TSGS#24(04)0248

Source: SA5 (Telecom Management)

Title: Rel-6 CR 32.363 (Entry Point IRP CORBA SS)

Document for: Decision

Agenda Item: 7.5.3

Doc-1st-	Spec	CR	R	Phas	Subject	Cat	Ver	Doc-2nd-	Workitem
SP-040248	32.363	001	-	Rel-6	Clarification of return value of getIRPReference and Correction of Distinguished Name (DN) and IDL errors	F	6.0.0	S5-046496	OAM-NIM

3GPP TSG-SA5 (Telecom Management) **S5-046496**
Meeting #38, Bejing, CHINA, 10 - 14 May 2004

CR-Form-v7

CHANGE REQUEST

⌘ **32.363** **CR** **001** ⌘ **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:	⌘	Clarification of return value of getIRPReference and Correction of Distinguished Name (DN) and IDL errors
Source:	⌘	SA5 (olaf.pollakowski@siemens.com)
Work item code:	⌘	OAM-NIM
Date:	⌘	14/05/2004
Category:	⌘	**F**
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:	⌘	The current spec needs clarification regarding the return value of getIRPReference. Also, it contains an IDL error and an error in DN of Annex B
Summary of change:	⌘	• Clarify that the stringified IOR is returned by getIRPReference • Correct return parameter of getIRPOutline • Correct DN example in Annex B
Consequences if not approved:	⌘	Incorrect IDL definitions and naming example not in line with naming conventions

Clauses affected:	⌘	5.2, Annex A, Annex B

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

Other comments:	⌘	

Change in Clause 5.2

5.2 Operation parameter mapping

The EPIRP: IS 3GPP TS 32.362 [6] defines semantics of parameters carried in operations across the EPIRP. The following tables indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

Table 2: Mapping from IS `getIRPOutline` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
iRPVersion	ManagedGenericIRPConstDefs::VersionNumber iRPVersion	M
supportedIRPList	EPIRPSystem::SupportedIRPListType supportedIRPList	M
status	EPIRPSystem::ResultType Exception: GetIRPOutline, InvalidIRPVersion	M

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

IS Operation parameter	SS Method parameter	Qualifier
managerIdentifier	EPIRPSystem::ManagerIdentifierType managerIdentifier	M
systemDn	EPIRPSystem::SystemDNType systemDn	M
iRPId	EPIRPSystem::IRPIdType irpId	M
iRPReference	string iRPReference (stringified IOR)	M
status	EPIRPSystem::ResultType Exception: GetIRPReference, InvalidRequestedParameters	M

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

IS Operation parameter	SS Method parameter	Qualifier
managerIdentifier	EPIRPSystem::ManagerIdentifierType managerIdentifier	M
iRPReference	string iRPReference (stringified IOR)	M
status	EPIRPSystem::ResultType Exception: ReleaseIRPReference, UnknownIRPReference	M

Table 5: Mapping from IS `getIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
versionNumberSet	Return value of type ManagedGenericIRPConstDefs::VersionNumberSet	M
status	Exception: GetEPIRPVersions	M

Table 6: Mapping from IS `getOperationProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
iRPVersion	ManagedGenericIRPConstDefs::VersionNumber iRPVersion	M
operationNameProfile,operationParameterProfile	Return of type ManagedGenericIRPConstDefs::MethodList	M
status	Exception: GetEPIRPOperationsProfile, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table 7: Mapping from IS `getNotificationProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
iRPVersion	ManagedGenericIRPConstDefs::VersionNumber iRPVersion	M
notificationNameProfile,notificationParameterProfile	Return value of type ManagedGenericIRPConstDefs::MethodList	M
status	Exception: GetEPIRPNotificationProfile, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

