ETSI TS 132 613 V8.1.0 (2010-01)

Technical Specification

Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Telecommunication management; Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS) (3GPP TS 32.613 version 8.1.0 Release 8)

Reference

RTS/TSGS-0532613v810

Keywords

GSM, 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:

32.611:	"Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Requirements".
32.612:	"Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Information Service (IS)".
32.613:	**"Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS)".**
32.615:	"Configuration Management (CM); Bulk CM Integration Reference Point (IRP): eXtensible Markup Language (XML) file format definition".
32.617:	"Configuration Management (CM); Bulk CM Integration Reference Point (IRP): SOAP Solution Set (SS)".

Configuration Management (CM), in general, provides the operator with the ability to assure correct and effective operation of the 3G network as it evolves. CM actions have the objective to control and monitor the actual configuration on the Network Element (NEs) and Network Resources (NRs), and they may be initiated by the operator or functions in the Operations Systems (OSs) or NEs.

CM actions may be requested as part of an implementation programme (e.g. additions and deletions), as part of an optimisation programme (e.g. modifications), and to maintain the overall Quality of Service. The CM actions are initiated either as a single action on a NE of the 3G network or as part of a complex procedure involving actions on many NEs.

1 Scope

The purpose of this *Bulk CM IRP: CORBA Solution Set* is to define the mapping of the IRP Information Service 3GPP TS 32.612 [3] to the protocol specific details necessary for implementation of this IRP in a CORBA/IDL environment.

This Solution Set specification is related to 3GPP TS 32.612 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 TS 32.101: "Telecommunication management; Principles and high level requirements".

[2] 3GPP TS 32.102: "Telecommunication management; Architecture".

[3] 3GPP TS 32.612: "Telecommunication management; Configuration Management (CM); Bulk CM Integration Reference Point (IRP); Information Service (IS)".

[4] 3GPP TS 32.622: "Telecommunication management; Configuration Management (CM); Generic network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[5] 3GPP TS 32.300: "Telecommunication management; Configuration Management (CM); Name convention for Managed Objects".

[6] OMG Notification Service, Version 1.0.

[7] OMG CORBA services: Common Object Services Specification, Update: November 22, 1996.

[8] The Common Object Request Broker: Architecture and Specification (for specification of valid version, see [1]).

[9] 3GPP TS 32.303: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS)".

[10] 3GPP TS 32.111-3: "Telecommunication management; Fault Management; Part 3: Alarm Integration Reference Point (IRP): Common Object Request Broker Architecture (CORBA) Solution Set (SS)".

[11] 3GPP TS 32.642: "Telecommunication management; Configuration Management (CM); UTRAN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[12] 3GPP TS 32.652: "Telecommunication management; Configuration Management (CM); GERAN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[13] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[14] 3GPP TS 32.632: "Telecommunication management; Configuration Management (CM); CN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[15] 3GPP TS 32.692 "Inventory Management (IM) network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[16] 3GPP TS 32.742: "Telecommunication management; Configuration Management (CM); Signalling Transport Network (STN) Interface Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

3 Definitions and abbreviations

3.1 Definitions

For terms and definitions please refer to 3GPP TS 32.101 [1], TS 32.102 [2], TS 32.612 [3], TS 32.622 [4], TS 32.632 [14], TS 32.642 [11], TS 32.652 [12], TS 32.692 [15] and TS 32.742 [16].

- IRP document version number string (or "IRPVersion"): See 3GPP TS 32.312 [13].

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CORBA	Common Object Request Broker Architecture
DN	Distinguished Name
IS	Information Service
IDL	Interface Definition Language (OMG)
IRP	Integration Reference Point
MO	Managed Object
MOC	Managed Object Class
NRM	Network Resource Model
OMG	Object Management Group
SS	Solution Set

4 Mapping

4.1 General Mappings

All MOs are arranged in a **containment** structure, according to the containment relations defined in the NRM. This structure is held internally by the IRPAgent. Externally, the MO containment structure is defined by the semantics in the distinguished name syntax. The distinguished name (DN) for a MO contains the distinguished name of the parent plus the Relative DN for the MO itself.

Associations as defined in the NRM (UML) are in this document mapped to attributes in the MIB. The names of the roles for an association in the NRM are used for defining attribute names in the MIB. When the cardinality for a role is 0..1 or 1..1 the datatype for the attribute is defined as a MO reference. The value of a MO reference contains the distinguished name of the referred MO. When the cardinality for a role allows more than one referred MO instances, the attribute will contain a sequence of MO references (i.e., DNs).

4.2 Operation and Notification mapping

The IS part of Bulk CM: IRP defines semantics of operations and notifications visible across the Bulk Configuration IRP. The table below indicates mapping of these operations and notifications to their equivalents defined in this document.

There are 3 qualifications for each row of the following mapping table.
The 3 qualifications correspond to the three IS-defined packages: Controlled Upload & Provisioning, Controlled Upload and Simple Upload.
Not all operations/notifications specified in the following table are required for all 3 packages.
An "-" indicates that the subject operation or notification is not allowed by that corresponding package.

Table 1: Mapping from IM Notification/Operation to SS equivalents

IS Operation/ notification	SS Method	Qualifier
startSession	start_session	M,M,-
endSession	end_session	M,M,-
upload	upload	M,M,M
download	download	M,-,-
activate	activate	M,-,-
getSessionStatus	get_session_status	M,M,-
getSessionIds	get_session_ids	M,M,-
getSessionLog	get_session_log	M,M,-
fallback	fallback	M,-,-
abortSessionOperation	abort_session_operation	M,M,-
getIRPVersion	get_bulk_cm_irp_versions get_controlled_upload_bulk_cm_irp_versions get_simple_upload_bulk_cm_irp_versions	M,-,- -,M,- -,-,M
notifySessionStateChanged	push_structured_event Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 5.1	M,M,M
notifyGetSessionLogEnded	push_structured_event Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 5.1.	M,M,-
preactivate	preactivate	O,-,-
validate	validate	O,-,-
getOperationProfile	get_bulk_cm_irp_operation_profile get_controlled_upload_bulk_cm_irp_operation_profile get_simple_upload_bulk_cm_irp_operation_profile	O,-,- -,O,- -,-,O
getNotificationProfile	get_bulk_cm_irp_notification_profile get_controlled_upload_bulk_cm_irp_notification_profile get_simple_upload_bulk_cm_irp_notification_profile	O,-,- -,O,- -,-,O

4.3 Operation Parameter Mapping

Reference Bulk CM IRP; Information Service [3] defines semantics of parameters carried in operations. The tables below indicate the mapping of these parameters, as per operation, to their equivalents defined in this SS.

