Technical Specification Group Services and System Aspects
Meeting #15, Cheju Island, Korea, 11-14 March 2002

TSGS#15(02)0019

Source: **SA5 (Telecom Management)**

Title: **2 Rel-4 CR 32.603 (Basic configuration management IRP: CORBA SS)**

Document for: **Decision**

Agenda Item: **7.5.3**

Doc-1st-Level	Spec	CR	R	Phase	Subject	Cat	Version-Current	Version-New	Doc-2nd-Level	Workitem
SP-020019	32.603	002		Rel-4	**Correction of erroneous CORBA module names and mapping tables**	F	4.1.0	4.2.0	S5-020062	OAM-CM
SP-020019	32.603	003		Rel-4	**Corrections to Basic CM IRP CORBA Solution Set IDLs**	F	4.1.0	4.2.0	S5-020063	OAM-CM

3GPP TSG-SA5 (Telecom Management) **S5-020062**
Meeting #25, Sophia Antipolis, France 14 - 18 January 2002 **S5C020044r1**

CR-Form-v5

CHANGE REQUEST

⌘ **32.603** **CR** **002** ⌘ **rev** **-** ⌘ Current version: **4.1.0** ⌘

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

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

Title:	⌘	Correction of erroneous CORBA module names and mapping tables
Source:	⌘	SA5
Work item code:	⌘	OAM-CM
Date:	⌘	18/01/2002
Category:	⌘	**F**
Release:	⌘	REL-4

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

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

Reason for change:	⌘	There are erroneous CORBA module names and mapping tables in the TS.
Summary of change:	⌘	• Correct erroneous CORBA modules in operation parameter and notification parameter mapping tables. • Correct erroneous IS parameter in operation parameter mapping table. • Add missing CORBA exception in operation parameter mapping table.
Consequences if not approved:	⌘	The TS would contain erroneous misleading information.

Clauses affected:	⌘	6.3, 7		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘			

6.3 Operation parameter mapping

●●●

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

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

●●●

7 Use of OMG Structured Event

●●●

Table 11: Use of OMG Structured Event

SS Attribute	OMG CORBA Structured Event attribute	Comment
●●●		
Event Time	One NV pair of filterable_ body_fields	It is an attribute of notificationHeader. Name of NV pair is a string, BasicCmNotifDefs::<interface>::EVENT_TIME where <interface> is either MOCreation, MODeletion or AttributeValueChange. Value of NV pair is a ~~CommonIRPConstDefs~~ ManagedGenericIRPConstDefs::IRPTime defined in 3GPP TS 32.303 [9]. See corresponding table in Notification IRP: CORBA SS (3GPP TS 32.303 [9]).
●●●		

●●●

3GPP TSG-SA5 (Telecom Management) **S5-020063**
Meeting #25, Sophia Antipolis, France 14 - 18 January 2002 **S5C010533rev4**

CR-Form-v4

CHANGE REQUEST

⌘ **32.603** **CR** **003** ⌘ ev **-** ⌘ Current version: **4.1.0** ⌘

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

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

Title:	⌘	Corrections to Basic CM IRP CORBA Solution Set IDLs
Source:	⌘	SA5
Work item code:	⌘	OAM-CM
Date:	⌘	18/01/2002
Category:	⌘	**F**
Release:	⌘	REL-4

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

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

Reason for change:	⌘	There are errors in Basic CM CORBA Solution Set IDLs.
Summary of change:	⌘	Change references to "`CommonIRPConstDefs.idl`" in Annex A to "`ManagedGenericIRPConstDefs.idl`". Correct the references related to `NotificationIRPConstDefs`:: in Annex B.
Consequences if not approved:	⌘	It will not be possible to process the IDL files with these errors.

