Technical Specification Group Services and System Aspects
Meeting #17, Biarritz, France, 9-12 September 2002

TSGS#17(02)0486

Source: SA5 (Telecom Management)

Title: Rel-5 CRs 32.611/2/3 (Bulk CM Integration Reference Point (IRP)) - upgrade to Rel-5

Document for: Approval

Agenda Item: 7.5.3

Doc-1st-	Spec	CR	Rev	Phase	Subject	Cat	Version	Doc-	Workite	Relation
SP-020486	32.611	002	-	Rel-5	**Additional Bulk CM IRP requirements for Rel-5**	C	5.0.0	S5-026688	OAM-NIM	**Parent CR**
SP-020486	32.612	003	-	Rel-5	**Add Bulk CM IRP IS Enhancements for Rel-5**	C	4.2.0	S5-026733	OAM-NIM	**Child CR**
SP-020486	32.613	005	-	Rel-5	**Add Bulk CM IRP CORBA Solution Set Enhancements Rel-5**	C	4.2.0	S5-026734	OAM-NIM	**Grandchild CR**

3GPP TSG-SA5 (Telecom Management) **S5-026688**
Meeting #30, Tampere, FINLAND, 19 – 23 August 2002

CR-Form-v5

CHANGE REQUEST

⌘ **32.611** **CR** **002** ⌘ **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: ⌘ (U)SIM [ ] ME/UE [ ] Radio Access Network [X] Core Network [X]

Title:	⌘	Additional Bulk CM IRP requirements for Rel-5
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**C**
Release:	⌘	REL-5

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:	⌘	Addition of requirements for Bulk CM “all or none” activation strategy to ensure that an activation is successful.
Summary of change:	⌘	The requirements for Bulk CM “all or none” activation strategy have been added to sub-clause 4.3.
Consequences if not approved:	⌘	Bulk CM for Release 5 will omit specifying an important aspect regarding activation and checking strategy.

Clauses affected:	⌘	4.3.		
Other specs affected:	⌘	[ ] Other core specifications [ ] Test specifications [X] O&M Specifications	⌘	32.612, 32.613, 32.614 CMIP (CR missing for the time being)
Other comments:	⌘	S5-026688 Rel-5 CR 32611 Additional Bulk CM IRP requirements for Rel-5 - Parent CR S5-026733 Rel-5 CR 32612 Bulk CM IS - R5 Enhancements and New Methodology - Child CR S5-026734 Rel-5 CR 32613 Bulk CM CORBA SS - R5 Enhancements - Grandchild CR		

4.1 Bulk CM principles

The Itf-N (see ref. 3GPP TS 32.102 [2]) is an object oriented interface, i.e. all resources of the 3G network (functional and physical resources) whose management is standardised by the present document are represented as Managed Object Instances (MOI) of a Network Resource Model (NRM).

The NRM shall be highly simplified for the purpose of the NM, based on the assumption that all of the detailed CM actions are performed by an Element Manager (EM), which knows the vendor-specific NRM and configuration.

There are two types of CM functions - *Passive* CM and *Active* CM (see 3GPP TS 32.600 [4] for ~~defintions~~definitions).

There are also at least two approaches to CM - *Basic CM* and *Bulk CM.* Refer to 3GPP TS 32.600 [4] for Basic CM.

Bulk CM is characterised by

- Bulk (file-oriented) data retrieval (configuration parameters) over Interface-N from single NEs, a collection of NEs or the whole network. (The passive aspect of Bulk CM.)
- Bulk (file-oriented) data download of configuration parameters to EM/NEs over Interface-N. (An active aspect of Bulk CM.)
- The network-wide activation of those parameters through a single operation. (An active aspect of Bulk CM.)
- The ability to fallback to a previous stable configuration through a single operation. (An active aspect of Bulk CM.)

This document describes the specific functional requirements related to Bulk CM of Network Resources (NRs) on the Itf-N.

4.2 Overview of IRPs related to Bulk CM

The Itf-N for CM is built up by a number of Integration Reference Points (IRPs) and a related Name Convention, which realise the functional capabilities over this interface. The basic structure of the IRPs is defined in 3GPP TS 32.101 [1] and 3GPP TS 32.102 [2]. For CM, a number of IRPs (and a Name Convention) are defined, used by this as well as other specifications For Telecom Management (TM) produced by 3GPP. All these IRPs are defined in separate 3GPP specifications, and listed in the Introduction clause.

4.3 Bulk CM Requirements

Interface-N shall provide efficient mechanisms to upload current CM data from the IRP Agent and download new CM data to the IRP Agent.
It shall be possible to transfer a CM file containing parameters for any specified Network Resource Model (e.g. radio network, Core Network) from the NM to the IRP Agent using a standardised file format and transfer mechanism. The IRP Agent shall also be capable of making the necessary configuration changes in its managed NEs, using the parameters and information contained in the transferred CM file.
The following requirements have been identified regarding the file format and transfer control mechanism over Interface-N.

1. It shall be possible to initiate the upload (IRP Agent to NM) of CM data over Interface-N.

2. It shall be possible to scope the Objects to be uploaded from the IRP Agent, e.g. parameters for a Cell, an RNC, or all the NEs managed by the IRP Agent.

3. It shall be possible to initiate the download (NM to IRP Agent) of CM data over Interface-N.

4. The parameters in the file for downloading to the IRP Agent may relate to creating Managed Objects, deleting Managed Objects or changing some or all modifiable attributes of existing Managed Objects. These parameters may be applicable to some or all the Managed Objects controlled by the IRP Agent, e.g. a Cell, an RNC, or all NEs managed by the IRP Agent.

5. The IRP Agent should check the consistency, syntax and semantic of the downloaded file to ensure that the configuration changes contained in the file can be implemented in the network.

6. It shall be possible to activate a previously downloaded configuration file in the EM/NE via a control facility.

7. Two activation modes may be defined for this IRP, distinguishable by their impact on network services in a live network. In the first activation mode the IRP Agent shall attempt to keep impact on service to a minimum, e.g. by the IRP Agent executing only one command or configuration change at a time in a NE. In the second activation mode all configuration changes contained in the file shall be implemented in the network in the minimum possible time, regardless of impact on service.

8. Activation shall employ a best effort mechanism, i.e. if parts of the activation cannot be successfully completed, the remainder shall still be attempted, where possible. Optionally, other activation strategies may also be supported.

9. The Bulk CM IRP shall specify an optional capability to achieve as near an "all or none" activation strategy as possible. This strategy may be achieved by the use of a pre-activation operation that provides the maximum possible verification that the subsequent activate operation will succeed.

~~9.~~10. The activation of the new configuration in the NEs shall be logged, the objective being to enable an operator (if necessary) to analyse the log (e.g. analyse failed commands) and to subsequently achieve a full activation. Note that "activation" means execution of each command in the downloaded file, and the result of each command execution shall be logged, whether successful or unsuccessful.

~~10.~~11. It shall be possible to selectively retrieve the information contained in the log, e.g. only unsuccessful operations.

~~11.~~12. It shall be possible to check the status of a configuration file operation.

~~12.~~13. It shall be possible for the IRP Agent to fallback to a previously known working configuration, initiated by the IRP Manager.

~~13.~~14. Interface-N shall support notifications, e.g. to indicate completion of an operation, error cases.

~~14.~~15. The file format shall be flexible enough to include all possible CM parameter types, i.e. standard parameters as well as vendor specific parameters. The meaning, syntax, units, etc. of standard parameters shall be specified. The representation of vendor specific parameters will be proprietary. A uniform mechanism for handling vendor specific parameters shall be specified.

~~15.~~16. Since the files are transferred via a machine-machine interface, the file format shall be machine readable using industry standard tools, e.g. XML parser.

~~16.~~17. Moreover, the files shall be formatted in a human readable way, e.g. to allow manual editing of its contents.

~~17.~~18. The file format shall be specified by using a standardised language, e.g. the Extensible Mark-up Language (XML), in order to provide the possibility of visualisation in a web browser.

~~18.~~19. The same file format shall be used for the upload and download to the IRP Agent.

~~19.~~20. The file format shall be independent of the data transfer protocol used to carry the file from one system to another.

~~20.~~21. The file transfer facility shall be implemented using a commonly available protocol, e.g. FTP.

~~21.~~22. The Managed Object Class identifiers used in the file shall be the same as those used for Basic CM, Fault Management and Performance Management.

~~22.~~23. Bulk CM IRP shall be sufficient to configure a complete radio network, including vendor specific parameters of the UTRAN.