End of Change in Clause 5.2

Change in Annex A & Annex B

Annex A (normative): IDL specifications

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

```
#ifndef EPIRPSystem_idl
#define EPIRPSystem_idl

#include "NotificationIRPConstDefs.idl"
#include "ManagedGenericIRPConstDefs.idl"
#include "ManagedGenericIRPSystem.idl"

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

/* ## Module: EPIRPSystem
This module implements capabilities of EPIRP.
================================================================
*/
module EPIRPSystem
{
  enum ResultType {OK, Failure};

  typedef string IRPIdType;
  typedef string SystemDNType;
  typedef sequence<string> DNListType;

  /*
   IRPManagementScopeOpt is a type carrying an optional parameter.
   If the boolean is TRUE, then the value is present.
   Otherwise the value is absent.

  */
  union IRPManagementScopeOpt switch (boolean)
  {
    case TRUE: DNListType value;
  };

  /*
  The IRPElement defines the structure to be returned as part of
  getIRPOutline().
  */
  struct IRPElement
  {
    IRPIdType irpId;
    ManagedGenericIRPConstDefs::VersionNumberSet irpVersions;
    IRPManagementScopeOpt irpManagementScope;
  };
```

```
/*
 List of all IRPElement and their associated parameters.
*/
typedef sequence<IRPElement> IRPListType;

struct SupportedIRPListTypeElement
{
  SystemDNType systemDN;
  IRPListType irpList;
};

typedef sequence<SupportedIRPListTypeElement> SupportedIRPListType;

typedef string ManagerIdentifierType;

typedef string IRPDnType;
enum ChangeModeType {REGISTER, DEREGISTER, MODIFY};

/*
Define the parameters specified in
the notifyEpInfoChanges notification.
*/
interface AttributeNameValue
{
   const string IRP_DN = "IRP_DN";
   const string CHANGE_MODE = "CHANGE_MODE";
   const string ADDITIONAL_TEXT = "ADDITIONAL_TEXT";
};


const string ET_IRPINFO_CHANGES = "notifyIrpInfoChanges";

exception InvalidIRPVersion { string reason; };
exception InvalidRequestedParameters { string reason; };
exception UnknownIRPReference { string reason; };

/*
 System 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 GetIRPOutline { string reason; };
exception GetIRPReference { string reason; };
exception ReleaseIRPReference { string reason; };
exception GetEPIRPVersions { string reason; };
exception GetEPIRPOperationsProfile { string reason; };
exception GetEPIRPNotificationProfile { string reason; };

/*
*/
 interface EPIRP
 {
   /**
   * The IRPManager uses this operation to request the EPIRP to
   * return the outline information of the supported IRPs. The EPIRP
   * shall return the outline information of all the IRPs supported by the
   * IRPAgent that contains the EPIRP. The EPIRP may
   * additionally return the outline information of all the IRPs supported
   * by other IRPAgents.
   */
   ResultType    get_IRP_outline(
     in ManagedGenericIRPConstDefs::VersionNumber iRPVersion,
     out SupportedIRPListType supportedIRPList
   )
   raises (GetIRPOutline,InvalidIRPVersion);

   /**
   * The IRPManager uses this operation to request the EPIRP
   * to return a reference for a specific version of a specific IRP.
   * The IRPManager uses this operation to request the EPIRP to return
   * the stringified IOR of the IRP identified by systemDn and irpId.
   */
   ResultType    get_IRP_reference(
     in ManagerIdentifierType managerIdentifier,
     in SystemDNType systemDn,
     in IRPIdType irpId,
     out string iRPReference
   )
```

```
  raises (GetIRPReference,
     InvalidRequestedParameters);

  /**
  * The IRPManager uses this operation to request the IRPAgent to
  * release a specific IRP reference. Whether the IRP reference
  * is really released or not in the IRPAgent is outside the
  * scope of this document.
  */
  ResultType    release_IRP_reference(
    in ManagerIdentifierType managerIdentifier,
    in string iRPReference
  )
  raises (ReleaseIRPReference,
     UnknownIRPReference);

   /**
   * Return the list of all supported EPIRP versions.
   */
   ManagedGenericIRPConstDefs::VersionNumberSet get_EP_IRP_versions (
   )
   raises (GetEPIRPVersions);


   /**
   * Return the list of all supported operations and their supported
   * parameters for a specific EPIRP version.
   */
   ManagedGenericIRPConstDefs::MethodList get_EP_IRP_operations_profile (
      in ManagedGenericIRPConstDefs::VersionNumber iRPVersion
   )
   raises (GetEPIRPOperationsProfile,
           ManagedGenericIRPSystem::OperationNotSupported,
           ManagedGenericIRPSystem::InvalidParameter);

   /**
   * Return the list of all supported notifications and their supported
   * parameters for a specific EPIRP version.
   */
   ManagedGenericIRPConstDefs::MethodList get_EP_IRP_notification_profile
   (
      in ManagedGenericIRPConstDefs::VersionNumber iRPVersion
   )
   raises (GetEPIRPNotificationProfile,
           ManagedGenericIRPSystem::OperationNotSupported,
           ManagedGenericIRPSystem::InvalidParameter);
};


/**
*  Constant definitions for the EPInfoChanges notification
*/
module notifyIRPInfoChanges
{
  const string MANAGED_OBJECT_CLASS =
    NotificationIRPConstDefs::AttributeNameValue::MANAGED_OBJECT_CLASS;
  const string MANAGED_OBJECT_INSTANCE =
    NotificationIRPConstDefs::AttributeNameValue::MANAGED_OBJECT_INSTANCE;
  const string NOTIFICATION_ID =
    NotificationIRPConstDefs::AttributeNameValue::NOTIFICATION_ID;
  const string EVENT_TIME =
    NotificationIRPConstDefs::AttributeNameValue::EVENT_TIME;
  const string SYSTEM_DN =
    NotificationIRPConstDefs::AttributeNameValue::SYSTEM_DN;
  const string EVENT_TYPE = ET_IRPINFO_CHANGES;

  /**
  * This constant defines the name of the iRPDn property,
  * which is transported in the filterable_body fields.
  * The data type for the value of this property
  * is IRPDnType.
  */
  const string IRP_DN = AttributeNameValue::IRP_DN;

  /**
  * This constant defines the name of the changeMode property,
  * which is transported in the filterable_body fields.
  * The data type for the value of this property is ChangeModeType.
```

```
      */
      const string CHANGE_MODE = AttributeNameValue::CHANGE_MODE;

      /**
      * This constant defines the name of the additionalText property,
      * which is transported in the filterable_body fields.
      * The data type for the value of this property is string.
      */
      const string ADDITIONAL_TEXT = AttributeNameValue::ADDITIONAL_TEXT;
      };

};

#endif
```

