ETSI TS 132 533 V8.3.0 (2011-01)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
LTE;
Telecommunication management;
Software management (SwM);
Integration Reference Point (IRP);
Common Object Request Broker Architecture (CORBA)
Solution Set (SS)
(3GPP TS 32.533 version 8.3.0 Release 8)**

Reference

RTS/TSGS-0532533v830

Keywords

LTE, UMTS

ETSI

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

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

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

Intellectual Property Rights

Foreword

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

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

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

Contents

Foreword

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

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

Version x.y.z

where:

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

Introduction

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

32.531:	Software management; Concepts and Integration Reference Point (IRP) Requirements
32.532:	Software management Integration Reference Point (IRP); Information Service (IS)
32.533:	**Software management Integration Reference Point (IRP); Common Object Request Broker Architecture (CORBA) Solution Set (SS)**

1 Scope

The present document is the "CORBA Solution Set" of Software Management IRP for the IRP whose semantics is specified in Software Management IRP Information Service (3GPP TS 32.532 [7]).

This Solution Set specification is related to 3GPP TS 32.532 [7] V8.1.X.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1]	3GPP TR 21.905: "Vocabulary for 3GPP Specifications".
[2]	3GPP TS 32.101: "Telecommunication management; Principles and high level requirements".
[3]	3GPP TS 32.102: "Telecommunication management; Architecture".
[4]	3GPP TS 21905: "Vocabulary for 3GPP Specifications".
[5]	3GPP TR 32.816: "Telecommunication management; Study on Management of Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC)".
[6]	3GPP TS 32.531: "Telecommunication management; Software management; Concepts and Integration Reference Point (IRP) Requirements".
[7]	3GPP TS 32.532: "Telecommunication management; Software management Integration Reference Point (IRP); Information Service (IS)".
[8]	OMG TC Document telecom/98-11-01: "OMG Notification Service". http://www.omg.org/technology/documents/

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 32.101 [2], TS 32.102 [3] and TR 21.905 [4] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TS 32.532 [7], TS 32.531 [6], TS 32.101 [1], TS 32.102 [2] and TS 21.905 [4], in that order.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [4], TS 32.sco [6] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TS 32.532 [7], TS 32.531 [6], TS 32.101 [1], TS 32.102 [2] and TS 21.905 [4], in that order..

4 Architectural Features

The overall architectural feature of Software Management IRP is specified in 3GPP TS 32.532 [7].

5 Mapping

5.1 Operation and Notification mapping

Software Management IRP: IS 3GPP TS (see 3GPP TS 32.532 [7]) defines semantics of operations and notifications visible across the Itf-N. Table 5.1.1 indicates mapping of these operations and notifications to their equivalents defined in this SS.

Table 5.1.1: Mapping from IS Notification/Operation to SS equivalents

IS Operation/ notification Software Management IRP: IS 3GPP TS 32.532[7]	SS Method	Qualifier
listSwMCapabilities	listSwMCapabilities	M
listSwMProfiles	listSwMProfiles	M
createSwMProfile	createSwMProfile	M
deleteSwMProfile	deleteSwMProfile	M
listSwMProcesses	listSwMProcesses	M
resumeSwMProcess	resumeSwMProcess	M
swFallback	swFallback	M
terminateSwMProcess	terminateSwMProcess	M
changeSwMProfile	changeSwMProfile	O
notifySwMProfileCreation	notifySwMProfileCreation	M
notifySwMProfileDeletion	notifySwMProfileDeletion	M
notifySwMProcessCreation	notifySwMProcessCreation	M
notifySwMProcessStage	notifySwMProcessStage	M
notifySwMProcessDeletion	notifySwMProcessDeletion	M
notifyNewSwAvailability	notifyNewSwAvailability	M
notifySwMProfileChange	notifySwMProfileChange	O

5.2 Operation parameter mapping

Reference 3GPP TS 32.532 [6] defines semantics of parameters carried in operations across the Itf-N. The following set of tables indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

Table 5.2-1: Mapping from IS `listSwMCapabilities` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
nEInformation	SwMIRPConstDefs::NEInformation	M
capabilitiesList	SwMIRPConstDefs::SwMCapabilitiesList	M
result	Exceptions: SwMIRPConstDefs::ListSwMCapabilities, GenericIRPManagementSystem::ParameterNotSupported, GenericIRPManagementSystem::InvalidParameter, GenericIRPManagementSystem::ValueNotSupported, GenericIRPManagementSystem::OperationNotSupported	M

