ETSI TS 132 603 V5.0.0 (2002-09)

Technical Specification

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
Telecommunication management;
Configuration Management (CM);
Basic Configuration Management
Integration Reference Point (IRP): CORBA solution set
(3GPP TS 32.603 version 5.0.0 Release 5)

Reference
RTS/TSGS-0532603v500

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

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.fr

Copyright Notification

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 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.

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 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 (CM): Concept and High-level Requirements".

[4] 3GPP TS 32.602: "Configuration Management (CM); Basic CM Integration Reference Point (IRP) information model".

[5] 3GPP TS 32.300: "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: "Configuration Management (CM); Notification Integration Reference Point; CORBA solution set".

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

[11] 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: [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: IM (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_basicCmInformations	M
getContainment	BasicCmIrpOperations::find_managed_objects BasicCmInformationIterator::next_basicCmInformations	O
getBasicCmIRPVersion	get_basicCm_IRP_version	M
cancelOperation	BasicCmInformationIterator::destroy	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 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	in DN baseObject	M
scope	in SearchControl searchControl (SearchControl.type and SearchControl.level)	M
filter	in SearchControl searchControl (SearchControl.filter)	M
attributeListIn	in AttributeNameSet requestedAttributes	M

IS Operation parameter	SS Method parameter	Qualifier
managedObjectClass managedObjectInstance attributeListOut	Return value of type BasicCmInformationIterator - parameter out ResultSet fetchedElements of method next_basicCmInformations	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	in DN baseObject	M
scope	in SearchControl searchControl (SearchControl.type and SearchControl.level)	O
Not specified in IS	in SearchControl searchControl (SearchControl.filter)	M
containment	Return value of type BasicCmInformationIterator - parameter out ResultSet fetchedElements of method next_basicCmInformations	M
status	Exceptions: FindManagedObjects, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ManagedGenericIRPSystem::ValueNotSupported, UndefinedMOException, IllegalDNFormatException, UndefinedScopeException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalFilterFormatException, 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	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

IS Operation parameter	SS Method parameter	Qualifier
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
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

7 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.

7.1 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.

7.2 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
{

   /**
    * 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;


   /**
    *  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 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.
     */
   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" 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<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 (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
Sep 2001	S_13	SP-010476	001	--	Correction of invokeIdentifier usage	4.0.0	4.1.0
Mar 2002	S_15	SP-020019	002	--	Correction of erroneous CORBA module names and mapping tables	4.1.0	4.2.0
Mar 2002	S_15	SP-020019	003	--	Corrections to Basic CM IRP CORBA Solution Set IDLs	4.1.0	4.2.0
Mar 2002	S_15	SP-020038	004	--	Addition of missing CORBA exception "ManagedGenericIRPSystem::ValueNotSupported" onto CORBA method "find_managed_objects"	4.1.0	4.2.0
Jun 2002	S_16	SP-020294	005	--	Correcting IDL definitions of notification structured event Name Value pair names	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	006	--	Add Active Basic CM feature - CORBA Solution Set	4.3.1	5.0.0

History

Document history		
V5.0.0	September 2002	Publication