Table 2: Mapping from IS startSession parameters to SS equivalents

IS Operation parameter	SS parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
status	Exceptions: StartSessionException, SessionIdInUseException, MaxSessionReachedException, ManagedGenericIRPSystem::InvalidParameter	M

Table 3: Mapping from IS endSession parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
status	Exceptions: EndSessionException, UnknownSessionIdException, NotValidInCurrentStateException, ManagedGenericIRPSystem::InvalidParameter	M

Table 4: Mapping from IS upload parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
uploadDataFileReference	BulkCmIRPConstDefs::FileDestination sink	M
baseObjectInstance	BulkCmIRPConstDefs::DistinguishedName base_object	M
scope, filter	BulkCmIRPConstDefs::SearchControl search_control	M
status	Exceptions: UploadException, UnknownSessionIdException, MaxSessionReachedException, NotValidInCurrentStateException, ConcurrencyException, IllegalDNFormatException, IllegalFilterFormatException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalURLFormatException, ManagedGenericIRPSystem::InvalidParameter	M
NOTE: The IllegalURLFormatException does not imply that the transfer protocol used must be a URL. The transfer protocol is dependant on the file format definition, i.e. in the case of XML, FileDestination will be a URL.		

Table 5: Mapping from IS download parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
downloadDataFileReference	BulkCmIRPConstDefs::FileDestination source	M
status	Exceptions: DownloadException, UnknownSessionIdException, MaxSessionReachedException, NotValidInCurrentStateException, IllegalURLFormatException, ManagedGenericIRPSystem::InvalidParameter	M
NOTE: The IllegalURLFormatException does not imply that the transfer protocol used must be a URL. The transfer protocol is dependant on the file format definition, i.e. in the case of XML, FileDestination will be a URL.		

Table 6: Mapping from IS activate parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
activationMode	BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode	O
fallbackEnabled	boolean fallback	M
status	Exceptions: ActivateException, UnknownSessionIdException, NotValidInCurrentStateException, ConcurrencyException, IllegalActivationModeException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table 7: Mapping from IS fallback parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
status	Exceptions: FallbackException, UnknownSessionIdException, NoFallbackException, NotValidInCurrentStateException, ConcurrencyException, ManagedGenericIRPSystem::InvalidParameter	M

Table 8: Mapping from IS abortSessionOperation parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
status	Exceptions: AbortSessionOperationException, UnknownSessionIdException, NotValidInCurrentStateException, ManagedGenericIRPSystem::InvalidParameter	M

Table 9: Mapping from IS getSessionIds parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionIdList	Return value of type BulkCmIRPConstDefs::SessionIdList	M
status	Exceptions: GetSessionIdsException, ManagedGenericIRPSystem::InvalidParameter	M

Table 10: Mapping from IS getSessionStatus parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
sessionState	Return value of type BulkCmIRPConstDefs::SessionState	M
Not specified in IS	BulkCmIRPConstDefs::ErrorInformation error_information	M
status	Exceptions: GetSessionStatusException, UnknownSessionIdException, ManagedGenericIRPSystem::InvalidParameter	M

Table 11: Mapping from IS getSessionLog parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
logFileReference	BulkCmIRPConstDefs::FileDestination sink	M
contentType	boolean only_error_info	M
status	Exceptions: GetSessionLogException, UnknownSessionIdException, IllegalURLFormatException, ManagedGenericIRPSystem::InvalidParameter	M
NOTE: The IllegalURLFormatException does not imply that the transfer protocol used must be a URL. The transfer protocol is dependant on the file format definition, i.e. in the case of XML, FileDestination will be a URL.		

Table 12: Mapping from IS getBulkCmIRPVersion parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
versionNumberList	Return value of type ManagedGenericIRPConstDefs::VersionNumberSet	M
status	Exceptions: GetBulkCmIRPVersionsException	M

Table 13: Mapping from IS validate parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
activationMode	BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode	O
status	Exceptions: ValidateException , UnknownSessionIdException, NotValidInCurrentStateException, ConcurrencyException, IllegalActivationModeException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ManagedGenericIRPSystem::OperationNotSupported	M

Table 14: Mapping from IS preactivate parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
verificationMode	BulkCmIRPConstDefs::VerificationModeTypeOpt verification_mode	O
activationMode	BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode	O
fallbackEnabled	boolean fallback	M
status	Exceptions: PreactivateException, UnknownSessionIdException, NotValidInCurrentStateException, ConcurrencyException, IllegalActivationModeException, IllegalVerificationModeException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ManagedGenericIRPSystem::OperationNotSupported	M

Table 15: Mapping from IS getOperationProfile parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
irpVersion	ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version	M
operationNameProfile, operationParameterProfile	Return value of type ManagedGenericIRPConstDefs::MethodList	M
status	Exceptions: GetBulkCMIRPOperationProfileException, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table 16: Mapping from IS getNotificationProfile parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
irpVersion	ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version	M
notificationNameProfile, notificationParameterProfile	Return value of type ManagedGenericIRPConstDefs::MethodList	M
status	Exceptions: GetBulkCMIRPNotificationProfileException, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

4.4 Notification parameter mapping

Reference 3G TS 32.612 [3] defines semantics of parameters carried in notifications. The following tables indicate the mapping of these parameters to their OMG CORBA Structured Event (defined in OMG Notification Service [6]) equivalents. The composition of OMG Structured Event, as defined in the OMG Notification Service [6], 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 Bulk CM IRP: IS [3] defined notification parameters.

Table 17: Mapping from IS notifyGetSessionLogEnded parameters to SS equivalents

IS Parameter	OMG CORBA Structured Event Attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name	M	It carries the IRP document version number string. See sub-clause 3.3. It indicates the syntax and semantics of the Structured Event as defined by this specification.
notificationType	type_name	M	It carries the string NOTIFY_GET_SESSION_LOG_ENDED.
sessionLogStatus	event_name	M	It carries either the string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY or GET_SESSION_LOG_COMPLETED_UNSUCCESSFULLY. In the case of the latter, the NV pair indicating ERROR_INFORMATION may be present.
There is no corresponding IS attribute.	Variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. Name of NV pair is the MANAGED_OBJECT_INSTANCE of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a string. See encoding of this string in [5]. These are attributes of Header defined in the IS.
notificationId	One NV pair of remaining_body	M	Name of NV pair is the NOTIFICATION_ID of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a long. This is an attribute of Header defined in the IS.
eventTime	One NV pair of filterable_body_fields	M	Name of NV pair is the EVENT_TIME of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a IRPTime. This is an attribute of Header of the IS.

