
Source: SA5 (Telecom Management)
Title: 2 Rel-5 CRs 32.613 (Bulk Configuration Management IRP; CORBA SS)
Document for: Approval
Agenda Item: 7.5.3

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Ver-Current	Doc-2nd-Level	Workitem
SP-030140	32.613	008	-	Rel-5	Add subphases "PreactivationPhase" and "ValidationPhase" in 'BulkCmlRPCConstDefs' IDL definition	F	5.0.0	S5-036060	OAM-NIM
SP-030140	32.613	009	-	Rel-5	Add missing Rel-4 CORBA IDL exceptions	F	5.0.0	S5-036324	OAM-NIM

CHANGE REQUEST

⌘ **32.613 CR 008** ⌘ rev **-** ⌘ Current version: **5.0.0** ⌘

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

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

Title:	⌘ Add subphases "PreactivationPhase" and "ValidationPhase" in 'BulkCmIRPConstDefs' IDL definition		
Source:	⌘ S5		
Work item code:	⌘ OAM-NIM	Date:	⌘ 28/02/2003
Category:	⌘ F	Release:	⌘ Rel-5
	Use <u>one</u> of the following categories:		Use <u>one</u> of the following releases:
	F (correction)	2 (GSM Phase 2)	
	A (corresponds to a correction in an earlier release)	R96 (Release 1996)	
	B (addition of feature),	R97 (Release 1997)	
	C (functional modification of feature)	R98 (Release 1998)	
	D (editorial modification)	R99 (Release 1999)	
	Detailed explanations of the above categories can be found in 3GPP TR 21.900 .		Rel-4 (Release 4)
			Rel-5 (Release 5)
			Rel-6 (Release 6)

Reason for change:	⌘ Corba IDL: BulkCmIRPConstDefs do not include SubPhases "PreactivationPhase" and "ValidationPhase".
Summary of change:	⌘ In Annex A, IDL module BulkCmIRPConstDefs , Subphase enum definition is modified to include two new subphases: PreactivationPhase and ValidationPhase
Consequences if not approved:	⌘ complete interface-N Corba operations can not be used if subphases are not defined correctly

Clauses affected:	⌘ Annex A										
Other specs affected:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;">X</td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;">X</td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;">X</td> <td style="text-align: center;">X</td> </tr> </table>	Y	N	X	X	X	X	X	X	Other core specifications	⌘
Y	N										
X	X										
X	X										
X	X										
		Test specifications									
		O&M Specifications									
Other comments:	⌘										

How to create CRs using this form:

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
{

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

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

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

```

```

    const string PREACTIVATION_COMPLETED = "x15";
};

/*
This block defines all possible values for sessionLogStatus
One of these strings appear in the event_name of the Structured
Event of notifyGetSessionLogEnded notification.
*/
interface LogStateNotification
{
    const string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY = "x1";
    const string GET_SESSION_LOG_COMPLETED_UNSUCCESSFULLY = "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, PreactivationPhase, ValidationPhase};

/*
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. "SubNetwork=10001,ManagedElement=400001" identifies an
G3ManagedElement instance of the object model.
*/
typedef string DistinguishedName;

/*
Used within the upload method to give filter criteria
*/
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 mandatory (The filter will have to be
set to an empty string if it has no other value).
*/
struct SearchControl
{
    ScopeType type;
    unsigned long level;
    FilterType filter;
};

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

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

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

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

```

```
};  
#endif
```

CHANGE REQUEST

⌘ **32.613 CR 009** ⌘ rev **-** ⌘ Current version: **5.0.0** ⌘

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

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

Title:	⌘ Add missing Rel-4 CORBA IDL exceptions		
Source:	⌘ S5		
Work item code:	⌘ OAM-NIM	Date:	⌘ 28/02/2003
Category:	⌘ F	Release:	⌘ Rel-5
	<i>Use <u>one</u> of the following categories:</i> 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 .		<i>Use <u>one</u> of the following releases:</i> 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:	⌘ Release 4 CORBA exceptions are missing from R5 version.
Summary of change:	⌘ Add missing R4 CORBA IDL exceptions
Consequences if not approved:	⌘ R5 CORBA IDL will not compile.

Clauses affected:	⌘ Annex B										
Other specs affected:	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">Y</td> <td style="width: 20px; text-align: center;">N</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;">X</td> </tr> <tr> <td style="text-align: center;"> </td> <td style="text-align: center;">X</td> </tr> </table>	Y	N		X		X		X	Other core specifications Test specifications O&M Specifications	⌘
Y	N										
	X										
	X										
	X										
Other comments:	⌘										

How to create CRs using this form:

Change in 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; };

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

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

/*
System otherwise fails to complete the operation. System can provide reason
to qualify the exception. The semantics carried in reason
is outside the scope of this IRP.
*/
exception GetBulkCmIRPVersionsException { string reason; };
exception UploadException { string reason; };
exception DownloadException { string reason; };
exception ActivateException { string reason; };
exception ValidateException { string reason; };
exception PreactivateException { string reason; };
exception GetBulkCMIRPOperationProfileException { string reason; };
exception GetBulkCMIRPNotificationProfileException { string reason; };
exception GetSessionLogException { string reason; };
exception StartSessionException { string reason; };
exception GetSessionStatusException { string reason; };
exception FallbackException { string reason; };
exception EndSessionException { string reason; };
exception AbortSessionOperationException { string reason; };
exception GetSessionIdsException { string reason; };
```

```

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

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

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

    /*
    Uploads a configuration from the subnetwork. The result is put in a
    configuration data file in an area specified by the IRPManager.
    The MIB of the subnetwork is iterated by means of containment search,
    using a SearchControl to control the search and the returned results.
    All MOs in the scope constitutes a set that the filter works on.
    In case of a concurrent running session the function will
    return an exception. If the value of the given baseObject or 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,
        ManagedGenericIRPSystem::InvalidParameter);

    /*

```

```

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

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

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

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

/*
Returns the state of a configuration session.
*/
BulkCmIRPConstDefs::SessionState get_session_status (
    in BulkCmIRPConstDefs::SessionId session_id,
    out BulkCmIRPConstDefs::ErrorInformation error_information
)

```

```

raises (GetSessionStatusException, UnknownSessionIdException,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Activates 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
   exist at a time for a specific scope of the subnetwork.
2. When a fallback area is available and triggered by this method, the
   previous valid area is replaced with the data stored in
   the fall back area.
If the EM detects that the former configuration has never been
changed it returns an exception because it does not trigger an
activation of the former data.
*/
void fallback (
    in BulkCmIRPConstDefs::SessionId session_id
)
raises (FallbackException, UnknownSessionIdException, NoFallbackException,
        NotValidInCurrentStateException, ConcurrencyException,
        ManagedGenericIRPSystem::InvalidParameter);

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

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

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

/*
Validates previously downloaded bulk configuration data inside a session.
Detects errors in the data prior to requesting preactivation or

```

```
activation.
*/
void validate (
    in BulkCmIRPConstDefs::SessionId session_id,
    in BulkCmIRPConstDefs::ActivationModeTypeOpt activation_mode
)
raises (ValidateException, UnknownSessionIdException,
        NotValidInCurrentStateException, ConcurrencyException,
        IllegalActivationModeException,
        ManagedGenericIRPSystem::ParameterNotSupported,
        ManagedGenericIRPSystem::InvalidParameter,
        ManagedGenericIRPSystem::OperationNotSupported);

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

End of Change in Annex B
End of Document