Clauses affected:	⌘	Annex A, Annex B.
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

Annex A (normative):
CORBA IDL, Access Protocol

```
#ifndef BasicCmIRPSystem_idl
#define BasicCmIRPSystem_idl

#include "CommonIRPConstDefs.idl"
#include "ManagedGenericIRPConstDefs.idl"

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

module BasicCmIRPSystem
{

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

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


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

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

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


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

```
    */
   typedef sequence<string> StringSet;

};

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

exception UndefinedScopeException {
    string reason;
};

exception FilterComplexityLimit {
   string reason;
};

exception NextBasicCmInformations {
   string reason;
};

exception InvalidParameter {
   string parameter;
};

exception GetBasicCmIRPVersion {
   string reason;
};


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

 /**
  * ResultContents is used to tell how much information to get back
  * from the find_managed_objects operation.
  *
  * NAMES: Used to get only Distinguished Name
  *        for MOs.
  *        The name contains both the MO class
  *        and the names of all superior objects in the naming
  *        tree.
  *
  * NAMES_AND_ATTRIBUTES: Used to get both NAMES plus
  *    MO attributes (all or selected).
  */
 enum ResultContents
```

```
 {
    NAMES,
    NAMES_AND_ATTRIBUTES
 };

 /**
  * ScopeType defines the kind of scope to use in a search
  * together with SearchControl.level, in a SearchControl value.
  *
  * SearchControl.level is always >= 0. If a level is bigger than the
  * depth of the tree there will be no exceptions thrown.
  * BASE_ONLY: level ignored, just return the base object.
  * BASE_NTH_LEVEL: return all subordinate objects that are on "level"
  * distance from the base object, where 0 is the base object.
  * BASE_SUBTREE: return the base object and all of its subordinates
  * down to and including the nth level.
  * BASE_ALL: level ignored, return the base object and all of it's
  * subordinates.
  */
 enum ScopeType
 {
    BASE_ONLY,
    BASE_NTH_LEVEL,
    BASE_SUBTREE,
    BASE_ALL
 };

 /**
  * SearchControl controls the find_managed_object search,
  * and contains:
  * the type of scope ("type" field),
  * the level of scope ("level" field), level 0 means the "baseObject",
  *    level 1 means baseobject including its sub-ordinates etc..
  * the filter ("filter" field),
  * the result type ("contents" field).
  * The type, level and contents fields are all mandatory.
  * The filter field contains the filter expression.
  *  The string "TRUE" indicates "no filter",
  * i.e. a filter that matches everything.
  */
 struct SearchControl
 {
    ScopeType type;
    unsigned long level;
    FilterType filter;
    ResultContents contents;
 };

 /**
  * Represents an attribute: "name" is the attribute name
  * and "value" is the attribute value in form of a CORBA Any.
  * The allowed attribute value types are defined in the
  * AttributeTypes module.
  */
struct MOAttribute
{
   string name;
   any value;
};

typedef sequence<MOAttribute> MOAttributeSet;
```

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

typedef sequence<Result> ResultSet;


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

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

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

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

     @parm how_many how many elements to return in the "fetchedElements" out
      parameter.
     @parm fetchedElements the elements.
     @returns A boolean indicating if any elements are returned.
      "fetchedElements" is empty when the BasicCmInformationIterator is
      empty.
    */

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

    /**
    This method destroys the iterator.
    */

    void destroy ();

}; // end of BasicCmInformationIterator


typedef sequence<string> AttributeNameSet;
```