IS Parameter	OMG CORBA Structured Event Attribute	Qualifier	Comment
systemDN	One NV pair of filterable_body_fields	M	Name of NV pair is the SYSTEM_DN of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a string. This is an attribute of Header defined in the IS.
sessionId	One NV pair of remaining_body	M	Name of NV pair is the SESSION_ID of interface NotifyGetSessionLogEnded of module BulkCMIRPNotifications. Value of NV pair is a string.
sourceIndicator	One NV pair of remaining_body	O	Name of NV pair is the SOURCE_INDICATOR of interface NotifyGetSessionLogEnded of module BulkCMIRPNotifications. Value of NV pair is a string.
There is no corresponding IS attribute.	One NV pair of remaining_body	M	Name of NV pair is the ERROR_INFORMATION of interface NotifyGetSessionLogEnded of module BulkCMIRPNotifications. Value of NV pair is a string.

Table 18: Mapping from IS notifySessionStateChanged parameters to SS equivalents

IS Parameter	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name	M	It carries the IRP document version number string. See sub-clause 3.3. It indicates the syntax and semantics of the Structured Event as defined by this specification.
notificationType	type_name	M	It carries the string NOTIFY_SESSION_STATE_CHANGED. This is an attribute of Header defined in the IS.
sessionState	event_name	M	It carries one of the following: UPLOAD_FAILED, UPLOAD_COMPLETED, DOWNLOAD_FAILED, DOWNLOAD_COMPLETED, ACTIVATION_FAILED, ACTIVATION_PARTLY_REALISED, ACTIVATION_COMPLETED, FALLBACK_FAILED, FALLBACK_PARTLY_REALISED, FALLBACK_COMPLETED, VALIDATION_FAILED, VALIDATION_COMPLETED, PREACTIVATION_FAILED, PREACTIVATION_PARTLY_REALISED, PREACTIVATION_COMPLETED. In the case of XXX_FAILED and XXX_PARTLY_REALISED, the NV pair indicating ERROR_INFORMATION may be present.
There is no corresponding IS attribute.	Variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. Name of NV pair is the MANAGED_OBJECT_INSTANCE of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a string. See encoding of this string in [5]. These are attributes of Header defined in the IS.
notificationId	One NV pair of remaining_body	M	Name of NV pair is the NOTIFICATION_ID of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a long. This is an attribute of Header defined in the IS.
eventTime	One NV pair of filterable_body_fields	M	Name of NV pair is the EVENT_TIME of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a IRPTime. This is an attribute of Header of the IS.

IS Parameter	OMG CORBA Structured Event attribute	Qualifier	Comment
systemDN	One NV pair of filterable_body_fields	M	Name of NV pair is the SYSTEM_DN of interface AttributeNameValue of module NotificationIRPConstDefs. Value of NV pair is a string. This is an attribute of Header defined in the IS.
sessionId	One NV pair of remaining_body	M	Name of NV pair is the SESSION_ID of interface NotifySessionStateChange of module BulkCMIRPNotifications. Value of NV pair is a string.
sourceIndicator	One NV pair of remaining_body	O	Name of NV pair is the SOURCE_INDICATOR of interface NotifySessionStateChange of module BulkCMIRPNotifications. Value of NV pair is a string.
There is no corresponding IS attribute.	One NV pair of remaining _body	M	Name of NV pair is the ERROR_INFORMATION of interface NotifySessionStateChange of module BulkCMIRPNotifications. Value of NV pair is a string.

4.5 Two modes of operations

The upload, download, validate, preactivate, activate, get_session_log, and fallback are methods that use asynchronous mode of operation. The IRPManager uses the methods to request a task to be done. The IRPAgent, via the method return, indicates that it has understood the request and has begun to perform the task requested. When the IRPAgent has completed the requested task, either successfully or not, the IRPAgent will emit a notification, e.g., notifySessionStateChanged() defined in IS level and mapped to push() in SS level, to indicate the completion status of the requested task. If the IRPManager has subscribed (e.g., via the attach_push() of Notification IRP) for notifications, then the IRPManager will receive the notification.

The start_session, end_session, abort_session_operation, get_session_status, get_session_ids, get_bulk_CM_IRP_operation_profile, get_bulk_CM_IRP_notification_profile and get_bulkCM_IRP_version are methods that use synchronous mode of operation. The IRPManager uses these methods to request some information or a task to be done. The IRPAgent performs the requested task and, via the method return, indicates the requested information or if the requested task has completed successfully or not.

4.6 Mapping from IS State Names to SS equivalents

State names, as defined in the IS part of Bulk CM, consists of two sub-parts in this SS, namely SubPhase and SubState. The table below shows the mapping between these substates and the IS state name. All combinations of SubPhase and SubState not described below are considered invalid.

Table 19: Mapping from IS State Names to SS equivalents

IS State Name	SS SubPhase	SS SubState
IDLE	IDLE_PHASE	COMPLETED
UPLOAD_FAILED	UPLOAD_PHASE	FAILED
UPLOAD_IN_PROGRESS	UPLOAD_PHASE	IN_PROGRESS
UPLOAD_COMPLETED	UPLOAD_PHASE	COMPLETED
DOWNLOAD_FAILED	DOWNLOAD_PHASE	FAILED
DOWNLOAD_IN_PROGRESS	DOWNLOAD_PHASE	IN_PROGRESS
DOWNLOAD_COMPLETED	DOWNLOAD_PHASE	COMPLETED
ACTIVATION_FAILED	ACTIVATION_PHASE	FAILED
ACTIVATION_IN_PROGRESS	ACTIVATION_PHASE	IN_PROGRESS
ACTIVATION_COMPLETED	ACTIVATION_PHASE	COMPLETED
ACTIVATION_PARTLY_COMPLETED	ACTIVATION_PHASE	PARTLY_REALISED
FALLBACK_FAILED	FALLBACK_PHASE	FAILED
FALLBACK_IN_PROGRESS	FALLBACK_PHASE	IN_PROGRESS
FALLBACK_COMPLETED	FALLBACK_PHASE	COMPLETED
FALLBACK_PARTLY_COMPLETED	FALLBACK_PHASE	PARTLY_REALISED
VALIDATION_FAILED	VALIDATION_PHASE	FAILED
VALIDATION_IN_PROGRESS	VALIDATION_PHASE	IN_PROGRESS
VALIDATION_COMPLETED	VALIDATION_PHASE	COMPLETED
PREACTIVATION_FAILED	PREACTIVATION_PHASE	FAILED
PREACTIVATION_IN_PROGRESS	PREACTIVATION_PHASE	IN_PROGRESS
PREACTIVATION_COMPLETED	PREACTIVATION_PHASE	COMPLETED
PREACTIVATION_PARTLY_COMPLETED	PREACTIVATION_PHASE	PARTLY_REALISED

4.7 Package Mapping

The Bulk CM IRP: IS (3GPP TS 32.612 [3]) clause 7 specifies packages of capabilities.
The IS-defined packages are mapped into IDL module constructs. Specifically:

- The operations named in the IS-defined packages Simple Upload, Controlled Upload and Controlled Upload & Provisioning are mapped to methods in SimpleUploadBulkCMIRSystem::SimpleUploadBulkCMIRP, ControlledUploadBulkCMIRPSystem::ControlledUploadBulkCMIRP and BulkCmIRPSystem::BulkCmIRP respectively (see clause A.2).