Table 5.2-2: Mapping from IS `listSwMProfiles` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
nEInformation	SwMIRPConstDefs::NEInformation	M
profileList	SwMIRPConstDefs::SwMProfileList	M
result	Exceptions: SwMIRPConstDefs::ListSwMProfile, GenericIRPManagementSystem::ParameterNotSupported, GenericIRPManagementSystem::InvalidParameter, GenericIRPManagementSystem::ValueNotSupported, GenericIRPManagementSystem::OperationNotSupported	M

Table 5.2-3: Mapping from IS `createSwMProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
id	SwMIRPConstDefs::IdOpt	O
nEInformation	SwMIRPConstDefs::NEInformation	M
swVersionToBeInstalled	SwMIRPConstDefs::SwVersionToBeInstalledConditional	CM
stepsAndSelectedStopPointList	SwMIRPConstDefs::StepsAndSelectedStopPointList	M
selectedFinalAdministrativeState	SwMIRPConstDefs::SelectedFinalAdministrativeState	M
result	Exceptions: SwMIRPConstDefs::CreateSwMProfile, GenericIRPManagementSystem::ParameterNotSupported, GenericIRPManagementSystem::InvalidParameter, GenericIRPManagementSystem::ValueNotSupported, GenericIRPManagementSystem::OperationNotSupported	M

Table 5.2-4: Mapping from IS `deleteSwMProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
id	SwMIRPConstDefs::Id	M
result	Exceptions: SwMIRPConstDefs::DeleteSwMProfile, GenericIRPManagementSystem::ParameterNotSupported, GenericIRPManagementSystem::InvalidParameter, GenericIRPManagementSystem::ValueNotSupported, GenericIRPManagementSystem::OperationNotSupported	M

Table 5.2-5: Mapping from IS `listSwMProcesses` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`nEIdentification`	`SwMIRPConstDefs::NEIdentificationOpt`	O
`processList`	`SwMIRPConstDefs::ProcessList`	M

`result`	`Exceptions:`
	`SwMIRPConstDefs::ListSwMProcesses,`
	`GenericIRPManagementSystem::ParameterNotSupported,`
	`GenericIRPManagementSystem::InvalidParameter,`
	`GenericIRPManagementSystem::ValueNotSupported,`
	`GenericIRPManagementSystem::OperationNotSupported`
		M

Table 5.2-6: Mapping from IS `resumeSwMProcess` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`id`	`SwMIRPConstDefs::Id`	M
`startStepName`	`SwMIRPConstDefs::NameOfStep`	M

`result`	`Exceptions:`
	`SwMIRPConstDefs::ResumeSwMProcess,`
	`GenericIRPManagementSystem::ParameterNotSupported,`
	`GenericIRPManagementSystem::InvalidParameter,`
	`GenericIRPManagementSystem::ValueNotSupported,`
	`GenericIRPManagementSystem::OperationNotSupported`
		M

Table 5.2-7: Mapping from IS `swFallback` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`filter`	`SwMIRPConstDefs::Filter`	M
`nEList`	`SwMIRPConstDefs::NEList`	M

`result`	`Exceptions:`
	`SwMIRPConstDefs::SwFallback,`
	`GenericIRPManagementSystem::ParameterNotSupported,`
	`GenericIRPManagementSystem::InvalidParameter,`
	`GenericIRPManagementSystem::ValueNotSupported,`
	`GenericIRPManagementSystem::OperationNotSupported`
		M

Table 5.2-8: Mapping from IS `terminateSwMProcess` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`id`	`SwMIRPConstDefs::Id`	M

`result`	`Exceptions:`
	`SwMIRPConstDefs::TerminateSwMProcess,`
	`GenericIRPManagementSystem::ParameterNotSupported,`
	`GenericIRPManagementSystem::InvalidParameter,`
	`GenericIRPManagementSystem::ValueNotSupported,`
	`GenericIRPManagementSystem::OperationNotSupported`
		M

Table 5.2-9: Mapping from IS `changeSwMProfile` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`id`	`SwMIRPConstDefs::Id`	M
`nEInformation`	`SwMIRPConstDefs::NEInformation`	M
`swVersionToBeInstalled`	`SwMIRPConstDefs::SwVersionToBeInstalledConditional`	CM
`stepsAndSelectedStopPointList`	`SwMIRPConstDefs::StepsAndSelectedStopPointList`	M
`selectedFinalAdministrativeState`	`SwMIRPConstDefs::SelectedFinalAdministrativeState`	M
`versionNumber`	`SwMIRPConstDefs::VersionNumber`	M
`conflictingProfileId`	`SwMIRPConstDefs::ConflictingProfileIdConditional`	C

`result`	`Exceptions:`
	`SwMIRPConstDefs::CreateSwMProfile,`
	`GenericIRPManagementSystem::ParameterNotSupported,`
	`GenericIRPManagementSystem::InvalidParameter,`
	`GenericIRPManagementSystem::ValueNotSupported,`
	`GenericIRPManagementSystem::OperationNotSupported`
		M

5.3 Notification parameter mapping

Reference 3GPP TS 32.532 [7] defines semantics of parameters carried in notifications. The following tables indicate the mapping of these parameters to their SS equivalents."