Annex A (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
Jun 2002	S_16	SP-020295	001	--	Adding Bulk CM IRP requirements for Rel-5	4.0.0	5.0.0

3GPP TSG-SA5 (Telecom Management) Meeting #30 **S5-026733**
Tampere, Finland, 19 – 23 August 2002

CR-Form-v7

CHANGE REQUEST

⌘ **32.612** **CR** **003** ⌘ **rev** **-** ⌘ Current version: **4.2.0** ⌘

*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:	⌘	Add Bulk CM IRP IS Enhancements for Rel-5
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**C**
Release:	⌘	REL-5

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:	⌘	Support of Validation, Preactivation, Modes
Summary of change:	⌘	Add new operations validation and preactivate. Add Mode parameters. Other changes.
Consequences if not approved:	⌘	Valiadtion, Preactivation, Modes not supported.

Clauses affected:	⌘	2, 4,6,7,8,9,10, Annex A and B

		Y	N		
Other specs affected:	⌘		X	Other core specifications	
			X	Test specifications	
		X		O&M Specifications	32.613, 32.614 CMIP (CR missing for the time being)

Other comments:	⌘	S5-026688	Rel-5 CR 32611 Additional Bulk CM IRP requirements for Rel-5 - Parent CR
		S5-026733	Rel-5 CR 32612 Bulk CM IS - R5 Enhancements and New Methodology - Child CR
		S5-026734	Rel-5 CR 32613 Bulk CM CORBA SS - R5 Enhancements - Grandchild CR

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 by 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 actions on single NEs of the 3G network or as part of a complex procedure involving actions on many resources/objects in one or several NEs.

1 Scope

The present document (Bulk Configuration Management IRP: Information Service) defines an Integration Reference Point (IRP) through which an 'IRPAgent' (typically an Element Manager or Network Element) can communicate bulk Configuration Management related information to one or several 'IRPManagers' (typically Network Managers).

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.

[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 32.302: "Telecommunication Management; Notification Management; Part 2: Notification Integration Reference Point; Information Service".

[4] 3GPP TS 32.622: "3G Configuration Management: Generic Network Resources IRP: NRM".

[5] 3GPP TS 32.642: "3G Configuration Management: UTRAN Network Resources IRP: NRM".

[6] 3GPP TS 32.652: "3G Configuration Management: GERAN Network Resources IRP: NRM".

[7] 3GPP TS 32.300: "Name Convention for Managed Objects".

[8] 3GPP TS 32.600: "3G Configuration Management: Concepts and requirements".

[9] 3GPP TS 32.312: "Generic IRP Management: Information Service".

[10] 3GPP TS 32.632: “3G Configuration Management: Core Network Resource IRP: NRM”

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply. For terms and definitions not found here, please refer to 3GPP TS 32.101 [1], 3GPP TS 32.102 [2] and 3GPP TS 32.600 [8].

Association: In general it is used to model relationships between Managed Objects. Associations can be implemented in several ways, such as:

(1) name bindings,

(2) reference attributes, and

(3) association objects.

This IRP stipulates that containment associations shall be expressed through name bindings, but it does not stipulate the implementation for other types of associations as a general rule. These are specified as separate entities in the object models (UML diagrams). Currently (in R99) however, all (non-containment) associations are modelled. by means of reference attributes of the participating MOs.

Data: is any information or set of information required to give software or equipment or combinations thereof a specific state of functionality.

Element Manager (EM): provides a package of end-user functions for management of a set of closely related types of Network Elements (NEs). These functions can be divided into two main categories:

- *Element Management Functions* for management of NEs on an individual basis. These are basically the same functions as supported by the corresponding local terminals.
- *Sub-Network Management Functions* that are related to a network model for a set of NEs constituting a clearly defined sub-network, which may include relations between the NEs. This model enables additional functions on the sub-network level (typically in the areas of network topology presentation, alarm correlation, service impact analysis and circuit provisioning).

IRP: See 3GPP TS 32.101 [1].

IRP Information Service (IS): See 3GPP TS 32.101 [1].

IRP Network Resource Model (NRM): See 3GPP TS 32.101 [1].

IRP Solution Set (SS): See 3GPP TS 32.101 [1].

Managed Element (ME): An instance of the Managed Object Class G3ManagedElement/ManagedElement.

Managed Object (MO): In the context of the present document, a Managed Object (MO) is a software object that encapsulates the manageable characteristics and behaviour of a particular Network Resource. The MO is instance of a MO class defined in a MIM/NRM. An MO class has attributes that provide information used to characterize the objects that belong to the class (the term "attribute " is taken from TMN and corresponds to a "property " according to CIM). Furthermore, a MO class can have operations that represent the behaviour relevant for that class (the term "operation " is taken from TMN and corresponds to a "method " according to CIM). An MO class may support notifications that provide information about an event occurrence within a network resource.

Managed Object Class (MOC): a description of all the common characteristics for a number of MOs, such as their attributes, operations, notifications and behaviour.

Managed Object Instance (MOI): an instance of a MOC, which is the same as a MO as described above.

Management Information Base (MIB): A MIB is an instance of an NRM and has some values on the defined attributes and associations specific for that instance. In the context of the present document , a MIB consist of (1) a Name space (describing the MO containment hierarchy in the MIB through Distinguished Names), (2) a number of Managed Objects with their attributes and (3) a number of Associations between these MOs. Also note that TMN (X.710 [7]) defines a concept of a Management Information Tree (also known as a Naming Tree) that corresponds to the name space (containment hierarchy) portion of this MIB definition. The following figure depicts the relationships between a Name space and a number of participating MOs (the shown association is of a non-containment type)

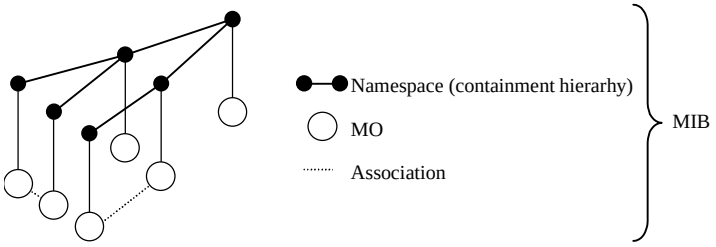


Figure 1: Relationships between a Name space and a number of participating MOs

Management Information Model (MIM): Also referred to as NRM – see the definition below. There is a slight difference between the meaning of MIM and NRM – the term MIM is generic and can be used to denote any type of management model, while NRM denotes the model of the actual managed telecommunications Network Resources (NRs).

Name space: A name space is a collection of names. The IRP name convention [7] restricts the name space to a hierarchical containment structure, including its simplest form - the one-level, flat name space. All Managed Objects in a MIB shall be included in the corresponding name space and the MIB/name space shall only support a strict

hierarchical containment structure (with one root object). A Managed Object that contains another is said to be the superior (parent); the contained Managed Object is referred to as the subordinate (child). The parent of all MOs in a single name space is called a Local Root. The ultimate parent of all MOs of all managed systems is called the Global Root.

Network Element (NE): is a discrete telecommunications entity, which can be, managed over a specific interface e.g. the RNC.

Network Manager (NM): provides a package of end-user functions with the responsibility for the management of a network, mainly as supported by the EM(s) but it may also involve direct access to the NEs. All communication with the network is based on open and well-standardised interfaces supporting management of multi-vendor and multi-technology NEs.

Network Resource (NR): is a component of a NE, which can be identified as a discrete separate entity and is in an object oriented environment for the purpose of management represented by an abstract entity called Managed Object (MO).

Network Resource Model (NRM): a model representing the actual managed telecommunications Network Resources (NRs) that a System is providing through the subject IRP. An NRM describes Managed Object Classes (MOC), their associations, attributes and operations. The NRM is also referred to as "MIM" (see above) which originates from the ITU-T TMN.

Operator: is either

- a human being controlling and managing the network; or
- a company running a network (the 3G network operator).

3.2 Abbreviations

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

CM	Configuration Management
CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture
EM	Element Manager
FM	Fault Management
IRP	Integration Reference Point
ITU-T	International Telecommunication Union, Telecommunication Standardisation Sector
MIB	Management Information Base
MIM	Management Information Model
MO	Managed Object
MOC	Managed Object Class
MOI	Managed Object Instance
NE	Network Element
NM	Network Manager
NR	Network Resource
NRM	Network Resource Model
PM	Performance Management
SS	Solution Set
SW	Software
TM	Telecom Management
UML	Unified Modelling Language (OMG)
UMTS	Universal Mobile Telecommunications System
XML	EXtensible Markup Language

4 System Overview

4.1 System Context

~~Figure 2 and Figure 3 identify system contexts of the subject IRP in terms of its implementation called IRPAgent and the user of the IRPAgent, called IRPManager. For a definition of IRPManager and IRPAgent, see 3GPP TS 32.102 [2].~~

~~The IRP Agent implements and supports the Bulk CM IRP. The IRP Agent shall be an Element Manager (EM) or a mediator that interfaces to several NE (see Figure 2)or it can be a Network Element (NE) (see Figure 3). In the former case, the interfaces (represented by the a thick dotted line) between the EM and the NEs are not subject of this IRP.~~

~~An IRPManager using this IRP shall choose one of the two System Contexts defined here, for each NE. For instance, if an EM is responsible for managing a number of NEs, the NM shall access this IRP through the EM and not directly to those NEs. For another IRP though, the System Context may be different. For Bulk CM IRP its judged System A in most application is most appropriate, but this does not preclude use of System B when the need is appropriate.~~

~~For another IRP the System Context may be different.~~

Figure 4.1 and 4.2 identify system contexts of the IRP defined by the present specification in terms of its implementation called IRPAgent and the user of the IRPAgent, called IRPManager. For a definition of IRPManager and IRPAgent, see 3GPP TS 32.102 [2].