Annex B (informative): Convention when using INS to fulfill part of EPIRP functions

The implementation of the EPIRP and in particular, the management of CORBA object references within EPIRP, is not a subject matter for 3GPP standardization.

ITU-T SG4 Framework for CORBA-Based Telecommunications Management Network Interfaces (ITU-T Recommendation Q.816.1 [10]) uses OMG Interoperable Naming Service (INS) [11] for the management of CORBA object references. Furthermore, it specifies a convention to name and populate the CORBA object entries within the INS.

This Annex notes that, in the event that an EPIRP implementation uses INS to fulfill part of EPIRP functions, it is advantageous to populate the INS using the ITU-T defined convention.

Convention

The OMG INS CORBA *name component* (in short, called CORBA compound name) has the following IDL definition:

```
// IDL
typedef string Istring;
struct NameComponent {
    Istring id;
    Istring kind;
};
```

`Istring` is a placeholder for a future IDL internationalized string. The id and kind attributes must be composed of characters from the ISO 8859-1 [12] character set, excluding the null character and other non-printable characters. The strings cannot exceed 255 characters. The id attribute cannot be an empty string but the kind attribute can be an empty string.

The CORBA compound name (see `Name` below) is defined as a sequence of name components:

```
// IDL
typedef sequence<NameComponent> Name;
```

The 3GPP defined DN (in short, called DN) of a managed object is represented by the CORBA *compound name*. For example, a DN, quoted from 3GPP TS 32.300 [9], say "`DC=se.companyZ.lmc,Network=9,SubNetwork=1,IRPAgent=1,AlarmIRP.iRPId=2`", shall be represented by a sequence of 6 *name components* where the *id* and *kind* of the first *name component* shall be "se_companyZ_lmc" and "DC" respectively. The CORBA *compound name*, shall be:

index	kind	id
0	"DC"	"se_companyZ_lmc"
1	"Network"	"9"
2	"SubNetwork"	"1"
3	"IRPAgent"	"1"
4	"AlarmIRP.iRPId"	"2"
5	""	"Object"

The CORBA *compound name*, in stringified name form, shall be "se_companyZ_lmc.DC/9.Network/1.SubNetwork/1.IRPAgent/2.AlarmIRP.iRPId/Object".

NOTE 1: DN appears in interactions (e.g. operations, notifications) across the Itf-N.

NOTE 2: The CORBA compound name is used internally with the IRPAgent (and its INS) and does not appear in interactions across the Itf-N.

The use of the last row of the CORBA *compound name*, i.e. kind == "" and id ==”Object”, is in accordance to the convention standardized by ITU-T Recommendation Q.816.1 [10]. According to convention standardized by ITU-T Recommendation Q.816.1 [10], the use of index 0 to 4 inclusive is to indicate the naming context of the object and the

use of index 0 to 5 inclusive is to indicate the object itself.

DN DC component is composed of multiple words separated by separator, i.e. a dot. It is suggested that the applications (e.g. IRPAgent codes) that process CORBA compound name and DN should map the dot, used as separator in DN DC component, with underscore. This mapping is necessary because in the stringified CORBA *compound name*, the dot is used for the separation of *id* and *kind*. This replacement rule also implies that underscore should not be used as character of DC words.

End of Change in Annex A & Annex B

Annex C (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2003	S_21	SP-030425	--	--	Submitted to TSG SA#21 for Information	1.0.0	
Dec 2003	S_22	SP-030636	--	--	Submitted to TSG SA#22 for Approval	2.0.0	6.0.0