- The notifications named in the IS-defined Simple Upload, Controlled Upload and Controlled Upload & Provisioning are mapped to SS Interfaces defined in IDL module BulkCMIRPNotifications (see clause A.3).

5 BulkCMIRPNotifications Interface

OMG CORBA Notification push operation is used to realise the notification of BulkCMIRPNotifications. All the notifications in this interface are implemented using this push_structured_event method.

5.1 Method push (M)

```
module CosNotifyComm {

        …

        Interface SequencePushConsumer : NotifyPublish {
                void push_structured_events(

                        in CosNotification::EventBatch notifications)

            raises( CosEventComm::Disconnected);

                …

        }; // SequencePushConsumer

        …

}; // CosNotifyComm
```

NOTE 1: The push_structured_events method takes an input parameter of type EventBatch as defined in the OMG CosNotification module (OMG Notification Service [6]). This data type is the same as a sequence of Structured Events. Upon invocation, this parameter will contain a sequence of Structured Events being delivered to IRPManager by IRPAgent to which it is connected.

NOTE 2: The maximum number of events that will be transmitted within a single invocation of this operation is controlled by IRPAgent wide configuration parameter.

NOTE 3: The amount of time the supplier (IRPAgent) of a sequence of Structured Events will accumulate individual events into the sequence before invoking this operation is controlled by IRPAgent wide configuration parameter as well.

NOTE 4: IRPAgent may push EventBatch with only one Structured Event.

Annex A (normative): IDL Specifications

A.1 IDL specification (file name BulkCmIRPConstDefs.idl)

```
//File: BulkCmIRPConstDefs.idl
#ifndef _BULK_CM_IRP_CONST_DEFS_IDL_
#define _BULK_CM_IRP_CONST_DEFS_IDL_

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

/* ## Module: BulkCmIRPConstDefs
This module contains type definitions for the Bulk CM IRP
=================================================================
*/
module BulkCmIRPConstDefs
{
   /*
   This block identifies the notification types defined by
   this Bulk CM IRP version.
   This string is used in the second field of the Structured
   Event.
   */
   interface NotificationType
   {
      const string NOTIFY_SESSION_STATE_CHANGED = "x1";
      const string NOTIFY_GET_SESSION_LOG_ENDED = "x2";
   };

   /*
   This block assigns value for the name of the NV of the Structured Event.
   */
   interface AttributeNameValue
   {
      const string SESSION_ID = "k";
      const string SOURCE_INDICATOR = "m";
      const string ERROR_INFORMATION = "n";
   };

   /*
   This block defines all possible values for sessionState.
   One of these strings appear in the event_name of the
   Structured Event of notifySessionStateChanged notification.
   */
   interface SessionStateChangeNotification
   {
      const string UPLOAD_FAILED = "x1";
      const string UPLOAD_COMPLETED = "x2";
      const string DOWNLOAD_FAILED = "x3";
      const string DOWNLOAD_COMPLETED = "x4";
      const string ACTIVATION_FAILED = "x5";
      const string ACTIVATION_PARTLY_REALISED = "x6";
      const string ACTIVATION_COMPLETED = "x7";
      const string FALLBACK_FAILED = "x8";
      const string FALLBACK_PARTLY_REALISED = "x9";
      const string FALLBACK_COMPLETED = "x10";
      const string VALIDATION_FAILED = "x11";
      const string VALIDATION_COMPLETED = "x12";
      const string PREACTIVATION_FAILED = "x13";
      const string PREACTIVATION_PARTLY_REALISED = "x14";
      const string PREACTIVATION_COMPLETED = "x15";
   };

   /*
   This block defines all possible values for sessionLogStatus
   One of these strings appear in the event_name of the Structured
   Event of notifyGetSessionLogEnded notification.
   */
```

```
interface LogStateNotification
{
   const string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY = "x1";
   const string GET_SESSION_LOG_COMPLETED_UNSUCESSFULLY = "x2";
};

/*
For each started configuration session a unique identifier is generated
by the IRPManager. An sessionId can not be used for an upload if it is
already in use of a download configuration and vice versa.
*/
typedef string SessionId;

/*
This string field is used in order to provide additional error information
if an operation has failed.
*/
typedef string ErrorInformation;

/*
Defines the different subphases of a configuration session
e.g. thus it is easy to implement a detection of an upload
or a download/activate session.
*/
enum SubPhase {IDLE_PHASE, DOWNLOAD_PHASE, UPLOAD_PHASE, ACTIVATION_PHASE,
               FALLBACK_PHASE, PREACTIVATION_PHASE, VALIDATION_PHASE};

/*
Defines the different substates of a configuration session. This includes
the transition state as well.
*/
enum SubState {COMPLETED, FAILED, PARTLY_REALISED, IN_PROGRESS};

/*
Defines state of a configuration session with the phase and the substate
of the configuration.
*/
struct SessionState
{
   SubPhase sub_phase;
   SubState sub_state;
};

/*
Contains the list of all current sessionIds
*/
typedef sequence <SessionId> SessionIdList;

/*
Specifies a complete destination path (including filename).
*/
typedef string FileDestination;

/*
The format of Distinguished Name is specified in
the Naming Conventions for Managed Objects; TS 32.300.
e.g. "SubNetwork=10001,ManagedElement=400001" identifies a
ManagedElement instance of the object model.
*/
typedef string DistinguishedName;

/*
Used within the upload method to give filter critera
*/
typedef string Filter;

/*
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.
*/
enum ScopeType {BASE_ONLY, BASE_NTH_LEVEL, BASE_SUBTREE, BASE_ALL};

/*
Controls the searching for MOs during upload, and contains:
the type of scope ("type" field),
the level of scope ("level" field),
   the filter ("filter_" field),
   The type and level fields are mandatory.
   The filter field is mandatory (The filter will have to be
   set to an empty string if it has no other value).
   */
   struct SearchControl
   {
      ScopeType type;
      unsigned long level;
      Filter filter_;
   };

   /*
   This indicates how the activation is executed, either with least service
   impact or least elapsed time.
   */
   enum ActivationMode {LEAST_SERVICE_IMPACT, LEAST_ELAPSED_TIME};

   /*
   This indicates the level of verification of bulk configuration data done,
   either full or limited checking.
   */
   enum VerificationMode {FULL_CHECKING, LIMITED_CHECKING};

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

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

};

#endif // _BULK_CM_IRP_CONST_DEFS_IDL_
```