The IRPAgent implements and supports this IRP. The IRPAgent can reside in an Element Manager (EM) or a Network Element (NE) (see also [2] clause 8). In the former case, the interfaces (represented by a thick dotted line) between the EM and the NEs is not the subject of this IRP.

An IRPManager using this IRP shall choose one of the two System Contexts defined here, for each NE. For instance, if an EM is responsible for managing a number of NEs, the NM shall access this IRP through the EM and not directly to those NEs. For another IRP though, the System Context may be different.

As indicated in Figure 2 and Figure 3,the subject IRP needs to be complemented with the Notification IRP 3GPP TS 32.302 [3]. (This is to allow the IRP Manager to subscribe and unsubscribe to notifications issued by the IRP Agent).

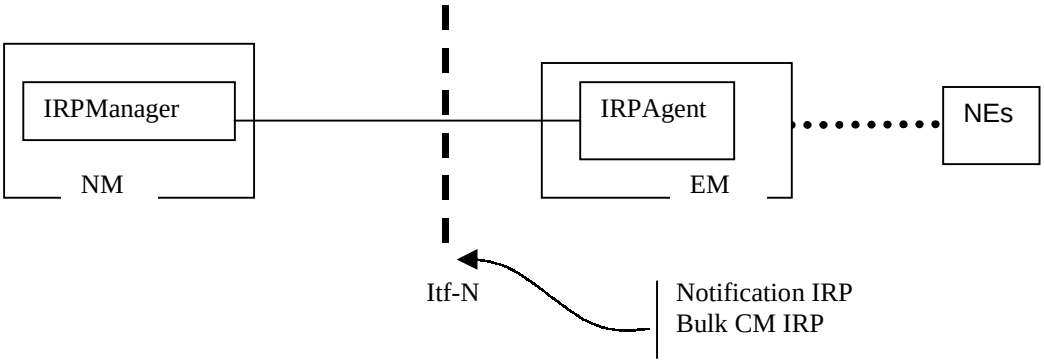


Figure 2: System Context A

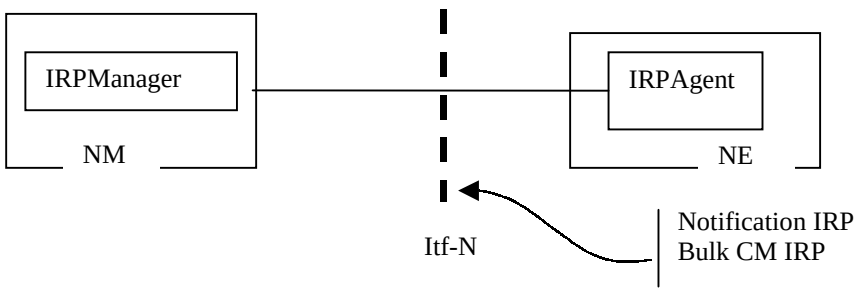


Figure 3: System Context B

4.2 Compliance rules

For general definitions of compliance rules related to qualifiers (Mandatory/Optional/Conditional) for *operations*, *notifications and parameters* (of operations and notifications) please refer to 3GPP TS 32.102 [2].

The following defines the meaning of Mandatory and Optional attributes and associations for Operations, in Solution Sets to the Bulk CMIRP:

- The IRPManager shall support all mandatory attributes/associations. The IRPManager shall be prepared to receive information related to mandatory as well as optional attributes/associations without failure; however the IRPManager does not have to support handling of the optional attributes/associations.
- The IRPAgent shall support all mandatory attributes/associations. It may support optional attributes/associations.

An IRPAgent that incorporates vendor-specific extensions must support normal communication with a 3GPP SA5-compliant IRPManager with respect to all mandatory and optional managed object classes, attributes, associations, operations, parameters and notifications without requiring the IRPManager to have any knowledge of the extensions.

Given that

- rules for vendor-specific extensions remain to be fully specified, and
- many scenarios under which IRPManager and IRPAgent interwork may exist,

it is recognised that in R4/R5 the IRPManager, even though it is not required to have knowledge of vendor-specific extensions, may be required to be implemented with an awareness that extensions can exist and behave accordingly.

4.3 Scope of Bulk CM Management Specification

Within the scope of this document it is specified how Bulk CM IRP IS allows an IRPManager to actively configure NEs over interface-N using an IRPAgent supporting Bulk CM IRP IS. It is not within the scope of this document to specify how Bulk CM IRP IS and the IRPAgent shall resolve any potentially conflicting CM management activities that could arise from either multiple concurrent active IRPManager management Bulk CM IRP sessions, any other IRP conflicting CM management activities, or any CM management activities outside of the scope of an IRP and interface-N. From a system perspective such potential conflicts need to be guarded against, but how this done e.g. operational procedures or implementation specific recovery in an IRPManager or IRPAgent, is beyond the scope of this document.

5 Modelling approach

This clause identifies the modelling approach adopted and used in this IRP.

The modelling approach adopted and used in this IRP is the same as that defined in 3GPP TS 32.622: "Generic Network Resources IRP: NRM" [4].

6 Information Object Classes

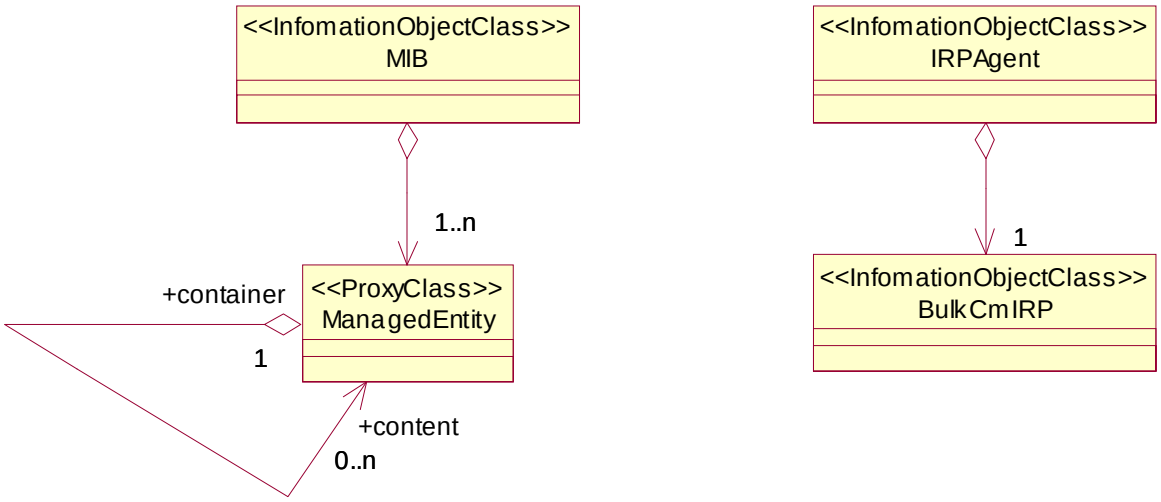
6.1 Information entities imported and local label

Label reference	Local label
32.622, information object class, Top	Top
32.302 [3], information object class, NotificationIRP	NotificationIRP
32.302 [3], interface, notificationIRPNotification	notificationIRPNotification
32.622 [4], [information object class, `GenericIRP`	`GenericIRP`
32.622 [4], information object class, IRPAgent	IRPAgent
32.312 [9], information object class, ManagedGenericIRP	ManagedGenericIRP

