported
 *   parameters for the specified version.
 */
ManagedGenericIRPConstDefs::MethodList
   get_basic_cm_irp_notification_profile (
      in ManagedGenericIRPConstDefs::VersionNumber basic_cm_irp_version
)
raises (
   GetBasicCmIRPNotificationProfile,
   ManagedGenericIRPSystem::OperationNotSupported,
   ManagedGenericIRPSystem::InvalidParameter);

/**
 * 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 base_object The start MO in the containment tree.
 * @parm search_control the SearchControl to use.
 * @parm requested_attributes 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 IRPAgent.
 * @see SearchControl
 * @see BasicCmInformationIterator
 */
BasicCmInformationIterator find_managed_objects(
   in BasicCMIRPConstDefs::DN base_object,
   in BasicCMIRPConstDefs::SearchControl search_control,
   in BasicCMIRPConstDefs::AttributeNameSet requested_attributes)
raises (
```

```
   FindManagedObjects,
   ManagedGenericIRPSystem::ParameterNotSupported,
   ManagedGenericIRPSystem::InvalidParameter,
   ManagedGenericIRPSystem::ValueNotSupported,
   ManagedGenericIRPSystem::OperationNotSupported,
   UndefinedMOException,
   IllegalDNFormatException,
   UndefinedScopeException,
   IllegalScopeTypeException,
   IllegalScopeLevelException,
   IllegalFilterFormatException,
   FilterComplexityLimit);

/**
 * Performs the creation of a MO instance in the MIB maintained
 * by the IRPAgent.
 *
 * @parm object_name: the distinguished name of the MO to create.
 * @parm reference_object: 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 attribute_errors: 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.
 * @raises ParentObjectDoesNotExist: The parent MO instance of the
 *  ManagedEntity specified to be created does not exist.
 */
void create_managed_object (
   in BasicCMIRPConstDefs::DN object_name,
   in BasicCMIRPConstDefs::DN reference_object,
   inout BasicCMIRPConstDefs::MOAttributeSet attributes,
   out BasicCMIRPConstDefs::AttributeErrorSeq attribute_errors
)
raises (
   CreateManagedObject,
   ManagedGenericIRPSystem::OperationNotSupported,
   ManagedGenericIRPSystem::ParameterNotSupported,
   ManagedGenericIRPSystem::InvalidParameter,
   UndefinedMOException,
   IllegalDNFormatException,
   DuplicateMO,
   CreateNotAllowed,
   ObjectClassMismatch,
   NoSuchObjectClass,
   ParentObjectDoesNotExist);

/**
 * 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 base_object: the start MO in the containment tree.
 * @parm search_control: 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 BasicCMIRPConstDefs::DN base_object,
       in BasicCMIRPConstDefs::SearchControl search_control
    )
    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 base_object: the start MO in the containment tree.
     * @parm search_control: 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 BasicCMIRPConstDefs::DN base_object,
       in BasicCMIRPConstDefs::SearchControl search_control,
       in BasicCMIRPConstDefs::AttributeModificationSet modifications
    )
    raises (
       ModifyManagedObjects,
       ManagedGenericIRPSystem::OperationNotSupported,
       ManagedGenericIRPSystem::InvalidParameter,
       UndefinedMOException,
       IllegalDNFormatException,
```

```
            UndefinedScopeException,
            IllegalScopeTypeException,
            IllegalScopeLevelException,
            IllegalFilterFormatException,
            FilterComplexityLimit);
    };

};

#endif // _BASIC_CM_IRP_SYSTEM_IDL_
```

Annex B (informative): Change history

Change history								
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Cat**	**Old**	**New**
Jun 2001	S_12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	--	2.0.0	4.0.0
Sep 2001	S_13	SP-010476	0001	--	Correction of invokeIdentifier usage	F	4.0.0	4.1.0
Mar 2002	S_15	SP-020019	0002	--	Correction of erroneous CORBA module names and mapping tables	F	4.1.0	4.2.0
Mar 2002	S_15	SP-020019	0003	--	Corrections to Basic CM IRP CORBA Solution Set IDLs	F	4.1.0	4.2.0
Mar 2002	S_15	SP-020038	0004	--	Addition of missing CORBA exception "ManagedGenericIRPSystem::ValueNotSupported" onto CORBA method "find_managed_objects"	F	4.1.0	4.2.0
Jun 2002	S_16	SP-020294	0005	--	Correcting IDL definitions of notification structured event Name Value pair names	F	4.2.0	4.3.0
Jul 2002	--	--	--	--	Updated the Version number (420->431) and the Date on the cover page	--	4.3.0	4.3.1
Sep 2002	S_17	SP-020483	0006	--	Add Active Basic CM feature - CORBA Solution Set	B	4.3.1	5.0.0
Mar 2003	S_19	SP-030139	0007	--	Add CORBA equivalents to IS operations "get{Operation Notification}Profile" - alignment with 32.602 & 32.312	F	5.0.0	5.1.0
Mar 2003	S_19	SP-030139	0008	--	Correction of IDL errors	F	5.0.0	5.1.0
Mar 2003	S_19	SP-030144	0009	--	Add description for notifications of each activeCM operation and one exception for createMO - alignment with 32.602, Information Service	F	5.0.0	5.1.0
Jun 2003	S_20	SP-030279	0010	--	Alignment with Basic CM IRP information service (32.602) - add one exception for the operation createMO	F	5.1.0	5.2.0
Mar 2004	S_23	SP-040105	--	--	Automatic upgrade to Rel-6 (no CR)	--	5.2.0	6.0.0
Sep 2004	S_25	SP-040567	0012	--	Removal of Rules for NRM extensions - Align with 32.622 (Generic NRM IS)	A	6.0.0	6.1.0
Sep 2004	S_25	SP-040566	0014	--	Removal of unused/duplicate definition of types MOReference and MOReferenceSet	A	6.0.0	6.1.0
Dec 2004	S_26	SP-040806	0015	--	Align the IDL style in the CORBA SS with the IDL Style Guide in 32.150	F	6.1.0	6.2.0
Mar 2005	S_27	SP-050044	0016	--	IDL incompliant to the style guide	F	6.2.0	6.3.0
Mar 2005	S_27	SP-050044	0017	--	Generic System Context, update of reference to IS specification	F	6.2.0	6.3.0
Apr 2005	--	--	--	--	Changed back TS Title from Bulk (v630) to Basic (620); erroneously changed with corrupted TS cover.	--	6.3.0	6.3.1
Sep 2005	SA_29	SP-050461	0018	--	Align the CORBA SS IDL with TS 32.150 Style Guide	F	6.3.1	6.4.0
Jun 2007	SA_36	--	--	--	Automatic upgrade to Rel-7 (no CR) at freeze of Rel-7. Deleted reference to CMIP SS, discontinued from R7 onwards.	--	6.1.0	7.0.0
Dec 2008	SA_42	--	--	--	Upgrade to Release 8	--	7.0.0	8.0.0

History

Document history		
V8.0.0	January 2009	Publication