A.2 IDL specification (file Name BulkCMIRPSystem.idl)

```
//File: BulkCMIRPSystem.idl
#ifndef _BULK_CM_IRP_SYSTEM_IDL_
#define _BULK_CM_IRP_SYSTEM_IDL_

#include <BulkCmIRPConstDefs.idl>
#include <ManagedGenericIRPConstDefs.idl>
#include <ManagedGenericIRPSystem.idl>

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

/* ## Module: BulkCmIRPSystem
This module implements capabilities of Bulk CM IRP.
================================================================
*/
module BulkCmIRPSystem
{
   /*
   The request cannot be processed due to a situation of concurrency.
   E.g. two concurrent activation requests involving the same ManagedElement
   instance. The semantics carried in reason is outside the scope of this IRP.
   */
   exception ConcurrencyException { string reason; };

   /*
   The provided filter is malformed or invalid. The semantics carried in reason
   is outside the scope of this IRP.
   */
   exception IllegalFilterFormatException { string reason; };

   /*
   The provided Distinguished Name is malformed or invalid. The semantics
   carried in reason is outside the scope of this IRP.
   */
   exception IllegalDNFormatException { string reason; };

   /*
   The provided scope type is illegal. The semantics carried in reason is
   outside the scope of this IRP.
   */
   exception IllegalScopeTypeException { string reason; };

   /*
   The provided scope level is illegal. The semantics carried in reason is
   outside the scope of this IRP.
   */
   exception IllegalScopeLevelException { string reason; };

   /*
   The request cannot be processed because no fallback data is available, i.e.
   fallback capability was previously not asked for.
   */
   exception NoFallbackException {};

   /*
   The provided sessionId value is already used for another configuration
   session. The semantics carried in reason is outside the scope of this IRP.
   */
   exception SessionIdInUseException { string reason; };

   /*
   The provided URL is malformed or invalid. The semantics carried in reason is
   outside the scope of this IRP.
   */
   exception IllegalURLFormatException{ string reason; };

   /*
   The provided sessionId value does not identify any existing configuration
   session.
   */
   exception UnknownSessionIdException {};

   /*
```

```
The request cannot be processed because it is not valid in the current state
of the configuration session.
*/
exception NotValidInCurrentStateException
{
   BulkCmIRPConstDefs::SessionState current_state;
};

/*
The request cannot be processed because the maximum number of simultaneously
running configuration sessions has been reached. The semantics carried in
reason is outside the scope of this IRP.
*/
exception MaxSessionReachedException { string reason; };

/*
The provided ActivationMode type is illegal. The semantics carried in reason
is outside the scope of this IRP.
*/
exception IllegalActivationModeException { string reason; };

/*
The provided VerificationMode type is illegal. The semantics carried in
reason is outside the scope of this IRP.
*/
exception IllegalVerificationModeException { string reason; };

/*
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 GetBulkCmIRPVersionsException { string reason; };
exception UploadException { string reason; };
exception DownloadException { string reason; };
exception ActivateException { string reason; };
exception ValidateException { string reason; };
exception PreactivateException { string reason; };
exception GetBulkCMIRPOperationProfileException { string reason; };
exception GetBulkCMIRPNotificationProfileException { string reason; };
exception GetSessionLogException { string reason; };
exception StartSessionException { string reason; };
exception GetSessionStatusException { string reason; };
exception FallbackException { string reason; };
exception EndSessionException { string reason; };
exception AbortSessionOperationException { string reason; };
exception GetSessionIdsException { string reason; };

/*
Defines the System interface of a EM. It defines all methods which are
necessary to control a configuration session from a IRPManager.
*/
interface BulkCmIRP
{
   /*
   Return the list of all supported Bulk CM IRP versions.
   */
   ManagedGenericIRPConstDefs::VersionNumberSet get_bulk_cm_irp_versions (
   )
   raises (GetBulkCmIRPVersionsException);

   /*
   Return the list of all supported operations and their supported
   parameters for a specific BulkCM IRP version.
   */
   ManagedGenericIRPConstDefs::MethodList get_bulk_cm_irp_operation_profile (
      in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
   )
   raises (GetBulkCMIRPOperationProfileException,
           ManagedGenericIRPSystem::OperationNotSupported,
           ManagedGenericIRPSystem::InvalidParameter);

   /*
   Return the list of all supported notifications and their supported
   parameters for a specific BulkCM IRP version.
   */
   ManagedGenericIRPConstDefs::MethodList
      get_bulk_cm_irp_notification_profile
```

```
(
   in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
)
raises (GetBulkCMIRPNotificationProfileException,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Uploads a configuration from the subnetwork. The result is put in a
configuration data file in an area specified by the IRPManager.
The MIB of the subnetwork is iterated by means of containment search,
using a SearchControl to control the search and the returned results.
All MOs in the scope constitutes a set that the filter works on.
In case of a concurrent running session the function will
return an exception. If the value of the given baseObject or Filter
does not exist then this asynchronous error condition will be notified.
*/
void upload (
   in BulkCmIRPConstDefs::SessionId session_id,
   in BulkCmIRPConstDefs::FileDestination sink,
   in BulkCmIRPConstDefs::DistinguishedName base_object,
   in BulkCmIRPConstDefs::SearchControl search_control
)
raises (UploadException, UnknownSessionIdException,
        MaxSessionReachedException, NotValidInCurrentStateException,
        ConcurrencyException,
        IllegalDNFormatException, IllegalFilterFormatException,
        IllegalScopeTypeException, IllegalScopeLevelException,
        IllegalURLFormatException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Indicates the EM that it can download a configuration data file from
a given configuration data file storage area. The EM will check the
consistence of the configuration data and the software compatibilty.
*/
void download (
   in BulkCmIRPConstDefs::SessionId session_id,
   in BulkCmIRPConstDefs::FileDestination source
)
raises (DownloadException, UnknownSessionIdException,
        MaxSessionReachedException, NotValidInCurrentStateException,
        IllegalURLFormatException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Activates a previously downloaded and sucessfully parsed configuration
inside a session. This means that the configuration will be introduced
in the live sub-network. In case of a concurrent running session
the function will return an exception.
*/
void activate (
   in BulkCmIRPConstDefs::SessionId session_id,
   in BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode,
   in boolean fallback
)
raises (ActivateException, UnknownSessionIdException,
        NotValidInCurrentStateException, ConcurrencyException,
        IllegalActivationModeException,
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Uploads a log from the subnetwork which is usally used for error
analysis. The log is put in a logfile in the filesystem which can
be accessed by the EM. If there are no log entries an empty log file
is uploaded.
*/
void get_session_log (
   in BulkCmIRPConstDefs::FileDestination sink,
   in BulkCmIRPConstDefs::SessionId session_id,
   in boolean only_error_info
)
raises (GetSessionLogException, UnknownSessionIdException,
        IllegalURLFormatException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
```

```
Creates an instance of the configuration session state machine. The
IDLE_PHASE & COMPLETED is notified
*/
void start_session (
   in BulkCmIRPConstDefs::SessionId session_id
)
raises (StartSessionException, SessionIdInUseException,
        MaxSessionReachedException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Returns the state of a configuration session.
*/
BulkCmIRPConstDefs::SessionState get_session_status (
   in BulkCmIRPConstDefs::SessionId session_id,
   out BulkCmIRPConstDefs::ErrorInformation error_information
)
raises (GetSessionStatusException, UnknownSessionIdException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Actives a fallback area. Each time a configuration is activated a
fallback area can be created, s. activate parameter.
This area is backup of the complete configuration which can be
restored by this method. The process is as follows:
1. When the method activate(...,..., TRUE) is used,
   a copy of the valid area is taken before the activation
   of the new planned data has started. Only one fallback area can
   exists at a time for a specific scope of the subnetwork.
2. When a fallback area is avilable and triggered by this method, the
   previous valid area is replaced with the data stored in
   the fall back area.
If the EM detects that the former configuration has never been
changed it returns an exception because it does not trigger an
activation of the former data.
*/
void fallback (
   in BulkCmIRPConstDefs::SessionId session_id
)
raises (FallbackException, UnknownSessionIdException, NoFallbackException,
        NotValidInCurrentStateException, ConcurrencyException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
The IRPManager invokes this operation to delete all its temporary
entities and the related sessionId which belong to the scope of
a configuration session. This includes the related error and log
informationen too.
*/
void end_session (
   in BulkCmIRPConstDefs::SessionId session_id
)
raises (EndSessionException, UnknownSessionIdException,
        NotValidInCurrentStateException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
The IRPManager invokes this operation to abort an active operation
during a configuration session. It is only effecting
a configuration session in state IN_PROGRESS. In this case the
current session task is interrupted, e.g. the activating in progress,
using best effort strategy, and a state change is notified
*/
void abort_session_operation (
   in BulkCmIRPConstDefs::SessionId session_id
)
raises (AbortSessionOperationException, UnknownSessionIdException,