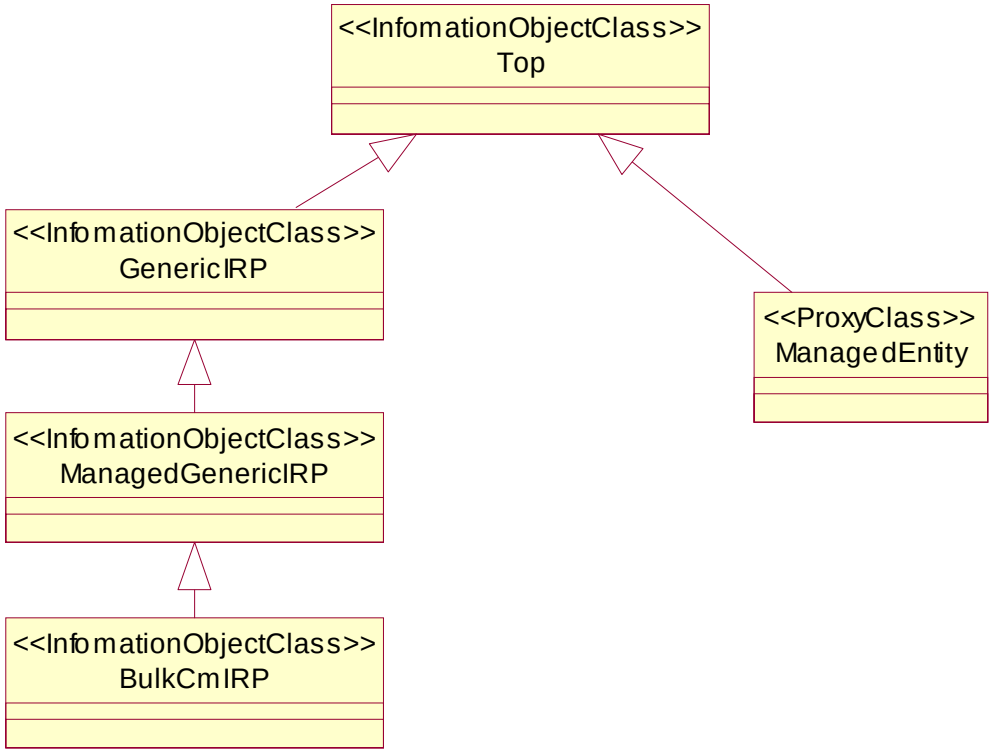
6.2 Class diagram

This sub-clause introduces the set of information object classes (IOCs) that encapsulate information within the `IRPAgent`. The intent is to identify the information required for the BulkCMIRP Agent implementation of its operations and notification emission. This sub-clause provides the overview of all support object classes in UML. Subsequent sub-clauses provide more detailed specification of various aspects of these support object classes.

6.2.1 Attributes and relations



6.2.2 Inheritance



6.3 Information object classes definition

6.3.1 BulkCMIRP

6.3.1.1 Definition

BulkCMIRP is the representation of the configuration management capabilities specified by this specification. This IOC inherits from ManagedGenericIRP IOC specified in TS.32.312 [9].

6.4 ~~6.3~~ Network Resource Model (NRM)

NRMs for Bulk CM IRP are defined in other Network Resource IRP documents of CM, For Bulk CM IRP IS these are:

32.622: "3G Configuration Management: Generic Network Resources IRP: NRM" [4],

32.632: "3G Configuration Management: Core Network Resource IRP: NRM" [10],

32.642: "3G Configuration Management: UTRAN Network Resources IRP: NRM" [5],

32.652: "3G Configuration Management: GERAN Network Resources IRP: NRM" [6].

These NRM documents define all the MOCs and attributes that can be configuration managed by Bulk CM IRP IS.

~~IRP Information Service~~

~~6.1 Introduction~~

7 Interface Definition

~~As already introduced in the previous clause, the present clause defines the Bulk CM IRP Information Service in the form of the IRP Information Service.~~

~~The corresponding Solution Set and Data Format documents provide protocol dependent object model solutions. They provide the actual realization of the operations and notifications defined in this subclause in each protocol environment. One may find that the operation names and operation parameters defined in this protocol neutral model differ from those defined in the Solution Sets.~~

~~6.2 IRP Information Service~~

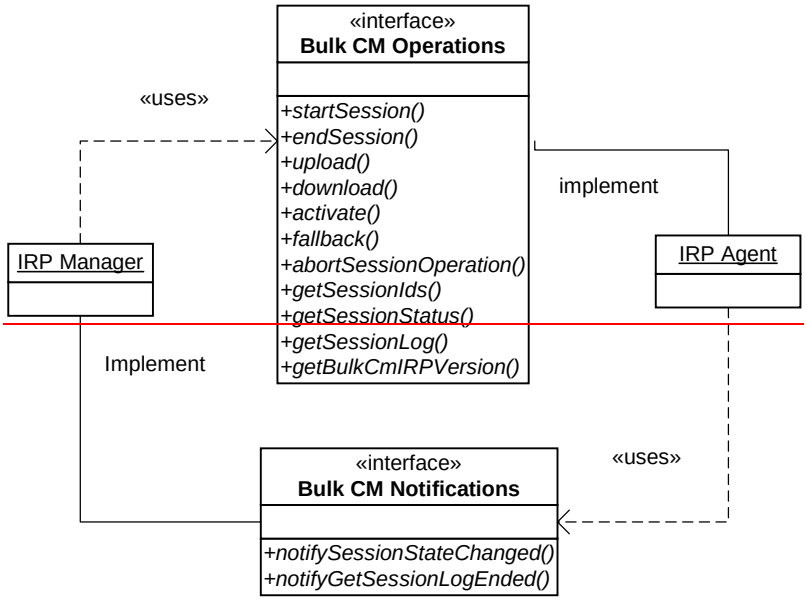
~~This subclause specifies the *operations* and *notifications* that are visible over this IRP. These operations are generic in the sense that they do not specify the MOs that are retrieved/manipulated over the interface.~~

~~6.2.1 Interfaces~~

~~Figure 5~~Clause 7.1 illustrates the operations and notifications defined as interfaces implemented and used by IRPAgent and IRPManager, described using UML notation (Interface in IRP Information Model is identical to concepts conveyed by stereotype <<interface>> of UML). Parameters and return status are not indicated.

~~Two interfaces are defined. One is called `BulkCmIRPOperations`. This interface defines operations implemented by IRPAgent and used (or called) by IRPManager. The other is called `BulkCmIRPNotifications`. This interface defines notifications implemented by IRPManager and used by IRPAgent.~~

The interfaces support multiple IRPManagers connected to an IRPAgent.



Configuration data files defined in clause ~~8~~ 10 define bulk configuration management changes. The following configuration data file handling operations exist in the Itf-N.

- startSession
- endSession
- upload
- download
- validate

- preactivate
- activate
- fallback
- abortSessionOperation
- getSessionIds
- getSessionStatus
- getSessionLog
- getBulkCmIRPVersion

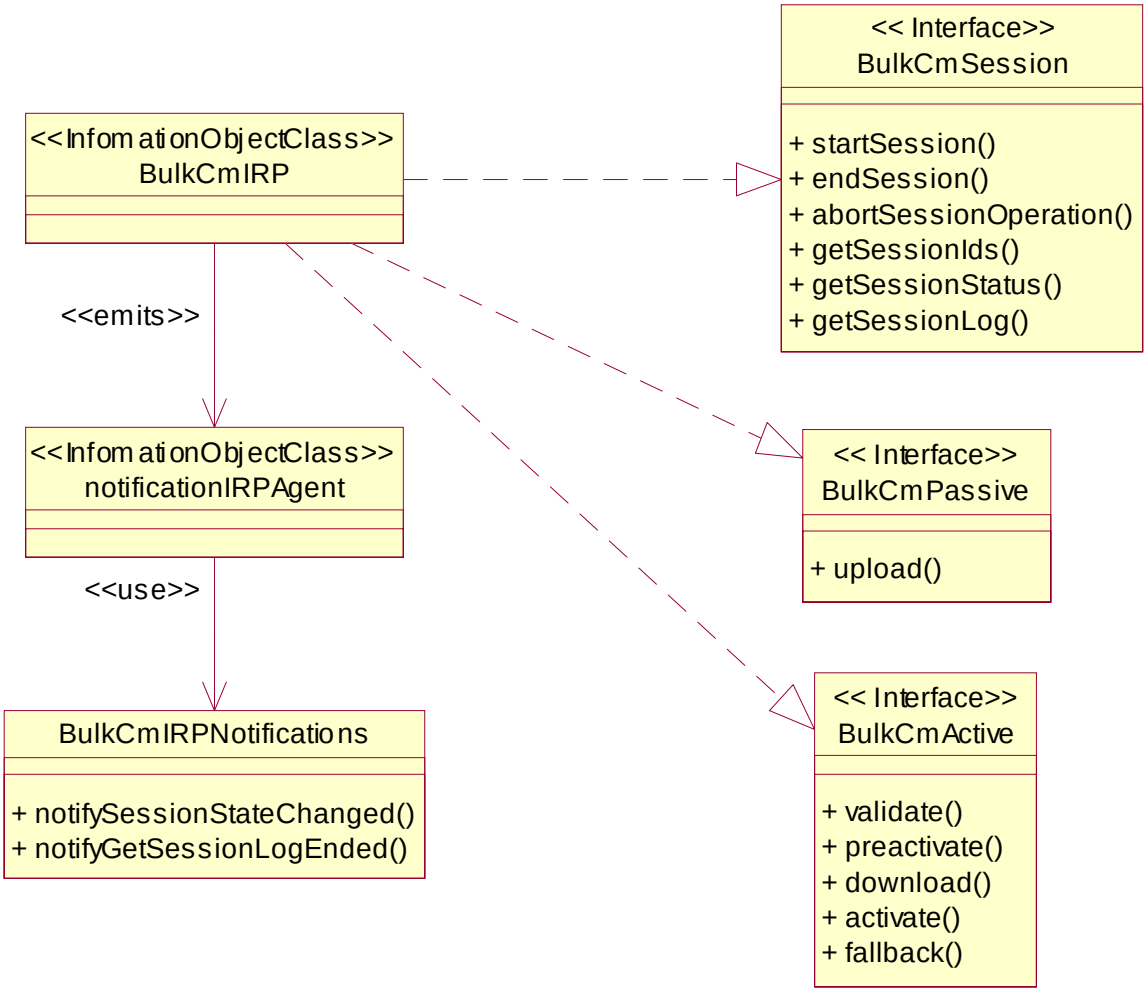
Notification IRP [3] related operations are also associated with Bulk CM IRP (e.g. Subscribe an Unsubscribe), but these operations are described in 32.302: "Telecommunication Management; Notification Management: Part 2: Notification IRP; Information Service " [3].).