The following tables indicate the mapping of these parameters to their OMG CORBA Structured Event (defined in OMG Notification Service [8]) equivalents. The composition of OMG Structured Event, as defined in the OMG Notification Service [8], is:

```
Header
      Fixed Header
           domain_name
           type_name
           event_name
      Variable Header
Body
      filterable_body_fields
      remaining_body
```

The following tables list all OMG Structured Event attributes in the second column. The first column identifies the Software Management IRP: IS [7] defined notification parameters.

Table 5.3.1: Mapping for `notifySwMProfileCreation`

IS Parameters	<SS> Parameters	Qualifier	Comment
`id`	`SwMIRPConstDefs::Id`	M	
`versionNumber`	`SwMIRPConstDefs::VersionNumber`	M	
`nEInformation`	`SwMIRPConstDefs::NEInformation`	M	
`swVersionToBeInstalled`	`SwMIRPConstDefs::SwVersionToBeInstalledConditional`	CM	
`stepsAndSelectedStopPointList`	`SwMIRPConstDefs::StepsAndSelectedStopPointList`	M	
`selectedFinalAdministrativeState`	`SwMIRPConstDefs::SelectedFinalAdministrativeState`	M	

Table 5.3.2: Mapping for `notifySwMProfileDeletion`

IS Parameters	<SS> Parameters	Qualifier	Comment
`id`	`SwMIRPConstDefs::Id`	M	

Table 5.3.3: Mapping for `notifySwMProcessCreation`

IS Parameters	<SS> Parameters	Qualifier	Comment
`id`	`SwMIRPConstDefs::Id`	M	
`nEIdentification`	`SwMIRPConstDefs::NEIdentification`	M	
`profileId`	`SwMIRPConstDefs::ProfileId`	M	
`matchingNEInformation`	`SwMIRPConstDefs::MatchingNEInformation`	M	
`stepInfoList`	`SwMIRPConstDefs::StepInfoList`	M	

Table 5.3.4: Mapping for `notifySwMProcessStage`

IS Parameters	<SS> Parameters	Qualifier	Comment
`id`	`SwMIRPConstDefs::Id`	M	
`stepInfoList`	`SwMIRPConstDefs::StepInfoList`	M	

Table 5.3.5: Mapping for `notifySwMProcessDeletion`

IS Parameters	<SS> Parameters	Qualifier	Comment
`id`	`SwMIRPConstDefs::Id`	M	
`triggerForDeletion`	`SwMIRPConstDefs::TriggerForDeletion`	M	
`additionalInformation`	`SwMIRPConstDefs::AdditionalInformationOptional`	O	

Table 5.3.6: Mapping for `notifyNewSwAvailability`

IS Parameters	<SS> Parameters	Qualifier	Comment
`nEandSWversion`	`SwMIRPConstDefs::NEandSWversion`	M	

Table 5.3.7: Mapping for `notifySwMProfileChange`

IS Parameters	<SS> Parameters	Qualifier	Comment
`id`	`SwMIRPConstDefs::Id`	M	
`versionNumber`	`SwMIRPConstDefs::VersionNumber`	M	
`nEInformation`	`SwMIRPConstDefs::NEInformation`	M	
`swVersionToBeInstalled`	`SwMIRPConstDefs::SwVersionToBeInstalledConditional`	CM	
`stepsAndSelectedStopPointList`	`SwMIRPConstDefs::StepsAndSelectedStopPointList`	M	
`selectedFinalAdministrativeState`	`SwMIRPConstDefs::SelectedFinalAdministrativeState`	M	