        NotValidInCurrentStateException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Returns a list all sessionIds of current running configuration sessions.
*/
BulkCmIRPConstDefs::SessionIdList get_session_ids (
)
raises (GetSessionIdsException);

/*
```

```
      Validates previously downloaded bulk configuration data inside a session.
      Detects errors in the data prior to requesting preactivation or
      activation.
      */
      void validate (
         in BulkCmIRPConstDefs::SessionId session_id,
         in BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode
      )
      raises (ValidateException, UnknownSessionIdException,
                 NotValidInCurrentStateException, ConcurrencyException,
                 IllegalActivationModeException,
                 ManagedGenericIRPSystem::ParameterNotSupported,
                 ManagedGenericIRPSystem::InvalidParameter,
                 ManagedGenericIRPSystem::OperationNotSupported);

      /*
      Preactivates previously downloaded bulk configuration data inside a
      session. This operation validates configuration data changes in the
      context of the current data and pre-processes the configuration data
      changes.
      */
      void preactivate (
         in BulkCmIRPConstDefs::SessionId session_id,
         in BulkCmIRPConstDefs::VerificationModeTypeOpt verification_mode,
         in BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode,
         in boolean fallback
      )
      raises (PreactivateException, UnknownSessionIdException,
              NotValidInCurrentStateException, ConcurrencyException,
              IllegalActivationModeException, IllegalVerificationModeException,
              ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter,
              ManagedGenericIRPSystem::OperationNotSupported);

   };

};

module SimpleUploadBulkCMIRPSystem
{
   exception GetSimpleUploadBulkCmIRPVersionsException { string reason; };
   exception GetSimpleUploadBulkCMIRPOperationProfileException
      { string reason; };
   exception GetSimpleUploadBulkCMIRPNotificationProfileException
      { string reason; };

   interface SimpleUploadBulkCMIRP
   {
      /*
      Return the list of all supported Bulk CM IRP versions.
      */
      ManagedGenericIRPConstDefs::VersionNumberSet
         get_simple_upload_bulk_cm_irp_versions (
      )
      raises (GetSimpleUploadBulkCmIRPVersionsException);

      /*
      Return the list of all supported operations and their supported
      parameters for a specific BulkCM IRP version.
      */
      ManagedGenericIRPConstDefs::MethodList
         get_simple_upload_bulk_cm_irp_operation_profile (
            in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
      )
      raises (GetSimpleUploadBulkCMIRPOperationProfileException,
              ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);

      /*
      Return the list of all supported notifications and their supported
      parameters for a specific BulkCM IRP version.
      */
      ManagedGenericIRPConstDefs::MethodList
         get_simple_upload_bulk_cm_irp_notification_profile
      (
         in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
      )
      raises (GetSimpleUploadBulkCMIRPNotificationProfileException,
               ManagedGenericIRPSystem::OperationNotSupported,
               ManagedGenericIRPSystem::InvalidParameter);

      /*
      Uploads a configuration from the subnetwork. The result is put in a
      configuration data file in an area specified by the IRPManager.
      The MIB of the subnetwork is iterated by means of containment search,
      using a SearchControl to control the search and the returned results.
      All MOs in the scope constitutes a set that the filter works on.
      In case of a concurrent running session the function will
      return an exception. If the value of the given baseObject or Filter
      does not exist then this asynchronous error condition will be notified.
      */
      void upload (
         in BulkCmIRPConstDefs::SessionId session_id,
         in BulkCmIRPConstDefs::FileDestination sink,
         in BulkCmIRPConstDefs::DistinguishedName base_object,
         in BulkCmIRPConstDefs::SearchControl search_control
      )
      raises (
         BulkCmIRPSystem::UploadException,
         BulkCmIRPSystem::UnknownSessionIdException,
         BulkCmIRPSystem::MaxSessionReachedException,
         BulkCmIRPSystem::NotValidInCurrentStateException,
         BulkCmIRPSystem::ConcurrencyException,
         BulkCmIRPSystem::IllegalDNFormatException,
         BulkCmIRPSystem::IllegalFilterFormatException,
         BulkCmIRPSystem::IllegalScopeTypeException,
         BulkCmIRPSystem::IllegalScopeLevelException,
         BulkCmIRPSystem::IllegalURLFormatException,
         ManagedGenericIRPSystem::InvalidParameter);
   };

};  // end of module SimpleUploadBulkCMIRPSystem

module ControlledUploadBulkCMIRPSystem
{
   exception GetControlledUploadBulkCmIRPVersionsException { string reason; };
   exception GetControlledUploadBulkCMIRPOperationProfileException
      { string reason; };
   exception GetControlledUploadBulkCMIRPNotificationProfileException
      { string reason; };

   interface ControlledUploadBulkCMIRP
   {
      /*
      Return the list of all supported Bulk CM IRP versions.
      */
      ManagedGenericIRPConstDefs::VersionNumberSet
         get_controlled_upload_bulk_cm_irp_versions (
      )
      raises (GetControlledUploadBulkCmIRPVersionsException);

      /*
      Return the list of all supported operations and their supported
      parameters for a specific BulkCM IRP version.
      */
      ManagedGenericIRPConstDefs::MethodList
         get_controlled_upload_bulk_cm_irp_operation_profile (
              in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
      )
      raises (GetControlledUploadBulkCMIRPOperationProfileException,
              ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);

      /*
      Return the list of all supported notifications and their supported
      parameters for a specific BulkCM IRP version.
      */
      ManagedGenericIRPConstDefs::MethodList
         get_controlled_upload_bulk_cm_irp_notification_profile (
            in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
      )
      raises (GetControlledUploadBulkCMIRPNotificationProfileException,
              ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);

      /*
```

```
Uploads a configuration from the subnetwork. The result is put in a
configuration data file in an area specified by the IRPManager.
The MIB of the subnetwork is iterated by means of containment search,
using a SearchControl to control the search and the returned results.
All MOs in the scope constitutes a set that the filter works on.
In case of a concurrent running session the function will
return an exception. If the value of the given baseObject or Filter
does not exist then this asynchronous error condition will be notified.
*/
void upload (
   in BulkCmIRPConstDefs::SessionId session_id,
   in BulkCmIRPConstDefs::FileDestination sink,
   in BulkCmIRPConstDefs::DistinguishedName base_object,
   in BulkCmIRPConstDefs::SearchControl search_control
)
raises (
   BulkCmIRPSystem::UploadException,
   BulkCmIRPSystem::UnknownSessionIdException,
   BulkCmIRPSystem::MaxSessionReachedException,
   BulkCmIRPSystem::NotValidInCurrentStateException,
   BulkCmIRPSystem::ConcurrencyException,
   BulkCmIRPSystem::IllegalDNFormatException,
   BulkCmIRPSystem::IllegalFilterFormatException,
   BulkCmIRPSystem::IllegalScopeTypeException,
   BulkCmIRPSystem::IllegalScopeLevelException,
   BulkCmIRPSystem::IllegalURLFormatException,
   ManagedGenericIRPSystem::InvalidParameter);

/*
Uploads a log from the subnetwork which is usally used for error
analysis. The log is put in a logfile in the filesystem which can
be accessed by the EM. If there are no log entries an empty log file
is uploaded.
*/
void get_session_log (
   in BulkCmIRPConstDefs::FileDestination sink,
   in BulkCmIRPConstDefs::SessionId session_id,
   in boolean only_error_info
)
raises (
   BulkCmIRPSystem::GetSessionLogException,
   BulkCmIRPSystem::UnknownSessionIdException,
   BulkCmIRPSystem::IllegalURLFormatException,
   ManagedGenericIRPSystem::InvalidParameter);

/*
Creates an instance of the configuration session state machine. The
IDLE_PHASE & COMPLETED is notified
*/
void start_session (
   in BulkCmIRPConstDefs::SessionId session_id
)
raises (
   BulkCmIRPSystem::StartSessionException,
   BulkCmIRPSystem::SessionIdInUseException,
   BulkCmIRPSystem::MaxSessionReachedException,
   ManagedGenericIRPSystem::InvalidParameter);

/*
Returns the state of a configuration session.
*/
BulkCmIRPConstDefs::SessionState get_session_status (
   in BulkCmIRPConstDefs::SessionId session_id,
   out BulkCmIRPConstDefs::ErrorInformation error_information
)
raises (
   BulkCmIRPSystem::GetSessionStatusException,
   BulkCmIRPSystem::UnknownSessionIdException,
   ManagedGenericIRPSystem::InvalidParameter);

/*
The IRPManager invokes this operation to delete all its temporary
entities and the related sessionId which belong to the scope of
a configuration session. This includes the related error and log
informationen too.
*/
void end_session (
   in BulkCmIRPConstDefs::SessionId session_id
```

```
      )
      raises (
         BulkCmIRPSystem::EndSessionException,
         BulkCmIRPSystem::UnknownSessionIdException,
         BulkCmIRPSystem::NotValidInCurrentStateException,
         ManagedGenericIRPSystem::InvalidParameter);

      /*
      The IRPManager invokes this operation to abort an active operation
      during a configuration session. It is only effecting
      a configuration session in state IN_PROGRESS. In this case the
      current session task is interrupted, e.g. the activating in progress,
      using best effort strategy, and a state change is notified
      */
      void abort_session_operation (
         in BulkCmIRPConstDefs::SessionId session_id
      )
      raises (
         BulkCmIRPSystem::AbortSessionOperationException,
         BulkCmIRPSystem::UnknownSessionIdException,
         BulkCmIRPSystem::NotValidInCurrentStateException,
         ManagedGenericIRPSystem::InvalidParameter);

      /*
      Returns a list all sessionIds of current running configuration sessions.
      */
      BulkCmIRPConstDefs::SessionIdList get_session_ids (
      )
      raises (
         BulkCmIRPSystem::GetSessionIdsException);

     };

};  // end of module ControlledUploadBulkCMIRPSystem

#endif // _BULK_CM_IRP_SYSTEM_IDL_
```