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

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


        /**
         * Performs a containment search, using a SearchControl to
         * control the search and the returned results.
         *
         * All MOs in the scope constitute a set that the filter works on.
         * The result BasicCmInformationIterator contains all matched MOs,
         * with the amount of detail specified in the SearchControl.
         * For the special case when no managed objects are matched in
         * find_managed_objects, the BasicCmInformationIterator will be returned.
         * Executing the next_basicCmInformations in the
         * BasicCmInformationIterator will return FALSE for
         * completion.
         *
         * @parm baseObject The start MO in the containment tree.
         * @parm searchControl the SearchControl to use.
         * @parm requestedAttributes defines which attributes to get.
         *   If this parameter is empty (""),  all attributes shall
         *   be returned. In this version this is the only supported semantics.
         *   Note that this argument is only
         *   relevant if ResultContents in the search control is
         *   specifed to NAMES_AND_ATTRIBUTES.
         *
         *
         * @raises UndefinedMOException The MO does not exist.
         * @raises IllegalDNFormatException The dn syntax string is
         * malformed.
         * @raises IllegalScopeTypeException The ScopeType in scope contains
         * an illegal value.
         * @raises IllegalScopeLevelException The scope level is negative
         * (<0).
         * @raises IllegalFilterFormatException The filter string is
         * malformed.
         * @raises FilterComplexityLimit if the filter syntax is correct,
         *   but the filter is too complex to be processed by the IRP agent.
         * @see SearchControl
         * @see BasicCmInformationIterator
         */
        BasicCmInformationIterator find_managed_objects(in DN baseObject,
                                    in SearchControl searchControl,
                                    in AttributeNameSet requestedAttributes)
           raises (UndefinedMOException,
                   IllegalDNFormatException,
                   UndefinedScopeException,
```

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


    };
};
#endif
```

Annex B (normative):
CORBA IDL, Notification Definitions

```
#ifndef BasicCmNotifDefs_idl
#define BasicCmNotifDefs_idl

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

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

module BasicCmNotifDefs
{


      /**
       * Definition of ITU-T defined semantics.
       * These constants are used in the type_name
       * (header.fixed_header.event_type.type_name)
       * field to denote the notification type
       * Note all values are unique among themselves.  Other IRP documents
       * cannot use the same values.
       */
      const string ET_OBJECT_CREATION = "x6";

      const string ET_OBJECT_DELETION = "x7";

      const string ET_ATTRIBUTE_VALUE_CHANGE = "x8";



      /**
       * Information about one attribute
       * - name defines the name of the attribute
       * - value defines the value of the attribute
       *
       */
      struct MOAttribute
      {
         string name;
         any value;
      };


      /**
       * A set of attribute names and values
       */
      typedef sequence<MOAttribute> MOAttributeSet;


      /**
       *  This interface defines fields that are common for all
       *  notification types.
       *  All constants in the scope of this interface will be
       *  visible in the interfaces that inherits this.
       *  For instance constant
       *  NotificationCommon::MANAGED_OBJECT_CLASS
       *  can be addressed by MODeletion::MANAGED_OBJECT_CLASS
```

```
 */
interface NotificationCommon
{

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

NotificationIRPConstDefs::~~NV_MANAGED_OBJECT_CLASS~~AttributeNameValue::MANAGED_OBJECT_CLASS;


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

NotificationIRPConstDefs::~~NV_MANAGED_OBJECT_INSTANCE~~AttributeNameValue::MANAGED_OBJECT_INSTANCE;


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

NotificationIRPConstDefs::~~NV_NOTIFICATION_ID~~AttributeNameValue::NOTIFICATION_ID;


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

NotificationIRPConstDefs::~~NV_EVENT_TIME~~AttributeNameValue::EVENT_TIME;


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

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

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


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

NotificationIRPConstDefs::~~NV_ADDITIONAL_TEXT~~AttributeNameValue::ADDITIONAL_TEXT;



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

NotificationIRPConstDefs::~~NV_CORRELATED_NOTIFICATIONS~~AttributeNameValue::CORRELATED_NOTIFICATIONS;

      };


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

         const string EVENT_TYPE = ET_OBJECT_DELETION;


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

```
      };


      /**
       *  Constant definitions for the MO created notification
       */
      interface MOCreation : NotificationCommon
      {
         const string EVENT_TYPE = ET_OBJECT_CREATION;


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

      };



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

         const string EVENT_TYPE = ET_ATTRIBUTE_VALUE_CHANGE;

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


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


       };

};

#endif
```