TSGS#25(04)0566

Source: **SA5 (Telecom Management)**

Title: **2 Rel-5/6 CR 32.603 Removal of unused/duplicate definition of types MOReference and MOReferenceSet**

Document for: **Approval**

Agenda Item: **7.5.3**

Doc-1st-Level	Doc-2nd-Level	Spec	CR	Rev	Phase	Subject	Cat	Ver-Cur	Wi
SP-040566	S5-046868	32.603	013	--	Rel-5	Removal of unused/duplicate definition of types MOReference and MOReferenceSet	F	5.2.0	OAM-NIM
SP-040566	S5-046869	32.603	014	--	Rel-6	Removal of unused/duplicate definition of types MOReference and MOReferenceSet	A	6.0.0	OAM-NIM

CR-Form-v7

CHANGE REQUEST

⌘	32.603	CR	013	⌘	rev	-	⌘	Current version:	5.2.0	⌘

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

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

Title:	⌘	Removal of unused/duplicate definition of types MOReference and MOReferenceSet
Source:	⌘	**SA5** (clemens.suerbaum@siemens.com)
Work item code:	⌘	OAM-NIM
Date:	⌘	20/08/2004
Category:	⌘	**F**
Release:	⌘	Rel-5

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

Use one of the following releases:

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

Reason for change:	⌘	MOReference and MOReferenceSet are not used within this spec and already defined in TS 32.623
Summary of change:	⌘	Removal of type definitions MOReference and MOReferenceSet
Consequences if not approved:	⌘	Ambiguous and duplicated definitions

Clauses affected:	⌘	Annex A

		Y	N			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
		X		O&M Specifications		Rel-6 32.603

Other comments:	⌘	Rel-6 Mirror CR in S5-.046869.

How to create CRs using this form:

Change in Annex A

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;
};

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; };


/**
 *
 * 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 ("objectName" field),
 * - the category of the error ("error" field),
 * - optionally, additional details on the error ("reason" field).
 */
struct DeleteError
{
   DN objectName;
   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 ("objectName" field),
 * - a sequence containing the attribute errors ("errors" field).
 */
struct ModifyAttributeErrors
```

```
{
   DN objectName;
   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 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;


/**
  * 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);

    /**
     * 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_basicCm_IRP_operation_profile
    (
       in ManagedGenericIRPConstDefs::VersionNumber basicCm_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_basicCm_IRP_notification_profile (
          in ManagedGenericIRPConstDefs::VersionNumber basicCm_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 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.
 * @raises ParentObjectDoesNotExist: The parent MO instance of the
 *  ManagedEntity specified to be created does not exist.
 */
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,
        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 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
```

End of Change in Annex A
End of Document

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
Mar 2003	S_19	SP-030139	007	--	Add CORBA equivalents to IS operations "get{Operation Notification}Profile" - alignment with 32.602 & 32.312	5.0.0	5.1.0
Mar 2003	S_19	SP-030139	008	--	Correction of IDL errors	5.0.0	5.1.0
Mar 2003	S_19	SP-030144	009	--	Add description for notifications of each activeCM operation and one exception for createMO - alignment with 32.602, Information Service	5.0.0	5.1.0
Jun 2003	S_20	SP-030279	010	--	Alignment with Basic CM IRP information service (32.602) - add one exception for the operation createMO	5.1.0	5.2.0

CR-Form-v7

CHANGE REQUEST

⌘	32.603	CR	014	⌘	rev	-	⌘	Current version:	6.0.0	⌘

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

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

Title:	⌘	Removal of unused/duplicate definition of types MOReference and MOReferenceSet
Source:	⌘	**SA5** (clemens.suerbaum@siemens.com)
Work item code:	⌘	OAM-NIM
Date:	⌘	20/08/2004
Category:	⌘	**A**
Release:	⌘	Rel-6

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

Use one of the following releases:

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

Reason for change:	⌘	MOReference and MOReferenceSet are not used within this spec and already defined in TS 32.623
Summary of change:	⌘	Removal of type definitions MOReference and MOReferenceSet
Consequences if not approved:	⌘	Ambiguous and duplicated definitions

Clauses affected:	⌘	Annex A

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

Other comments:	⌘	Rel-6 Mirror CR of S5-046868.

How to create CRs using this form:

Change in Annex A

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;
};

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; };


/**
 *
 * 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 ("objectName" field),
 * - the category of the error ("error" field),
 * - optionally, additional details on the error ("reason" field).
 */
struct DeleteError
{
   DN objectName;
   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 ("objectName" field),
 * - a sequence containing the attribute errors ("errors" field).
 */
struct ModifyAttributeErrors
```

```
{
   DN objectName;
   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 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;


/**
  * 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);

    /**
     * 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_basicCm_IRP_operation_profile
    (
       in ManagedGenericIRPConstDefs::VersionNumber basicCm_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_basicCm_IRP_notification_profile (
          in ManagedGenericIRPConstDefs::VersionNumber basicCm_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 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.
 * @raises ParentObjectDoesNotExist: The parent MO instance of the
 *  ManagedEntity specified to be created does not exist.
 */
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,
        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 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
```

End of Change in Annex A
End of Document

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
Mar 2003	S_19	SP-030139	007	--	Add CORBA equivalents to IS operations "get{Operation Notification}Profile" - alignment with 32.602 & 32.312	5.0.0	5.1.0
Mar 2003	S_19	SP-030139	008	--	Correction of IDL errors	5.0.0	5.1.0
Mar 2003	S_19	SP-030144	009	--	Add description for notifications of each activeCM operation and one exception for createMO - alignment with 32.602, Information Service	5.0.0	5.1.0
Jun 2003	S_20	SP-030279	010	--	Alignment with Basic CM IRP information service (32.602) - add one exception for the operation createMO	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