A.3 IDL specification (file name 'BulkCMIRPNotifications.idl')

```
//File: BulkCMNotifications.idl
#ifndef _BULK_CM_IRP_NOTIFICATIONS_IDL_
#define _BULK_CM_IRP_NOTIFICATIONS_IDL_

#include <NotificationIRPNotifications.idl>
#include <BulkCmIRPConstDefs.idl>

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

module BulkCMIRPNotifications
{

   interface NotifySessionStateChange: NotificationIRPNotifications::Notify
   {
      // This is the type_name (2nd field) of the fixed header.
      const string EVENT_TYPE =
         BulkCmIRPConstDefs::NotificationType::NOTIFY_SESSION_STATE_CHANGED;

      // ----------------------------
      // One of the strings here is the event_name (3rd field) of the
      // fixed header.
      // The first 2 are relevant for IS-defined packages Simple
      // Upload and Controlled Upload.
      // All are relevant for IS-defined package
      // Controlled Upload & Provisioning.

      const string UPLOAD_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::UPLOAD_FAILED;
      const string UPLOAD_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::UPLOAD_COMPLETED;
      const string DOWNLOAD_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::DOWNLOAD_FAILED;
      const string DOWNLOAD_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::DOWNLOAD_COMPLETED;
      const string ACTIVATION_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::ACTIVATION_FAILED;
      const string ACTIVATION_PARTLY_REALISED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::ACTIVATION_PARTLY_REALISED;
      const string ACTIVATION_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::ACTIVATION_COMPLETED;
      const string FALLBACK_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::FALLBACK_FAILED;
      const string FALLBACK_PARTLY_REALISED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::FALLBACK_PARTLY_REALISED;
      const string FALLBACK_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::FALLBACK_COMPLETED;
      const string VALIDATION_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::VALIDATION_FAILED;
      const string VALIDATION_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::VALIDATION_COMPLETED;
      const string PREACTIVATION_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::PREACTIVATION_FAILED;
      const string PREACTIVATION_PARTLY_REALISED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::PREACTIVATION_PARTLY_REALISED;
      const string PREACTIVATION_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::PREACTIVATION_COMPLETED;
      // -----------------------------

      const string SESSION_ID =
         BulkCmIRPConstDefs::AttributeNameValue::SESSION_ID;

      const string SOURCE_INDICATOR =
         BulkCmIRPConstDefs::AttributeNameValue::SOURCE_INDICATOR;
   };

   interface NotifyGetSessionLogEnded: NotificationIRPNotifications::Notify
   {
      // This is the type_name (2nd field) of the fixed header.
      const string EVENT_TYPE =
         BulkCmIRPConstDefs::NotificationType::NOTIFY_GET_SESSION_LOG_ENDED;
      // -----------------
      // One of the 2 strings here is the event_name (3rd field) of the
      // fixed header.
      const string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY =
         BulkCmIRPConstDefs::LogStateNotification::
            GET_SESSION_LOG_COMPLETED_SUCCESSFULLY;
      const string GET_SESSION_LOG_COMPLETED_UNSUCESSFULLY =
         BulkCmIRPConstDefs::LogStateNotification::
            GET_SESSION_LOG_COMPLETED_UNSUCESSFULLY;
      // ----------------------

      const string SESSION_ID =
         BulkCmIRPConstDefs::AttributeNameValue::SESSION_ID;

      const string SOURCE_INDICATOR =
         BulkCmIRPConstDefs::AttributeNameValue::SOURCE_INDICATOR;
   };

};

#endif // _BULK_CM_IRP_NOTIFICATIONS_IDL_
```

