3GPP TS 32.602-3 V2.0.0 (2001-06)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Telecommunication Management;
3G Configuration Management:
Bulk CM IRP: CORBA Solution Set
(Release 4)

Keywords
Configuration Management

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Copyright Notification

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

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.

Due to the growing number of specifications to model new services and Resource Models for Configuration Management (CM), as well as the expected growth in size of each of them from 3GPP Release 4 onwards, a new structure of the specifications is already needed in Release 4. This structure is needed for several reasons, but mainly to enable more independent development and release for each part, as well as a simpler document identification and version handling. Another benefit would be that it becomes easier for bodies outside 3GPP, such as the ITU-T, to refer to telecom management specifications from 3GPP. The new structure of the specifications does not lose any information or functionality supported by the Release 1999. The restructuring also includes defining new IRPs for the Network Resource Model (NRM) parts of R99 Basic CM IRP (Generic, Core Network and UTRAN NRM). These IRPs are named "Network Resources IRP".

Further, the Notification IRP (in Release 1999: 32.106-1 to -4) and the Name convention for Managed Objects (in Release 1999: 32.106-8) have been moved to a separate number series used for specifications common between several management areas (e.g. CM, FM, PM).

Finally, in addition to the restructuring mentioned above, the need to define some new functionality and IRPs for CM compared to Release 1999, has also been identified. Firstly, a new Bulk CM IRP, and secondly an a GERAN Network Resources IRP, have been created. Thirdly, the Generic, UTRAN and GERAN Network Resources IRPs have been extended with support for GSM-UMTS Inter-system handover (ISH), and the 32.600 (Concept and High-level Requirements) has been modified to cover the high-level Bulk CM and ISH requirements.

Table 1: Mapping between Release '99 and the new specification numbering scheme

R99 Old no.	Old (R99) specification title	Rel-4 spec. no. with Bulk CM /ISH	Rel-4 specification title with Bulk CM/ISH
32.106-1	3G Configuration Management: Concept and Requirements	32.600	**3G Configuration Management: Concept and High-level Requirements**
32.106-1	<Notification IRP requirements from 32.106-1 and 32.106-2>	32.301-1	**Notification IRP**: Requirements
32.106-2	Notification IRP: IS	32.301-2	Notification IRP: Information Service
32.106-3	Notification IRP: CORBA SS	32.301-3	Notification IRP: CORBA SS
32.106-4	Notification IRP: CMIP SS	32.301-4	Notification IRP: CMIP SS
32.106-8	Name convention for Managed Objects	32.300	**Name Convention for Managed Objects**
-	-	32.602-1	**Bulk CM IRP**: Requirements
-	-	32.602-2	Bulk CM IRP: Information Service
-	-	32.602-3	Bulk CM IRP: CORBA SS
-	-	32.602-4	Bulk CM IRP: CMIP SS
-	-	32.602-5	Bulk CM IRP: XML file format definition
32.106-1	<Basic CM IRP Generic NRM requirements from 32.106-1 and 32.106-5>	32.620-1	**Generic Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (Generic NRM part)	32.620-2	Generic Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (Generic NRM related part)	32.620-3	Generic Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (Generic NRM related part)	32.620-4	Generic Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP UTRAN NRM requirements from 32.106-1 and 32.106-5>	32.622-1	**UTRAN Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (UTRAN NRM part)	32.622-2	UTRAN Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (UTRAN NRM related part)	32.622-3	UTRAN Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (UTRAN NRM related part)	32.622-4	UTRAN Network Resources IRP: CMIP SS
-	-	32.623-1	**GERAN Network Resources IRP**: Requirements
-	-	32.623-2	GERAN Network Resources IRP: NRM
-	-	32.623-3	GERAN Network Resources IRP: CORBA SS
-	-	32.623-4	GERAN Network Resources IRP: CMIP SS

The present document is CORBA Solution Set - Part 3 of 3GPP TS 32.602 "Bulk Configuration Management IRP".

1 Scope

The purpose of this *Bulk CM IRP: CORBA Solution Set* is to define the mapping of the IRP information service (see 3GPP TS 32.602-2 [3]) to the protocol specific details necessary for implementation of this IRP in a CORBA/IDL environment.

The present document does not describe any Network Resource Model (NRM) – they are described in Generic Network Resources IRP: NRM 3GPP TS 32.620-2 [4], UTRAN Network Resources IRP: NRM 3GPP TS 32.622-2 [11], GERAN Network Resources IRP: NRM 3GPP TS 32.623-2 [12].

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: "3G Telecom Management principles and high level requirements".

[2]   3GPP TS 32.102: "3G Telecom Management architecture".

[3]   3GPP TS 32.602-2: "Telecommunication Management; Configuration Management; Part 2: Bulk CM IRP; Information Service".

