Technical Specification Group Services and System Aspects
Meeting #18, New Orleans, USA, 9-12 December 2002

TSGS#18(02)0745

Source: **SA5 (Telecom Management)**

Title: **Rel-4 CR 32.613 (Bulk CM IRP CORBA solution set) : Removal of the Concurrency exception in getSessionLog**

Document for: **Approval**

Agenda Item: **7.5.3**

Doc-1st-	Spec	CR	R	Phase	Subject	Cat	Version	Doc-2nd-	Workitem
SP-020745	32.613	007	-	Rel-4	Removal of the Concurrency exception in getSessionLog	F	4.3.0	S5-026871	OAM-CM

3GPP TSG-SA5 (Telecom Management) **S5-026871**
Meeting #31, Atlanta/GEORGIA, USA, 7-11 October 2002

CR-Form-v7

CHANGE REQUEST

⌘	**32.613**	**CR**	**007**	⌘**rev**	**-**	⌘	Current version:	**4.3.0**	⌘

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

Proposed change affects:	UICC apps⌘		ME		Radio Access Network	X	Core Network	X

Title:	⌘	Removal of the Concurrency exception in getSessionLog		
Source:	⌘	SA5		
Work item code:	⌘	OAM-CM	***Date:*** ⌘	11/10/2002
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)* *Rel-6 (Release 6)*

Reason for change:	⌘	It shall be possible to read the session log even if another operation is in progress.
Summary of change:	⌘	The concurrancy exception for the getSessionLog is removed. References to g3SubNetwork, g3ManagedElement and G3ManagedElements has been changed to subNetwork, managedElement and ManagedElement.
Consequences if not approved:	⌘	It is not possible to read the session log if another operation is in progress.

Clauses affected:	⌘	4.3, Annex A and Annex B				
		Y	N			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

Change in Clause 4.3

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 StartSessionException, exception SessionIdInUseException, exception MaxSessionReachedException	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 EndSessionException, exception UnknownSessionIdException, exception NotValidInCurrentStateException	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 UploadException, exception UnknownSessionIdException, exception MaxSessionReachedException, exception NotValidInCurrentStateException, exception ConcurrencyException, exception IllegalDNFormatException, 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
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
downloadDataFileReference	BulkCmIRPConstDefs::FileDestination source	M
status	exception DownloadException, exception UnknownSessionIdException, exception MaxSessionReachedException, exception NotValidInCurrentStateException, 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
fallbackEnabled	boolean fallback	M
status	exception ActivateException, exception UnknownSessionIdException, exception NotValidInCurrentStateException, exception ConcurrencyException	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	exception FallbackException, exception UnknownSessionIdException, exception NoFallbackException, exception NotValidInCurrentStateException, exception ConcurrencyException	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	exception AbortSessionOperationException, exception UnknownSessionIdException, exception NotValidInCurrentStateException	M

Table 9: Mapping from IS getSessionIds parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionIdList	return of type BulkCmIRPConstDefs::SessionIdList	M
status	exception GetSessionIdsException	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 of type BulkCmIRPConstDefs::SessionState	M
Not specified in IS	BulkCmIRPConstDefs::ErrorInformation error_information	M
status	exception GetSessionStatusException, exception UnknownSessionIdException	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	exception GetSessionLogException, exception UnknownSessionIdException, ~~exception ConcurrencyException,~~ exception IllegalURLFormatException	M

Table 12: Mapping from IS getBulkCmIRPVersion parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
versionNumberList	return of type ManagedGenericIRPConstDefs::VersionNumberSet	M
status	exception GetBulkCmIRPVersionsException	M

.

End of Change in Clause 4.3

Change in Clause Annex A

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 = "32.613 V4.1";

   /*
   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_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 32.300 Annex H.
   e.g. "~~g3S~~subNetwork=10001, ~~g3M~~managedElement=400001" identifies an
   ~~G3~~ManagedElement 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
```

End of Change in Annex A

Change in Clause Annex B

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
{
   /*
   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; };

/*
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 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);

   /*
```

```
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 (UploadException, UnknownSessionIdException,
        MaxSessionReachedException, NotValidInCurrentStateException,
        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 (DownloadException, UnknownSessionIdException,
        MaxSessionReachedException, NotValidInCurrentStateException,
        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 (ActivateException, UnknownSessionIdException,
        NotValidInCurrentStateException, 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 (GetSessionLogException, 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 (StartSessionException, SessionIdInUseException,
        MaxSessionReachedException);

/*
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);

/*
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);

/*
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);

/*
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);

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

#endif
```

End of Change in Annex B
End of Document