Annex B (informative): Change history

Change history								
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Cat**	**Old**	**New**
Jun 2001	SA_12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	--	2.0.0	4.0.0
Dec 2001	SA_14	SP-010644	0001	--	Correction of a notification name and Addition of missing table for fallback operation	F	4.0.0	4.1.0
Dec 2001	SA_14	SP-010644	0002	--	Corrections to the exceptions in the Bulk CM IRP CORBA Solution Set	F	4.0.0	4.1.0
Jun 2002	SA_16	SP-020297	0003	--	Add missing CORBA exceptions and descriptions of CORBA exception usage	F	4.1.0	4.2.0
Jun 2002	SA_16	SP-020296	0004	--	Correction of behaviour for IS parameter "saveFallback" of IS operation "activate"	F	4.1.0	4.2.0
Sep 2002	SA_17	SP-020485	0005	--	Correction of Mapping fallbackEnabled Qualifier	F	4.2.0	4.3.0
Sep 2002	SA_17	SP-020486	0006	--	Add Bulk CM IRP CORBA Solution Set Enhancements Rel-5	C	4.3.0	5.0.0
Mar 2003	SA_19	SP-030140	0008	--	Add subphases "PreactivationPhase" and "ValidationPhase" in "BulkCmIRPConstDefs" IDL definition	F	5.0.0	5.1.0
Mar 2003	SA_19	SP-030140	0009	--	Add missing Rel-4 CORBA IDL exceptions	F	5.0.0	5.1.0
Mar 2004	SA_23	SP-040105	--	--	Automatic upgrade to Rel-6 (no CR)	--	5.1.0	6.0.0
Dec 2004	SA_26	SP-040807	0010	--	Correct mapping of IS-defined non-filterable parameters to SS-defined non-filterable fields - Align with IS in 32.612	F	6.0.0	6.1.0
Dec 2004	SA_26	SP-040807	0011	--	Partition Bulk CM IRP capabilities into separate IDL modules – Align to IS in 32.612	F	6.0.0	6.1.0
Dec 2004	SA_26	SP-040807	0012	--	Add Signalling Transport Network (STN) NRM IRP in BulkCM IRP CORBA SS	B	6.0.0	6.1.0
Mar 2005	SA_27	SP-050045	0013	--	IDL incompliant to the style guide	F	6.1.0	6.2.0
Mar 2005	SA_27	SP-050045	0014	--	Generic System Context, update of reference to IS specification	F	6.1.0	6.2.0
Sep 2005	SA_29	SP-050461	0015		Align the CORBA SS IDL with TS 32.150 Style Guide	F	6.2.0	6.3.0
Jun 2007	SA_36	--	--	--	Automatic upgrade to Rel-7 (no CR) at freeze of Rel-7. Deleted reference to CMIP SS, discontinued from R7 onwards.	--	6.3.0	7.0.0
Mar 2009	SA_43	SP-090207	0016	--	Include reference to SOAP Solution Set specification	D	7.0.0	8.0.0
Dec 2009	SA-46	SP-090718	0017	--	Increase the linked IS version by the CR for IS to clarify scope parmaters for upload operation	F	8.0.0	8.1.0

History

Document history		
V8.0.0	April 2009	Publication
V8.1.0	January 2010	Publication