[4]   3GPP TS 32.620-2: "Telecommunication Management; Configuration Management; Part 2: Generic Network Resources IRP: NRM".

[5]   3GPP TS 32.300: "Telecommunication Management; Configuration Management; Part 8: 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 32.301-3: "Telecommunication Management; Configuration Management; Part 3: Notification Integration Reference Point: CORBA solution set".

[10]   3GPP TS 32.111-3: "Telecommunication Management; Fault Management; Part 3: Alarm Integration Reference Point: CORBA solution set".

[11]   3GPP TS 32.622-2: "Telecommunication Management; Configuration Management; Part 2: UTRAN Network Resources IRP: NRM".

[12]   3GPP TS 32.623-2: "Telecommunication Management; Configuration Management; Part 2: GERAN Network Resources IRP: NRM".

[13]   3GPP TS 32.112-1: "Generic IRP Management: Information Service".

3 Definitions and abbreviations

3.1 Definitions

For terms and definitions please refer to 3GPP TS 32.101 [1], 3GPP TS 32.102 [2], 3GPP TS 32.602-2 [3], 3GPP TS 32.620-2 [4], 3GPP TS 32.622-2 [11] and 3GPP TS 32.623-2 [12].

- IRP document version number string (or "IRPVersion"): See 3GPP TS 32.112-1 [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

3.3 IRP document version number string

The IRP document version number (sometimes called "IRPVersion" or "version number") string is used to identify this specification. The string is derived using a rule described in definition "IRP document version number string".

This string is returned in `getBulkCmIRPVersion` method and is carried in the first field of the notification header of all notifications related to this IRP.

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.

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

IS Operation/ notification	SS Method	Qualifier
startSession	start_session	M
endSession	end_session	M
upload	upload	M
download	download	M
activate	activate	M
getSessionStatus	get_session_status	M
getSessionIds	get_session_ids	M
getSessionLog	get_session_log	M
fallback	fallback	M
abortSessionOperation	abort_session_operation	M
getBulkCmIRPVersion	get_bulk_cm_IRP_version	M
notifySessionStateChanged	push_structured_event Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 5.1	M
notifySessionLogStatus	push_structured_event Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 5.1.	M

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	exception SessionIdInUseException	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	exception UnknownSessionIdException, exception TransitionStateException	M

Table 4: Mapping from IS upload parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
uploadDataFile Reference	BulkCmIRPConstDefs::FileDestination sink	M
baseObjectInstance	BulkCmIRPConstDefs::DistinguishedName base_object	M
scope, filter	BulkCmIRPConstDefs::SearchControl search_control	M
status	exception UnknownSessionIdException, exception TransitionStateException, exception ConcurrencyException, exception IllegalDistinguishedNameFormatException, exception IllegalFilterFormatException, exception IllegalScopeTypeException, exception IllegalScopeLevelException	M

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	exception UnknownSessionIdException	M

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

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
saveFallback	boolean fallback	O
status	exception UnknownSessionIdException, exception TransitionStateException, exception ConcurrencyException, exception ActivationModeException	M

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

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
status	exception UnknownSessionIdException	M

Table 8: Mapping from IS `getSessionIds` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionIdList	return of type BulkCmIRPConstDefs::SessionIdList	M
status	- no error condition identified	M

Table 9: Mapping from IS `getSessionStatus` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
sessionState	return of type BulkCmIRPConstDefs::SessionState	M
status	BulkCmIRPConstDefs::ErrorInformation error_information	M
status	exception UnknownSessionIdException	M

Table 10: 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`	`exception UnknownSessionIdException, exception ConcurrencyException`	M

Table 11: Mapping from IS `getBulkCmIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`versionNumberList`	return of type `ManagedGenericIRPConstDefs::VersionNumberSet`	M
`status`	- no error condition identified or described in SS	M

Table 12: Mapping from IS `getBulkCmIRPVersion` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
`versionNumberList`	Return value of type: CommonIRPConstDefs::VersionNumberSet	M
`status`	- (No failure conditions identified)	

4.4 Notification parameter mapping

Reference 3G TS 32.602-2 [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 13: 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.
`notification Type`	`type_name`	M	It carries the string NOTIFY_BULK_CM_LOG_STATE.
`sessionLogSt atus`	`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 parameter	`Variable Header`		
`managedObjec tClass, managedObjec tInstance`	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.
`notification Id`	One NV pair of `filterable_ body_fields`	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.
`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 `filterable_ body_fields`	M	Name of NV pair is the SESSION_ID of interface `AttributeNameValue` of module `BulkCMIRPConstDefs.`

			Value of NV pair is a string.
sourceIndicator	One NV pair of filterable_body_fields	O	Name of NV pair is the SOURCE_INDICATOR of interface AttributeNameValue of module BulkCMIRPConstDefs. Value of NV pair is a string.
There is no corresponding IS attribute.	One NV pair of filterable_body_fields		Name of NV pair is the ERROR_INFORMATION of interface AttributeNameValue of module BulkCMIRPConstDefs. Value of NV pair is a string.