The operations, `upload`, `download`, `validate`, `preactivate`, `activate`, `fallback` and `getSessionLog` are performed asynchronously in that when the operations are initiated, the IRPAgent returns an indication that the requested activity has begun, and the IRPManager may release and continue with other tasks. If the IRPManager has subscribed on event notifications, then the IRPManager will receive a notification when the task requested in the operation is complete.

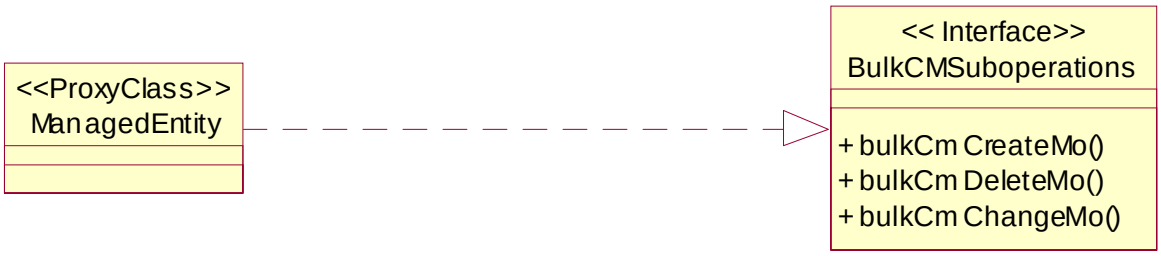
The operations `startSession`, `endSession`, `abortSessionOperation`, `getSessionIds`, `getSessionStatus` and `getBulkCmIRPVersion` are performed synchronously in that the result of the operation is returned as a callback to the operation, and the IRPManager will wait until the response is received before continuing. Refer to subclause 4.3 for system conditions that need to be potentially managed, but are outside the scope of this document.

7.1 Class Diagram

7.1.1 Main Operations and Notifications



7.1.2 Suboperations of clause 10



~~Figure 4: UML Interface Class Diagram~~

~~6.2.2 Bulk CM Operations~~

7.2 Generic rules

- rule 1 : each operation with at least one input parameter supports a pre-condition valid_input_parameter which indicates that all input parameters shall be valid with regards to their information type. Additionally, each such operation supports an exception operation_failed_invalid_input_parameter which is raised when pre-condition valid_input_parameter is false. The exception has the same entry and exit state.

- rule 2 : Each operation with at least one optional input parameter supports a set of pre-conditions supported_optional_input_parameter_xxx where "xxx" is the name of the optional input parameter and the pre-condition indicates that the operation supports the named optional input parameter. Additionally, each such operation supports an exception operation_failed_unsupported_optional_input_parameter_xxx which is raised when (a) the pre-condition supported_optional_input_parameter_xxx is false and (b) the named optional input parameter is carrying information. The exception has the same entry and exit state.

- rule 3 : each operation shall support a generic exception operation_failed_internal_problem which is raised when an internal problem occurs and that the operation cannot be completed. The exception has the same entry and exit state.

7.3 bulkCmSession Interface

7.3.1 Operation ~~6.2.2.1~~ startSession (M)

7.3.1.1 Definition

The IRPManager invokes this operation to start a session state machine and initialise temporary entities to be related with bulk data configuration sessionId in the IRPAgent.

7.3.1.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
sessionId	M	String identifying the session	Identifies the new session and process to be associated with a bulk data operation e.g. upload or download.

7.3.1.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) operation is successful and (b) operation failed because of specified or unspecified reasons

7.3.1.4 Pre-condition

sessionIdNotInUse

Assertion Name	Definition
sessionIdNotInUse	No state, see clause 9. The supplied sessionId is not already open in the Bulk CM IRP Agent.

7.3.1.5 Post-condition

sessionStarted

Assertion Name	Definition
sessionStarted	State = IDLE, see clause 9. The Bulk CM IRP Agent has sucessfully opened the session and is ready to handle other operations associated with the session.

7.3.1.6 Exceptions

7.3.1.6.1 operationFailed

Exception Name	Definition
`operationFailed`	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 1: startSession parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~SessionId~~	~~Input, M~~	~~Identifies the new session and process to be associated with a bulk data operation e.g. upload or download.~~
~~Status~~	~~Output, M~~	~~indicates (a) operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.3.2 Operation ~~6.2.2.2~~ endSession (M)

7.3.2.1 Definition

The IRPManager invokes this operation to end a session state machine and delete all temporary entities and their related bulk data configuration for a specified sessionId in the IRPAgent. If a preactivation had been invoked, endSession should release any internal local resources allocated for the preactivation. The deletion will be rejected if the configuration state is in a working state: e.g. uploading (including getting a log), downloading or activating.

7.3.2.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
`sessionId`	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download

7.3.2.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) operation is successful and (b) operation failed because of specified or unspecified reasons

7.3.2.4 Pre-condition

sessionInStableState

Assertion Name	Definition
sessionInStableState	The supplied sessionId is open in the Bulk CM IRP Agent and in not in a transition status as defined in clause 9, table 1.

7.3.2.5 Post-condition

sessionEnded

Assertion Name	Definition
sessionIdNotInUse	No state, see clause 9. The session is closed and the sessionId is no longer in use.

7.3.2.6 Exceptions

7.3.2.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 2: endSession parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download~~
~~status~~	~~Output, M~~	~~indicates (a) operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.3.3 Operation abortSessionOperation (M)

7.3.3.1 Definition

An IRPManager invokes this operation to request an IRPAgent to abort a currently activate asynchronous operation. The abort will cause the session state machine to exit the current state and enter a new state, see clause ~~7~~9.

7.3.3.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
sessionId	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the abort is required.

7.3.3.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of abort operation is successful and (b) abort operation failed because of specified or unspecified reasons

7.3.3.4 Pre-condition

operationInProgress

Assertion Name	Definition
operationInProgress	The supplied sessionId is open in the Bulk CM IRP Agent and an operation is in an ‘in progress’ state as defined in clause 9, table 1.

7.3.3.5 Post-condition

operationAborted

Assertion Name	Definition
operationAborted	State changed from ‘in progress’ to state as a function of the original state as defined in clause 9.

7.3.3.6 Exceptions

7.3.3.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

7.3.4 Operation ~~6.2.2.8~~ getSessionIds (M)

7.3.4.1 Definition

An IRPManager invokes this operation to request an IRPAgent to return a list of all its currently open sessionIds.

7.3.4.2 Input parameters

None.

7.3.4.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
sessionIdList	M	List of strings identifying sessions	A list of all the sessionIds an IRPAgent currently has open i.e. started with startSession and not ended with endSession operations.
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) operation is successful and (b) operation failed because of specified or unspecified reasons

7.3.4.4 Pre-condition

None

7.3.4.5 Post-condition

None.

~~**Table 8: getSessionIds parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionIdList~~	~~Output, M~~	~~A list of all the sessionIds an IRPAgent currently has open i.e. started with startSession and not ended with endSession operations.~~
~~status~~	~~Output, M~~	~~indicates (a) operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.3.5 Operation ~~6.2.2.9~~ getSessionStatus (M)

7.3.5.1 Definition

The IRPManager invokes this operation to request the IRPAgent to send the current state of the bulk ~~data~~ configuration data file operation. The IRPAgent returns the current state. See clause ~~7~~9.