Annex A (normative): IDL specifications

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

```
// File: SwMIRPConstDefs.idl
#ifndef _SWM_IRP_CONST_DEFS_IDL_
#define _SWM_IRP_CONST_DEFS_IDL_


#include <KernelCmConstDefs.idl>
#include <NotificationIRPConstDefs.idl>

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


/* ## Module: SwMIRPConstDefs */

module SwMIRPConstDefs
{

/*****************************************************************/
/* definition of types used in operations for Software Management : */
/*****************************************************************/

/* types used in several operations: */

   enum Result { SUCCESS, PARTLY_SUCCESSFUL, FAILURE, NE_INFORMATION_INTERSECTION,
STEPNAME_DOES_NOT_MATCH };

   typedef KernelCmConstDefs::DN Id;

   /*
   IdOpt is a type carrying an optional parameter.
   If the boolean is TRUE, then the value is present.
   Otherwise the value is absent.
   */
   union IdOpt switch (boolean)
   {
      case TRUE: KernelCmConstDefs::DN value;
   };


   /*
   ConflictingProfileIdConditional is a type carrying a conditional parameter.
   The boolean shall be TRUE, if the condition described in TS 32.532 or 32.502 is fulfilled.
   In this case the value is present. Otherwise the value is absent.
   */
   union ConflictingProfileIdConditional switch (boolean)
   {
      case TRUE: Id value;
   };


   typedef string NEInformation;

   typedef NEInformation MatchingNEInformation;

   typedef string AdditionalInformation;

   /*
   AdditionalInformationOpt is a type carrying a optional parameter.
   The boolean shall be TRUE, if the condition described in TS 32.532 is fulfilled.
   In this case the value is present. Otherwise the value is absent.
   */
   union AdditionalInformationOpt switch (boolean)
   {
      case TRUE: AdditionalInformation value;
   };
   typedef KernelCmConstDefs::DN NEIdentification;

   /*
   NEIdentificationOpt is a type carrying an optional parameter.
   If the boolean is TRUE, then the value is present.
   Otherwise the value is absent.
   */
   union NEIdentificationOpt switch (boolean)
   {
      case TRUE: NEIdentification value;
   };


   enum SwFallbackStatus { FALLBACK_SUCCESSFUL, FALLBACK_UNSUCCESSFUL };

   struct NeListEntry
      {
      SwMIRPConstDefs::NEIdentification nEIdentification;
      SwMIRPConstDefs::SwFallbackStatus swFallbackStatus;
      };

   typedef sequence<NeListEntry> NeList;



   typedef string SwVersionToBeInstalled;


   /*
   SwVersionToBeInstalledConditional is a type carrying a conditional parameter.
   The boolean shall be TRUE, if the condition described in TS 32.532 is fulfilled.
   In this case the value is present. Otherwise the value may be absent.
   */
   union SwVersionToBeInstalledConditional switch (boolean)
   {
      case TRUE: SwVersionToBeInstalled value;
   };


   enum NameOfStep
   {
   SW_DOWNLOAD,
   SW_INSTALLATION,
   SW_ACTIVATION,
   PREPARE_BASIC_CONFIGURATION_AND_OAMLINK,
   RETRIEVE_CONFIGURATION_DATA,
   SETUP_PRECONFIGURED_SIGNALLING_LINKS,
   SET_FINAL_STATE_OF_NE
   };
   /*
   The following values are not used in SWM IRP, but only in inheriting Self-Conf IRP TS 32.502:
   PREPARE_BASIC_CONFIGURATION_AND_OAMLINK,
   RETRIEVE_CONFIGURATION_DATA,
   SETUP_PRECONFIGURED_SIGNALLING_LINKS,
   SET_FINAL_STATE_OF_NE
   */

   */


   typedef unsigned short SequenceNumberInProcess;

   enum StopPointCanBeSetBeforeThisStep { YES, NO };

   struct StepsAndOfferedStopPointListEntry
      {
      SwMIRPConstDefs::NameOfStep nameOfStep;
      SwMIRPConstDefs::SequenceNumberInProcess sequenceNumberInProcess;
      SwMIRPConstDefs::StopPointCanBeSetBeforeThisStep stopPointCanBeSetBeforeThisStep
;
      };

   typedef sequence<StepsAndOfferedStopPointListEntry> StepsAndOfferedStopPointList;


   enum StopPointSetIndication { STOP_POINT_IS_SET_BEFORE_THIS_STEP, STOP_POINT_IS_NOT_SET };
   struct StepAndSelectedStopPointListEntry
      {
      SwMIRPConstDefs::NameOfStep nameOfStep;
      SwMIRPConstDefs::SequenceNumberInProcess sequenceNumberInProcess;
      SwMIRPConstDefs::StopPointSetIndication stopPointSetIndication
;
      };