Table 14: 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 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 filterable_body_fields	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.
`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 `filterable_ body_fields`	M	Name of NV pair is the SESSION_ID of interface `AttributeNameValue` of module `BulkCMIRPConstDefs.`
			Value of NV pair is a string.
`sourceIndica tor`	One NV pair of `filterable_ body_fields`	O	Name of NV pair is the SOURCE_INDICATOR of interface `AttributeNameValue` of module `BulkCMIRPConstDefs.`
			Value of NV pair is a string.
There is no corresponding IS attribute.	One NV pair of `filterable_ body_fields`		Name of NV pair is the ERROR_INFORMATION of interface `AttributeNameValue` of module `BulkCMIRPConstDefs.`
			Value of NV pair is a string.

4.5   Two modes of operations

The `upload`, `download`, `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` 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 15: 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

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: BulkCmIRPConstDefs

```
#ifndef BulkCmIRPConstDefs_IDL
#define BulkCmIRPConstDefs_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
{
   /*
   Defines the current Bulk CM IRP version
   This string is the return value for get_bulk_CM_IRP_versions(),
   get_notification_categories()

   It should be updated based on the rule of sub-clause
   titled "IRP document version number string".
   */
   const string BULK_CM_IRP_VERSION = "<to be updated using the rule>";

   /*
   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_BULK_CM_LOG_STATE = "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";
};

/*
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 {IdlePhase, DownloadPhase, UploadPhase, ActivationPhase,
               FallbackPhase};

/*
Defines the different substates of a configuration session. This includes
the transition state as well.
*/
enum SubState {Completed, Failed, PartlyRealised, InProgress};

/*
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 <BulkCmIRPConstDefs::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; 3G TS 32.106 Annex H.
   e.g. "g3SubNetwork=10001,g3ManagedElement=400001" identifies an
   G3ManagedElement instance of the object model.
   */
   typedef string DistinguishedName;

   /*
   Optionally used within the upload method to give filter critera
   */
   typedef string FilterType;

   /*
   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 {BaseOnly, BaseNthLevel, BaseSubtree, BaseAll};

   /*
   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 optional (defined by an empty string).
   */
   struct SearchControl
   {
      ScopeType type;
      unsigned long level;
      FilterType filter;     // optional parameter
   };
};

#endif
```

Annex B (normative):
IDL: BulkCmIRPSystem

```
#ifndef BulkCmIRPSystem_IDL
#define BulkCmIRPSystem_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
{
   /*
   System fails to complete the operation.  System can provide reason
   to qualify the exception.  The semantics carried in reason
   is outside the scope of this IRP.
   */
   exception GetBulkCmIRPVersions { string reason; };
   exception ConcurrencyException { string reason; };
   exception IllegalFilterFormatException { string reason; };
   exception IllegalDNFormatException { string reason; };
   exception IllegalScopeTypeException { string reason; };
   exception IllegalScopeLevelException { string reason; };
   exception MaxSubscriberException { string reason; };
   exception NoFallbackException {};
   exception SessionIdInUseException { string reason; };
   exception TransitionStateException { string reason; };
   exception UnknownSubscriberException{ string reason; };
   exception IllegalURLFormatException{ string reason; };
   exception UnknownSessionIdException {};

   /*
   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 (GetBulkCmIRPVersions);


      /*
      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 FilterType
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 (UnknownSessionIdException, TransitionStateException,
        ConcurrencyException,
        IllegalDNFormatException, IllegalFilterFormatException,
        IllegalScopeTypeException, IllegalScopeLevelException);


/*
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 (UnknownSessionIdException, TransitionStateException);

/*
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 boolean fallback
)
raises (UnknownSessionIdException, TransitionStateException,
        ConcurrencyException);

/*
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 (UnknownSessionIdException, ConcurrencyException);

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

/*
Returns the state of a configuration session.
```

```
    */
    BulkCmIRPConstDefs::SessionState get_session_status (
       in BulkCmIRPConstDefs::SessionId session_id,
       out BulkCmIRPConstDefs::ErrorInformation error_information
    )
    raises (UnknownSessionIdException);

    /*
    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 (UnknownSessionIdException, NoFallbackException,
            TransitionStateException, ConcurrencyException);


    /*
    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 (UnknownSessionIdException, TransitionStateException);

    /*
    The IRPManager invokes this operation to abort a configuration sesssion.
    This operation can be called in any state. But 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 (UnknownSessionIdException);

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

#endif
```

Annex C (informative):
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Jun 2001	S_12	SP-010283	--	--	Approved at TSG SA #12 and placed under Change Control	2.0.0	4.0.0