This operation can be invoked in any session state and does not change the session state.

7.3.5.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
sessionId	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current status is required.

7.3.5.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
sessionState	M	List of ENUM(Upload In Progress, Upload Failed, Upload Completed, Down Load In Progress, Download Failed, Download Completed, Validation In Progress, Validation Failed, Validation Completed, Preactivation In Progress, Preactivation Failed, Preactivation Partly Realised, Preactivation Completed, Activation In Progress, Activation Failed, Activation Partly Realised, Activation Completed, Fallback In Progress, Fallback Failed, Fallback Partly Realised, Fallback Completed)	Indicates current state of the configuration data file operation. See clause ~~7~~9, i.e. will be one of: Upload In Progress, Upload Failed, Upload Completed, Down Load In Progress, Download Failed, Download Completed, Validation In Progress, Validation Failed, Validation Completed, Preactivation In Progress, Preactivation Failed, Preactivation Partly Realised, Preactivation Completed, Activation In Progress, Activation Failed, Activation Partly Realised, Activation Completed, Fallback In Progress, Fallback Failed, Fallback Partly Realised, Fallback Completed.
status	M		Indicates (a) start of operation is successful or (b) operation failed because of specified or unspecified reasons

7.3.5.4 Pre-condition

knownSessionID

Assertion Name	Definition
knowSessionID	Session has been sucessfully started (clause 7.3.1) and not ended (clause 7.3.2).

7.3.5.5 Post-condition

None.

7.3.5.6 Exceptions

7.3.5.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 9: getSessionStatus parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current status is required.~~
~~sessionState~~	~~Output, M~~	~~Indicates current state of the configuration file operation. See clause 7, i.e. will be one of: Upload In Progress, Upload Failed, Upload Completed, Down Load In Progress, Download Failed, Download Completed, Activation In Progress, Activation Failed, Activation Partly Realised, Activation Completed, Fallback In Progress, Fallback Failed, Fallback Partly Realised, Fallback Completed,~~
~~status~~	~~Output, M~~	~~Indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.3.6 Operation ~~6.2.2.10~~ getSessionLog (M)

7.3.6.1 Definition

An IRPManager invokes this operation to request an IRPAgent to provide a log of the results from activities associated with bulk data configuration file sessionId operations.

This operation can be invoked in any session state and does not change the session state.

7.3.6.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
sessionId	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current log is required.
logFileReference	M	String of complete path of file and name.	Specifies the address and file name where the result is to be placed in the IRPManager.
contentType	M	Boolean	Identifies if retrieved file should include (a) complete log including errors, (b) only errors.

7.3.6.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded,	Indicates (a) start of operation is successful and (b) operation failed because of specified

Parameter Name	Qualifier	Matching Information	Comment
		OperationFailed).	or unspecified reasons

7.3.6.4 Pre-condition

knownSessionID

Assertion Name	Definition
knowSessionID	Session has been sucessfully started (clause 7.3.1) and not ended (clause 7.3.2).

7.3.6.5 Post-condition

sessionLogWrite

Assertion Name	Definition
sessionLogWrite	The Bulk CM IRP Agent will beging to write contents of log to the specified address and file.

7.3.6.6 Exceptions

7.3.6.6.1 operationFailed

Exception Name	Definition
`operationFailed`	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 10: getSessionLog parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current log is required.~~
~~LogFileReference~~	~~Input, M~~	~~Specifies the address and file name where the result is to be placed in the IRPManager.~~
~~contentType~~	~~Input, M~~	~~Identifies if retrieved file should include (a) complete log including errors, (b) only errors.~~
~~status~~	~~Output, M~~	~~Indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.4 bulkCmPassive Interface

7.4.1 Operation ~~6.2.2.3~~ upload (M)

7.4.1.1 Definition

An IRPManager invokes this operation to request the IRPAgent to create a file containing bulk configuration data (clause ~~8~~10) and transfer the file to the indicated globally unique data file reference.

7.4.1.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
SessionId	M	String identifying the session	Identifies this specific session and process associated with the requested bulk data upload.
UploadDataFileReference	M	String of complete path of file and name.	This specifies a globally unique file reference to where the specified scope of bulk data is to be uploaded and stored .
BaseObjectInstance	M	DistinguishedName	The MO where the search starts. This is a full Distinguished Name according to 3GPP TS 32.300 [7].
Scope	M	SEQUENCE < ENUM { BASE_OBJECT_ONLY, NTH_LEVEL_SUBORDINATES, BASE_NTH_LEVEL, BASE_ALL}, Integer>	This parameter defines how many levels of the containment hierarchy to search (i.e. apply the filter defined below). The search starts from the MO given by the baseObjectInstance parameter. The levels of search that may be performed are: the base object alone (default); the n-th level subordinates of the base object; the base object and all of its subordinates down to and including the n-th level; the base object and all of its subordinates.
filter	M	See comment.	This parameter defines a filter test to be applied to the scoped Managed Object(s). If the filter is empty, all of the managed objects included by the scope are selected. The actual syntax and capabilities of the filter is Solution Set specific. However, each Solution Set support a filter consisting of one or several assertions that may be grouped using the logical operators AND, OR and NOT. Each assertion is a logical expression of attribute existence, attribute value comparison (“equal to X, less than Y” etc.) and MO Class.

7.4.1.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons

7.4.1.4 Pre-condition

sessionIdle

Assertion Name	Definition
sessionIdle	State as defined in clause 9. The Bulk CM IRP Agent has sucessfully opened the session and is ready to handle the first operations of the session or repeat this operation.

7.4.1.5 Post-condition

uploadInProgress

Assertion Name	Definition
Upload in progress	State = UPLOAD_IN_PROGRESS, as defined in clause 9. The Bulk CM IRP Agent has successfully started the upload of the request configuration data.

7.4.1.6 Exceptions

7.4.1.6. 1 operationFailed

Exception Name	Definition
`operationFailed`	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 3 : upload parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with the requested bulk data upload.~~
~~uploadDataFileReference~~	~~Input, M~~	~~This specifies a globally unique file reference to where the specified scope of bulk data is to be uploaded and stored .~~
~~BaseObjectInstance~~	~~Input, M~~	~~The MO where the search starts. This is a full Distinguished Name according to 3GPP TS 32.300 [7].~~
~~scope~~	~~Input, M~~	~~This parameter defines how many levels of the containment hierarchy to search (i.e. apply the filter defined below). The search starts from the MO given by the baseObjectInstance parameter. The levels of search that may be performed are: the base object alone (default); the n-th level subordinates of the base object; the base object and all of its subordinates down to and including the n-th level; the base object and all of its subordinates.~~
~~filter~~	~~Input, M~~	~~This parameter defines a filter test to be applied to the scoped Managed Object(s). If the filter is empty, all of the managed objects included by the scope are selected. The actual syntax and capabilities of the filter is Solution Set specific. However, each Solution Set support a filter consisting of one or several assertions that may be grouped using the logical operators AND, OR and NOT. Each assertion is a logical expression of attribute existence, attribute value comparison ("equal to X, less than Y " etc.) and MO Class.~~
~~status~~	~~Output, M~~	~~indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.5 bulkCmActive Interface

7.5.1 Operation ~~6.2.2.4~~ download (M)

7.5.1.1 Definition

An IRPManager invokes this operation to request an IRPAgent to download and administer a file containing bulk configuration data (clause ~~8~~10). The IRPAgent obtains the configuration data file ~~data~~ from the indicated globally unique data file reference.

For checks made during download see subclause 7.5.6.

7.5.1.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
sessionId	M	String identifying the session	Identifies this specific session and process associated with the requested bulk data download.
downloadDataFileReference	M		This specifies a globally unique file reference from where the data to be fetched and download from.

7.5.1.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of operation is successful ~~and~~or (b) operation failed because of specified or unspecified reasons

7.5.1.4 Pre-condition

sessionIdle

Assertion Name	Definition
sessionIdle	State as defined in clause 9. The Bulk CM IRP Agent has sucessfully opened the session and is ready to handle the first operations of the session or repeat this operation.

7.5.1.5 Post-condition

downLoadInProgress

Assertion Name	Definition
downLoadInProgress	State = UDOWNLOAD_IN_PROGRESS, as defined in clause 9. The Bulk CM IRP Agent has successfully started the download of the configuration data changes.