   typedef sequence<StepAndSelectedStopPointListEntry> StepAndSelectedStopPointList;


   enum StepProgress { NOT_YET_STARTED, RUNNING, COMPLETED, AWAITING_RESUME, FAILURE, TERMINATED };

   struct StepInfoListEntry
      {
      SwMIRPConstDefs::NameOfStep nameOfStep;
      SwMIRPConstDefs::SequenceNumberInProcess sequenceNumberInProcess;
      SwMIRPConstDefs::StopPointSetIndication stopPointSetIndication;
      SwMIRPConstDefs::StepProgress stepProgress;
      };

   typedef sequence<StepInfoListEntry> StepInfoList;


   struct SwMProcessListEntry
      {
      SwMIRPConstDefs::Id id;
      SwMIRPConstDefs::NEIdentification nEIdentification;
      SwMIRPConstDefs::StepInfoList stepInfoList;
      };

   typedef sequence<SwMProcessListEntry> SwMProcessList;


   enum FinalAdministrativeStateValue { LOCKED, UNLOCKED, DETERMINED_BY_CONFIGURATION_DATA };

   typedef FinalAdministrativeStateValue OfferedFinalAdministrativeStateValue

   typedef sequence<OfferedFinalAdministrativeStateValue>
OfferedFinalAdministrativeStateInformation;


   typedef FinalAdministrativeStateValue SelectedFinalAdministrativeStateValue


   typedef sequence<SwVersionToBeInstalled> SwVersionToBeInstalledOfferList;


   /*
   SwVersionToBeInstalledOfferListConditional is a type carrying a conditional parameter.
   The boolean shall be TRUE, if the condition described in TS 32.532 is fulfilled.
   In this case the value is present. Otherwise the value may be absent.
   */
   union SwVersionToBeInstalledOfferListConditional switch (boolean)
   {
      case TRUE: SwVersionToBeInstalledOfferList value;
   };


   typedef unsigned short VersionNumber;


   struct ProfileId
      {
      SwMIRPConstDefs::Id id;
      SwMIRPConstDefs::VersionNumber versionNumber;
      };


   struct SwMCapability
      {
      SwMIRPConstDefs::Id id;
      SwMIRPConstDefs::NEInformation nEInformation;
      SwMIRPConstDefs::StepsAndOfferedStopPointList stepsAndOfferedStopPointList;
      SwMIRPConstDefs::OfferedFinalAdministrativeStateInformation
offeredFinalAdministrativeStateInformation;
      SwMIRPConstDefs::SwVersionToBeInstalledOfferListConditional swVersionToBeInstalledOfferList;
      };
   typedef sequence<SwMCapability> SwMCapabilitiesList;


   struct SwMProfile
      {
      SwMIRPConstDefs::Id id;
      SwMIRPConstDefs::VersionNumber versionNumber;
      SwMIRPConstDefs::NEInformation nEInformation;
      SwMIRPConstDefs::StepsAndSelectedStopPointList stepsAndSelectedStopPointList;
      SwMIRPConstDefs::SelectedFinalAdministrativeStateValue selectedFinalAdministrativeState;
      SwMIRPConstDefs::SwVersionToBeInstalledConditional swVersionToBeInstalled;
      };

   typedef sequence<SwMProfile> SwMProfilesList;


/*******************************************************************/
/* definition of types in notifications for software management : */
/*******************************************************************/

   typedef string Filter;

   typedef string NeAndSWVersion;

   enum TriggerForDeletion { IRP_AGENT_TERMINATION, IRP_MANAGER_TERMINATION,
AUTOMATED_SWM_SUCCESFULLY_CONCLUDED, SELF_CONFIGURATION_SUCCESFULLY_CONCLUDED };
   /*
   The following values are not used in SWM IRP, but only in inheriting Self-Conf IRP TS 32.502:
   SELF_CONFIGURATION_SUCCESFULLY_CONCLUDED
   */

interface AttributeNameValue
   {
   const string ID = "ID";
   const string VERSION_NUMBER = "VERSION_NUMBER";
   const string NE_INFORMATION = "NE_INFORMATION";
   const string SW_VERSION_TO_BE_INSTALLED = "SW_VERSION_TO_BE_INSTALLED";
   const string STEPS_AND_SELECTED_STOP_POINT_LIST = "STEPS_AND_SELECTED_STOP_POINT_LIST";
   const string SELECTED_FINAL_ADMINISTRATIVE_STATE = "SELECTED_FINAL_ADMINISTRATIVE_STATE";
   const string NE_IDENTIFICATION = "NE_IDENTIFICATION";
   const string PROFILE_ID = "PROFILE_ID";
   const string MATCHING_NE_INFORMATION = "MATCHING_NE_INFORMATION";
   const string STEP_INFO_LIST = "STEP_INFO_LIST";
   const string NE_AND_SW_VERSION = "NE_AND_SW_VERSION";
   const string TRIGGER_FOR_DELETION = "TRIGGER_FOR_DELETION";
   const string ADDITIONAL_INFORMATION = "ADDITIONAL_INFORMATION";
   };



};

#endif // _SWM_IRP_CONST_DEFS_IDL_
```

