Technical Specification Group Services and System Aspects
Meeting #14, Kyoto, Japan, 17-20 December 2001

TSGS#14(01)0644

Source: **SA5 (Telecom Management)**

Title: **Rel-4 CR 32.613 (S5-010668, S5-010669)**

Document for: **Decision**

Agenda Item: **7.5.3**

Doc-1st-	Spec	CR	R	Phas	Subject	C	Versi	Versi	Doc-2nd-	Workitem
SP-010644	32.613	001		Rel-4	**Correction of a notification name and Addition of missing table for fallback operation**	F	4.0.0	4.1.0	S5-010668	OAM-CM
SP-010644	32.613	002		Rel-4	**Corrections to the exceptions in the Bulk CM IRP CORBA Solution Set**	F	4.0.0	4.1.0	S5-010669	OAM-CM

3GPP TSG-SA5 (Telecom Management) **S5-010668**
Meeting #23, Sundsvall, Sweden 15 - 19 October 2001 **S5C010427rev1**

CR-Form-v4

CHANGE REQUEST

⌘ **32.613**	**CR 001**	⌘ ev	**-**	⌘ Current version:	**4.0.0**	⌘

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

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

Title: ⌘	Correction of a notification name and Addition of missing table for fallback operation		
Source: ⌘	SA5		
Work item code: ⌘	OAM-CM	***Date:*** ⌘	19/10/2001
Category: ⌘	**F**	***Release:*** ⌘	REL-4
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*

Reason for change: ⌘	The specification uses a faulty name for the NotifyGetSessionLogEnded notification and the mapping table for the fallback operation is missing.
Summary of change: ⌘	The notification name is corrected. The mapping table for the fallback operation is added. The reference to the Name Convention specification is changed to the Rel-4 version of the specification. The version of the specification is introduced in IDL part.
Consequences if not approved: ⌘	There will be a mismatch between the IS and CORBA SS specifications. Fallback functionality is missing. The wrong Naming Convention specification will be referenced. The IDL part can not perform the GetBulkCmIRPVersion operation correctly, i.e. the IRP manager can not find out with version the IRPAgent supports.

Clauses affected: ⌘	4.2, 4.3, 4.4 and Annex A.	
Other specs affected: ⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘
Other comments: ⌘		

.

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~~notifyGetSessionLogEnded	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`	M

	ActivationModeException	

Table 7: Mapping from IS `fallback` ~~`Operation`~~ parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
status	exception UnknownSessionIdException, exception NoFallbackException, exception TransitionStateException, exception ConcurrencyException	M

Table ~~7~~8: 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~~9: 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~~10: 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~~11: Mapping from IS `getSessionLog` parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId	M

	session_id	
logFileReference	BulkCmIRPConstDefs::FileDestination sink	M
contentType	boolean only_error_info	M
status	exception UnknownSessionIdException, exception ConcurrencyException	M

Table ~~11~~12: 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~~13: 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 32.612 [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 ~~13~~14: 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_BULK_CM_LOG_STATE~~ 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 parameter	`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.
`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~~15: 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~~16: 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~~32.613
V4.~~0~~1~~>~~";

   /*
   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 ~~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";
   };

   /*
   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~~ 32.300 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
```

3GPP TSG-SA5 (Telecom Management) **S5-010669**
Meeting #23, Sundsvall, Sweden 15 - 19 October 2001 **S5C010369rev1**

CR-Form-v4

CHANGE REQUEST

⌘	**32.613**	**CR**	**002**	⌘	ev	**-**	⌘	Current version: **4.0.0** ⌘

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

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

Title:	⌘	Corrections to the exceptions in the Bulk CM IRP CORBA Solution Set		
Source:	⌘	SA5		
Work item code:	⌘	OAM-CM	***Date:*** ⌘	19/10/2001
Category:	⌘	**F**	***Release:*** ⌘	REL-4
		Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*

Reason for change:	⌘	Corrections to exceptions listed in 32.613 (Bulk CM IRP CORBA Solution Set).
Summary of change:	⌘	The following exceptions are not relevant and should be deleted from the IDL module BulkCmIRPSystem in Annex B: MaxSubscriberException, UnknownSubscriberException. The exception TransitionStateException needs to be added to the download operation in sub-clause 4.3. The exception IllegalURLFormatException should be added to upload, download and getSessionLog operations in sub-clause 4.3 and Annexe B. A new exception, NoActiveOperationException, should be added to the abortSessionOperation in sub-clause 4.3 and Annexe B to align the CORBA Solution Set with the Information Service specification. Delete table 12 which is redundant and align table 11 with the IDL specification.
Consequences if not approved:	⌘	32.613 (CORBA Solution Set for Bulk CM) will contain wrong exceptions and some exceptions will be missing.

Clauses affected:	⌘	4.3, Annex B.		
Other specs affected:	⌘	☐ Other core specifications ☐ Test specifications ☐ O&M Specifications	⌘	
Other comments:	⌘			

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, exception IllegalURLFormatException`	M

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

IS Operation parameter	SS Method parameter	Qualifier
`downloadDataFileRef erence`	`BulkCmIRPConstDefs::FileDestination source`	M
`sessionId`	`BulkCmIRPConstDefs::SessionId session_id`	M

status	exception UnknownSessionIdException, exception TransitionStateException, exception IllegalURLFormatException	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, exception NoActiveOperationException	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	M

	sink	
contentType	boolean only_error_info	M
status	exception UnknownSessionIdException, exception ConcurrencyException, exception IllegalURLFormatException	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~~ ~~S~~exception GetBulkCmIRPVersions	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)~~	

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 {};
   exception NoActiveOperationException {};

   /*
   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,
              IllegalURLFormatException);


      /*
      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,
              IllegalURLFormatException);

      /*
      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,
              IllegalURLFormatException);

      /*
      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 an active operation
      during a configuration sesssion.
      ~~This operation can be called in any state. But it~~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, NoActiveOperationException);

       /*
       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