7.5.1.6 Exceptions

7.5.1.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 4: download parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with the requested bulk data download.~~
~~downloadDataFileReference~~	~~Input, M~~	~~This specifies a globally unique file reference from where the data to be fetched and download from.~~
~~status~~	~~Output, M~~	~~indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.5.2 Operation validate (O)

7.5.2.1 Definition

An IRPManager invokes this operation to request an IRPAgent to validate previously downloaded bulk configuration data (clause.10), see subclause 7.5.1. Use of this optional operation enables an IRPManager to detect errors with regard to the previously downloaded bulk configuration data before requesting preactivation or activation. See subclause ~~6.2.4~~ 7.5.6 for scope and types of errors attempted to be detected.

Specifying an activation mode is optional. There can only be one activation mode for a session. If an activation mode is specified for the validate, it shall be when the first validate operation is requested. If an activation mode was specified for the first validate operation, it is not possible to change the activation mode initially specified with any subsequent validate retries. (If another activation mode is required; a new session, download, validate, preactivate and activate should be started.). If no activation mode is specified for the first validate, it cannot be subsequently specified with any subsequent validate retries. (If specification of an activation mode is required; a new session, download, validate, preactivate and activate should be started.) If an activation mode is specified for the validate, it cannot be specified for the preactivation or activation. If no activation mode is specified for the validate operation, it can not be specified for the preactivation or activation. See also subclauses 7.5.3 and 7.5.4.

Use of the validate operation shall have no influence on the fallback behaviour of a session.

Invoking the validate operation shall not result in any of the suboperations specified in the downloaded bulk configuration data being applied (clause 10). The operation is essentially passive.

7.5.2.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
sessionId	M	String identifying the session	Identifies this specific session and process associated with the requested bulk data download.
ActivationMode	O		Identifies whether a specific activation mode is required. See also subclauses 7.5.3 and 7.5.4. The valid choices are defined in

Parameter Name	Qualifier	Information type	Comment
			the parameter table in clause 7.5.4.

7.5.2.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of operation is successful or (b) operation failed because of specified or unspecified reasons

7.5.2.4 Pre-condition

downLoaded

Assertion Name	Definition
downLoaded	State as defined in clause 9. The Bulk CM IRP Agent has sucessfully opened the session and download had been attempted or repeat this operation.

7.5.2.5 Post-condition

validationInProgress

Assertion Name	Definition
validationInProgress	State = VALIDATE_IN_PROGRESS, as defined in clause 9. The Bulk CM IRP Agent has successfully started the validation of the downloaded configuration data.

7.5.2.6 Exceptions

7.5.2.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

7.5.3 Operation preactivate (O)

7.5.3.1 Definition

An IRPManager invokes this operation to request an IRPAgent to preactivate previously downloaded bulk configuration data (clause 10) that may have optionally been validated (subclause7.5.3). The principal, but not mandatory, functions of the preactivate operation is to validate the configuration data changes in the context of current operational data and to pre-process the configuration data changes. Use of this optional operation enables the IRPManager to prepare the activation of the downloaded bulk configuration data at the EM or NE level before requesting its effective activation. The actions shall fall short of executing the bulk configuration data changes (clause

10) in the network and impacting service. (The actions may for example be to validate the configuration data changes in the context of current operational data or to pre-process the configuration data changes). Performing such actions prior to activate may help identify any potential problems prior to executing the changes on a live a network and may minimise activation elapse time. See also subclause 7.5.6 for scope of checks during a session and specifically for preactivate.

Specifying an activation mode is optional. There can only be one activation mode for a session. If an activation mode is specified for the preactivation, it shall be when the first preactivate or validate operation is requested. If an activation mode was specified by validate it is not possible to change the activation mode initially specified with any subsequent preactivate or activate operations. If an activation mode was specified for the first preactivate operation, it is not possible to change the activation mode initially specified with any subsequent preactivate retries, activate or activate retries. (If another activation mode is required, a new session, download, validate, preactivate and activate should be started.) If no activation mode is specified for the first preactivate, it cannot be subsequently specified with any subsequent preactivate retries, activation or activation retries. (If specification of an activation mode is required, a new session, download, validate, preactivate and activate should be started.) See also subclauses 7.5.2 and 7.5.4.

See subclause 6.2.4.3 for description of optional verification mode parameter and associated checking.

Selecting a fallback option is optional. There can only be one fallback option for a session.

If the option is selected it shall be initiated when the first preactivation operation is requested. If a fallback option is not requested for the first preactivation, it cannot be subsequently requested for repeated preactivations or activations during the session. If the fallback option was requested, it is not possible to change the fallback option initially selected with any subsequent re- preactivate retries i.e. for a session it is only possible to fallback to the configuration that existed when the first preactivate operation was requested. See also clause 7.5.5. (If a new fallback configuration is required a new session, download, activate and preactivate should be started. The old session can be ended, prior to which fallback can optionally be invoked).

Specifying how preactivate operation retries within a session shall be implemented following a partially successful preactivation (e.g. repeat all preactivation management actions or just the uncompleted delta of management actions that did not previously complete successfully) is beyond the scope of this document. Only the IRPManager can initiate preactivate retries. (The IRPAgent shall not initiate retries autonomously).

7.5.3.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
`sessionId`	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data download that is required to be activated.
`VerificationMode`	O		Selects the mode of checking. One of two choices may be selected: “full checking”, “limited checking”, see subclause 7.5.6.3.
`activationMode`	O		Identifies whether a specific activation mode is required. See also subclauses 7.5.2 and 7.5.4. The valid choices are defined in the parameter table in clause 7.5.4.
`fallbackEnabled`	M		Indicates whether or not it is required to initialise and enable fallback option prior to the preactivation. This option is only open for the first preactivate operation of a session. For any subsequent preactivate operation retries within a session the fallbackEnabled parameter must be set to indicate it is not required to initialise fallback otherwise the pre-activate

Parameter Name	Qualifier	Information type	Comment
			operation retry shall fail.

7.5.3.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of operation is successful or (b) operation failed because of specified or unspecified reasons

7.5.3.4 Pre-condition

Assertion Name	Definition
downLoaded	State as defined in clause 9. The Bulk CM IRP Agent has sucessfully opened the session and download had been attempted or repeat this operation.

7.5.2.5 Post-condition

preactivationInProgress

Assertion Name	Definition
preactivationInProgress	State = PREACTIVATION_IN_PROGRESS, as defined in clause 9. The Bulk CM IRP Agent has successfully started the validation of the downloaded configuration data.

7.5.3.6 Exceptions

7.5.3.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

7.5.4 Operation ~~6.2.2.5~~ activate (M)

7.5.4.1 Definition

An IRPManager invokes this operation to request an IRPAgent to activate previously downloaded bulk configuration data (clause ~~8~~10) that may have optionally been checked (subclause 7.5.2) and/or been preactivated (subclause 7.5.3).

Activate means that operations specified in a previously downloaded configuration data file, for example create, delete and modify of managed objects are carried out on the live network i.e. mobile subscribers are affected by the downloaded configuration data.

An IRPAgent may support an optional activationMode parameter. This enables the IRPManager to indicate to the IRPAgent the preference for how the activation shall be executed. One of two options may be selected: “least service impact” or “least elapse time”. If the “least service impact” option is selected the IRPAgent shall optimise the execution of the activation in a way that minimises disruption to network services. Elapse time to complete the activation is of secondary importance. If the “least elapse time” option is selected the IRPAgent shall optimise the execution of the activation in a way that minimises the elapse time for completing the execution of the activation. During the execution, disruption of network services is of secondary importance.

See subclause 7.5.6 for descriptions of checks made during activate execution.

Specifying an activation mode is optional. There can only be one activation mode for a session. If an activation mode is specified for the activation, it shall be when the first activate, validate or preactivate operation is requested. If an activation mode was specified by validate or preactivate operations, it is not possible to change the activation mode initially specified with any subsequent activate operations. If an activation mode was specified for the first activate, it is not possible to change the activation mode initially specified with any subsequent activate retries. (If another activation mode is required, a new session, download, validate, preactivate and activate should be started.) If no activation mode is specified for the first activate, it cannot be subsequently specified with any subsequent activate retries. (If specification of an activation mode is required, a new session, download, validate, preactivate and activate should be started.) See also subclauses 7.5.2 and 7.5.3.

If a preactivation had been invoked, successful completion of activate should release any internal local resources allocated for the preactivation.

Selecting a fallback option is optional. There can only be one fallback option for a session.