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

```
//File: SwMIRPSystem.idl
#ifndef _SWM_IRP_SYSTEM_IDL_
#define _SWM_IRP_SYSTEM_IDL_

#include <SwMIRPConstDefs.idl>
#include <GenericIRPManagementSystem.idl>

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

/* ## Module: SwMIRPSystem */

module SwMIRPSystem
{

   /*
   If the system fails to complete an operation, then it can provide a reason
   to qualify the exception. The semantics carried in this reason are outside
   the scope of the present document.
   */
   exception ListSwMCapabilities { string reason; };
   exception ListSwMProfiles { string reason; };
   exception CreateSwMProfile { string reason; };
   exception DeleteSwMProfile { string reason; };
   exception ListSwMProcesses { string reason; };
   exception ResumeSwMProcess { string reason; };
   exception SwFallback { string reason; };
   exception TerminateSwMProcess { string reason; };
   exception ChangeSwMProfile { string reason; };


   interface SwMIRPOperations_1
   {
      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result listSwMCapabilities
         (
         in SwMIRPConstDefs::NEInformation nEInformation,
         out SwMIRPConstDefs::SwMCapabilitiesList capabilitiesList
         )
      raises
         (
         ListSwMCapabilities,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result listSwMProfiles
         (
         in SwMIRPConstDefs::NEInformation nEInformation,
         out SwMIRPConstDefs::SwMProfileList profileList
         )
      raises
         (
         ListSwMProfiles,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result createSWMProfile
         (
         in SwMIRPConstDefs::IdOpt id,
         in SwMIRPConstDefs::NEInformation nEInformation,
         in SwMIRPConstDefs::SwVersionToBeInstalledConditional swVersionToBeInstalled,
         in SwMIRPConstDefs::StepsAndSelectedStopPointList stepsAndSelectedStopPointList,
         in SwMIRPConstDefs::SelectedFinalAdministrativeState selectedFinalAdministrativeState
         )
      raises
         (
         CreateSWMProfile,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result deleteSWMProfile
         (
         in SwMIRPConstDefs::Id id
         )
      raises
         (
         DeleteSWMProfile,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result listSwMProcesses
         (
         in SwMIRPConstDefs::NEIdentificationOpt nEIdentification,
         out SwMIRPConstDefs::ProcessList processList
         )
      raises
         (
         ListSwMProcesses,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result resumeSwMProcess
         (
         in SwMIRPConstDefs::Id id,
         in SwMIRPConstDefs::NameOfStep startStepName
         )
      raises
         (
         ResumeSwMProcess,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result swFallback
         (
         in SwMIRPConstDefs::Filter filter,
         out SwMIRPConstDefs::NEList nEList
         )
      raises
         (
         SwFallback,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );


      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result terminateSwMProcess
         (
         in SwMIRPConstDefs::Id id
         )
      raises
         (
         TerminateSwMProcess,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );

   };


   interface SwMIRPOperations_2
   {
      /* for the purpose of this operation see 3GPP TS 32.532 */
      SwMIRPConstDefs::Result changeSWMProfile
         (
         in SwMIRPConstDefs::Id id,
         in SwMIRPConstDefs::NEInformation nEInformation,
         in SwMIRPConstDefs::SwVersionToBeInstalledConditional swVersionToBeInstalled,
         in SwMIRPConstDefs::StepsAndSelectedStopPointList stepsAndSelectedStopPointList,
         in SwMIRPConstDefs::SelectedFinalAdministrativeState selectedFinalAdministrativeState
         out SwMIRPConstDefs::VersionNumber versionNumber
         out SwMIRPConstDefs::ConflictingProfileIdConditional conflictingProfileId
         )
      raises
         (
         ChangeSWMProfile,
         GenericIRPManagementSystem::ParameterNotSupported,
         GenericIRPManagementSystem::InvalidParameter,
         GenericIRPManagementSystem::ValueNotSupported,
         GenericIRPManagementSystem::OperationNotSupported
         );
   };


};


#endif // _SWM_IRP_SYSTEM_IDL_
```

A.3 IDL specification (file name "SwMIRPNotifications.idl")

```
//File: SwMIRPNotifications.idl
#ifndef _SWM_IRP_NOTIFICATIONS_IDL_
#define _SWM_IRP_NOTIFICATIONS_IDL_

#include <SwMIRPConstDefs.idl>
#include <NotificationIRPNotifications.idl>

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

/* ## Module: SwMIRPNotifications
This contains the specification of notifications of Software Management.
===========================================================================
*/
module SwMIRPNotifications
{

   /* Constant definitions for the notifySwMProfileCreation notification */

   interface NotifySwMProfileCreation: NotificationIRPNotifications::Notify
   {
      const string EVENT_TYPE = "notifySwMProfileCreation";

      /**
      * This constant defines the name of the Id property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::Id.
      */
      const string ID =
         SwMIRPConstDefs::AttributeNameValue::ID;

      /**
      * This constant defines the name of the VersionNumber property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::VersionNumber.
      */
      const string VERSION_NUMBER =
         SwMIRPConstDefs::AttributeNameValue::VERSION_NUMBER;

      /**
      * This constant defines the name of the NEInformation property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::NEInformation.
      */
      const string NE_INFORMATION =
         SwMIRPConstDefs::AttributeNameValue::NE_INFORMATION;

      /**
      * This constant defines the name of the SwVersionToBeInstalled property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::SwVersionToBeInstalledConditional.
      */
      const string SW_VERSION_TO_BE_INSTALLED =
         SwMIRPConstDefs::AttributeNameValue::SW_VERSION_TO_BE_INSTALLED;


      /**
      * This constant defines the name of the StepsAndSelectedStopPointList property,
      * which is transported in the remaining_body.
      * The data type for the value of this property is
      * SwMIRPConstDefs::StepsAndSelectedStopPointList.
      */
      const string STEPS_AND_SELECTED_STOP_POINT_LIST =
         SwMIRPConstDefs::AttributeNameValue::STEPS_AND_SELECTED_STOP_POINT_LIST;

      /**
      * This constant defines the name of the SelectedFinalAdministrativeState property,
      * which is transported in the remaining_body.
      * The data type for the value of this property is
      * SwMIRPConstDefs::SelectedFinalAdministrativeState.
      */
      const string SELECTED_FINAL_ADMINISTRATIVE_STATE =
         SwMIRPConstDefs::AttributeNameValue::SELECTED_FINAL_ADMINISTRATIVE_STATE;

   };


   /* Constant definitions for the notifySwMProfileDeletion notification */

   interface NotifySwMProfileDeletion: NotificationIRPNotifications::Notify
   {
      const string EVENT_TYPE = "notifySwMProfileDeletion";

      /**
      * This constant defines the name of the Id property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::Id.
      */
      const string ID =
         SwMIRPConstDefs::AttributeNameValue::ID;

   };


   /* Constant definitions for the notifySwMProcessCreation notification */

   interface NotifySwMProcessCreation: NotificationIRPNotifications::Notify
   {
      const string EVENT_TYPE = "notifySwMProcessCreation";

      /**
      * This constant defines the name of the Id property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::Id.
      */
      const string ID =
         SwMIRPConstDefs::AttributeNameValue::ID;

      /**
      * This constant defines the name of the NEIdentification property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::NEIdentification.
      */
      const string NE_IDENTIFICATION =
         SwMIRPConstDefs::AttributeNameValue::NE_IDENTIFICATION;

      /**
      * This constant defines the name of the ProfileId property,
      * which is transported in the remaining_body.
      * The data type for the value of this property is
      * SwMIRPConstDefs::ProfileId.
      */
      const string PROFILE_ID =
         SwMIRPConstDefs::AttributeNameValue::PROFILE_ID;

      /**
      * This constant defines the name of the MatchingNEInformation property,
      * which is transported in the remaining_body.
      * The data type for the value of this property is
      * SwMIRPConstDefs::MatchingNEInformation.
      */
      const string MATCHING_NE_INFORMATION =
         SwMIRPConstDefs::AttributeNameValue::MATCHING_NE_INFORMATION;

      /**
      * This constant defines the name of the StepInfoList property,
      * which is transported in the remaining_body.
      * The data type for the value of this property is
      * SwMIRPConstDefs::StepInfoList.
      */
      const string STEP_INFO_LIST =
         SwMIRPConstDefs::AttributeNameValue::STEP_INFO_LIST;
```

```
};


/* Constant definitions for the notifySwMProcessStage notification */

interface NotifySwMProcessStage: NotificationIRPNotifications::Notify
{
   const string EVENT_TYPE = "notifySwMProcessStage";

   /**
   * This constant defines the name of the Id property,
   * which is transported in the filterable_body_fields.
   * The data type for the value of this property is
   * SwMIRPConstDefs::Id.
   */
   const string ID =
      SwMIRPConstDefs::AttributeNameValue::ID;

   /**
   * This constant defines the name of the StepInfoList property,
   * which is transported in the remaining_body.
   * The data type for the value of this property is
   * SwMIRPConstDefs::StepInfoList.
   */
   const string STEP_INFO_LIST =
      SwMIRPConstDefs::AttributeNameValue::STEP_INFO_LIST;

};


/* Constant definitions for the notifySwMProcessDeletion notification */

interface NotifySwMProcessDeletion: NotificationIRPNotifications::Notify
{
   const string EVENT_TYPE = "notifySwMProcessDeletion";

   /**
   * This constant defines the name of the Id property,
   * which is transported in the filterable_body_fields.
   * The data type for the value of this property is
   * SwMIRPConstDefs::Id.
   */
   const string ID =
      SwMIRPConstDefs::AttributeNameValue::ID;

   /**
   * This constant defines the name of the TriggerForDeletion property,
   * which is transported in the remaining_body.
   * The data type for the value of this property is
   * SwMIRPConstDefs::TriggerForDeletion.
   */
   const string ID =
      SwMIRPConstDefs::AttributeNameValue::TRIGGER_FOR_DELETION;

   /**
   * This constant defines the name of the AdditionalInformation property,
   * which is transported in the remaining_body.
   * The data type for the value of this property is
   * SwMIRPConstDefs::AdditionalInformationOptional.
   */
   const string ID =
      SwMIRPConstDefs::AttributeNameValue::ADDITIONAL_INFORMATION;

};


/* Constant definitions for the notifyNewSwAvailability notification */

interface NotifyNewSwAvailability: NotificationIRPNotifications::Notify
{
   const string EVENT_TYPE = "notifyNewSwAvailability";

   /**
   * This constant defines the name of the NEandSWversion property,
   * which is transported in the filterable_body_fields.
   * The data type for the value of this property is
   * SwMIRPConstDefs::NEandSWversion.
```

```
      */
      const string NE_AND_SW_VERSION =
         SwMIRPConstDefs::AttributeNameValue::NE_AND_SW_VERSION;

   };


   /* Constant definitions for the notifySwMProfileChange notification */

   interface NotifySwMProfileChange: NotificationIRPNotifications::Notify
   {
      const string EVENT_TYPE = "notifySwMProfileChange";

      /**
      * This constant defines the name of the Id property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::Id.
      */
      const string ID =
         SwMIRPConstDefs::AttributeNameValue::ID;

      /**
      * This constant defines the name of the VersionNumber property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::VersionNumber.
      */
      const string VERSION_NUMBER =
         SwMIRPConstDefs::AttributeNameValue::VERSION_NUMBER;

      /**
      * This constant defines the name of the NEInformation property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::NEInformation.
      */
      const string NE_INFORMATION =
         SwMIRPConstDefs::AttributeNameValue::NE_INFORMATION;

      /**
      * This constant defines the name of the SwVersionToBeInstalled property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::SwVersionToBeInstalledConditional.
      */
      const string SW_VERSION_TO_BE_INSTALLED =
         SwMIRPConstDefs::AttributeNameValue::SW_VERSION_TO_BE_INSTALLED;

      /**
      * This constant defines the name of the StepsAndSelectedStopPointList property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::StepsAndSelectedStopPointList.
      */
      const string STEPS_AND_SELECTED_STOP_POINT_LIST =
         SwMIRPConstDefs::AttributeNameValue::STEPS_AND_SELECTED_STOP_POINT_LIST;

      /**
      * This constant defines the name of the SelectedFinalAdministrativeState property,
      * which is transported in the filterable_body_fields.
      * The data type for the value of this property is
      * SwMIRPConstDefs::SelectedFinalAdministrativeState.
      */
      const string SELECTED_FINAL_ADMINISTRATIVE_STATE =
         SwMIRPConstDefs::AttributeNameValue::SELECTED_FINAL_ADMINISTRATIVE_STATE;

   };


};

#endif // _SWM_IRP_NOTIFICATIONS_IDL_
```

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2008-12	SP-42	SP-080718	--	--	Submitted to SA#42 for information and approval	1.0.0	8.0.0
2009-06	SP-44	SP-090408	001	--	Add missing start step parameter for resume operation	8.0.0	8.1.0
2009-12	SP-46	SP-090718	011	--	Remove inconsistent notification parameter	8.1.0	8.2.0
2010-12	SP-50	SP-090831	016	--	Align support qualifier of notifyNewSwAvailability with 32.532	8.2.0	8.3.0

History

Document history		
V8.0.0	February 2009	Publication
V8.1.0	July 2009	Publication
V8.2.0	February 2010	Publication
V8.3.0	January 2011	Publication