If the fallback option is selected it shall be initiated when the first activation or preactivation operation is requested. If a fallback option is not requested for the first activation or preactivation, it cannot be subsequently requested for repeated activations or an activation following a preactivation during the session. If the fallback option was requested, it is not possible change the fallback option initially selected with any subsequent re-activate retries or an activation following a peractivation i.e. for a session it is only possible to fallback to the configuration that existed when the first activate or preactivate operation was requested. See also subclause- 7.5.5. (If a new fallback configuration is required a new session, download and activate should be started. The old session can be ended, prior to which fallback can optionally be invoked).~~Enabling the fallback option is optional. There can only be one choice for a session: enabled or not enabled. If enabling the fallback option is selected it shall be initiated when the first activation operation is requested. If enabling the fallback option is not requested for the first activation, it cannot be subsequently requested for repeated activations during the session. If enabling the fallback option was requested, it is not possible change this choice with any subsequent re-activate retries i.e. for a session it is only possible to fallback to the configuration that existed when the first activate operation was requested. See also subclause 6.2.2.6. (If a new fallback configuration is required a new session, download and activate should be started. The old session can be ended, prior to which fallback can optionally be invoked).~~

Specifying how activate operation retries within a session shall be implemented following a partially successful activation (e.g. repeat all activation management actions or just the uncompleted delta of management actions that did not previously complete successfully) is beyond the scope of this document. Only the IRPManager can initiate activate retries. (The IRPAgent shall not initiate retries autonomously).

7.5.4.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
`sessionId`	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data download that is required to be activated.
`activationMode`	O		Identifies whether a specific activation mode is required. See also subclauses 7.5.2 and- 7.5.3. It may be set to indicate “least

Parameter Name	Qualifier	Information type	Comment
			service impact" or "least elapse time" types of activation are required.
~~saveFallback~~fallbackEnabled	M		Indicates whether or not it is required to initialise and enable fallback option prior to the activation. This option is only open for the first activate operation of a session. For any subsequent activate operation retries within a session the ~~saveFallback~~fallbackEnabled parameter must be set to indicate it is not required to initialise fallback otherwise the ~~re-~~activate operation retry shall ~~be rejected~~fail.

7.5.4.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of operation is successful ~~and~~or (b) operation failed because of specified or unspecified reasons

7.5.4.4 Pre-condition

downLoaded

Assertion Name	Definition
downLoaded	State as defined in clause 9. The Bulk CM IRP Agent has sucessfully opened the session and download had been attempted or repeat this operation.

7.5.4.5 Post-condition

activationInProgress

Assertion Name	Definition
activationInProgress	State = ACTIVATE_IN_PROGRESS, as defined in clause 9. The Bulk CM IRP Agent has successfully started the activation of the downloaded configuration data.

7.5.4.6 Exceptions

7.5.4.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~**Table 5: activate parameters**~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with an earlier bulk data download that is required to be activated.~~
~~fallbackEnabled~~	~~Input, M~~	~~Indicates whether or not the fallback option is required to be enabled. If required to be enabled, the first activate operation of a session that gets properly initiated in the IRPAgent shall initialise and enable fallback option prior to the activation. Enabling or not the fallback option is only open for the first activate operation of a session. For any subsequent activate operation retries within a session the fallbackEnabled parameter shall have the same value as for the first activate operation otherwise the re-activate operation shall be rejected.~~
~~status~~	~~Output, M~~	~~indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.5.5 Operation ~~6.2.2.6~~ fallback (M)

7.5.5.1 Definition

An IRPManager invokes this operation to request an IRPAgent to ~~activate a fallback area if~~recover after a previously ordered activation or preactivation has failed.

If a fallback is requested after a preactivation but before an activation the IRPAgent should as necessary return any internal local resources impacted by the preactivation back to the same state they were in prior to the preactivation being invoked. There is no impact to the operational network resources as the activate operation has not been invoked.

If fallback is requested after an activation the IRPAgent shall instigate activating the fallback area to restore the operational network resources impacted by the configuration changes for the session back to the configuration they were in when the fallback option was selected during the session. If a preactivation was also performed, as necessary the IRPAgent should return any internal local resources impacted by the preactivation back to the same state they were in prior to the preactivation being invoked.

Specifying how fallback operation retries within a session shall be implemented after a fallback fails (e.g. repeat all fallback functions or just the delta of fallback functions that did not previously complete successfully) is beyond the scope of this document. Only the IRPManager can initiate the fallback operation. The IRPAgent shall not initiate fallback or fallback retries autonomously. Within a session the fallback operation shall only be accepted if an initial activate or preactivate operations was performed with fallback option enabled. For further discussion of enabling or not the fallback option see subclause ~~6.2.2.5~~ 7.5.4.

7.5.5.2 Input parameters

Parameter Name	Qualifier	Information type	Comment
SessionId	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current log

Parameter Name	Qualifier	Information type	Comment
			is required.

7.5.5.3 Output parameters

Parameter Name	Qualifier	Matching Information	Comment
Status	M	ENUM(OperationSucceded, OperationFailed).	indicates (a) start of operation is successful ~~and~~or (b) operation failed because of specified or unspecified reasons

7.5.5.4 Pre-condition

fallbackEnabled

Assertion Name	Definition
fallbackEnabled	State as defined in clause 9. The Bulk CM IRP Agent has sucessfully opened the session and fallbackEnables=True by either, preactivate or activate operations being successfully invoked, as defined in subclauses 7.5.3 and 7.5.4.

7.5.5.5 Post-condition

fallbackInProgress

Assertion Name	Definition
fallbackInProgress	State = FALLBACK_IN_PROGRESS, as defined in clause 9. The Bulk CM IRP Agent has successfully started the fallback.

7.5.5.6 Exceptions

7.5.5.6.1 operationFailed

Exception Name	Definition
operationFailed	**Condition:** Pre-condition is false or post-condition is false. **Returned information:** The output prameter status. **Exit state:** Entry state.

~~Table 6 : fallback parameters~~

~~Name~~	~~Qualifier~~	~~Description~~
~~sessionId~~	~~Input, M~~	~~Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current log is required.~~
~~status~~	~~Output, M~~	~~indicates (a) start of operation is successful and (b) operation failed because of specified or unspecified reasons~~

7.5.4 Operation 6.2.2.7 Table 7: abortSessionOperation parameters

Name	Qualifier	Description
sessionId	Input, M	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the abort is required.
status	Output, M	indicates (a) start of abort operation is successful and (b) abort operation failed because of specified or unspecified reasons

6.2.2.11 getBulkCmIRPVersion (M)

IRPManager invokes this operation when it wishes to find out the Bulk CM IRP SS versions supported by IRPAgent. IRPAgent shall respond with a list of supported Bulk CM IRP SS versions.

Table 11: Parameters of getBulkCmIRPVersion

Name	Qualifier	Description
VersionNumberList	Output, M	It indicates one or more SS version numbers supported by the IRPAgent.
status	Output, M	Operation succeeded in that versionNumberList contains valid result. (b) Operation failed. Output parameter versionNumberList may contain invalid result.

7.5.6 Validation and Checking Functions

7.5.6.1 Download Checks

During download the IRPAgent should check the consistency of imported configuration data against the data schema to ensure there are no errors. The IRPAgent is not required to check the semantic of the downloaded bulk configuration data during the download.

7.5.6.2 Validate Checks

During validation the IRPAgent should check the syntax and semantic of previously downloaded bulk configuration data.

7.5.6.3 Preactivation Checks

During preactivation the IRPAgent should check the semantic of previously downloaded bulk configuration data, and must also check the syntax if a validate operation has not previously been successfully performed.

An Element Manager should, if technically feasible, send the configuration data changes to all Network Elements (NE) for the NE to verify, to the extent possible, that the activate will successfully execute the configuration data changes. If any elements of configuration change data that will not successfully execute are identified, diagnostic data identifying the NEs and failing configuration data elements will be made available to the Manager.

An IRPAgent may support an optional verification mode parameter, see clause 7.5.2. When the IRPManager does not require extensive checking, this parameter may be used to constrain the scope of validation to avoid performing checks that potentially may require extensive real time to execute, for example checks actively involving entities outside the IRPAgent such as NE's. The validation mode parameter has two values: "full checking" and "limited checking". In the “full checking” mode, the checking should be as complete as possible with the intent of achieving the greatest assurance that the subsequent activation operation will be successful. In the "limited checking" mode, checking that can be performed by the IRPAgent rapidly is still performed, but further checking that may cause significant delays to execute should be omitted.

7.5.6.4 Activate Checks

During the activation the same checks as for validate and preactivate should be performed if these operations have not previously been successfully performed. These checks may also be repeated if the context may have changed.

8.1 Interface BulkCmIRPNotifications#1 ~~6.2.3 Configuration File Notifications~~

~~The following configuration file Notifications exist in the Itf-N.~~

- ~~notifySessionStateChanged~~
- ~~notifyGetSessionLogEnded~~

~~(Subscribe and Unsubscribe are also associated with the Bulk CM IRP, but these operations are part of the 32.302: "TM; Notification Management; Part 2: Notification IRP; IS " [3]).~~

~~6.2.3.1 General~~

~~Operations that IRPManager uses to manage subscription to receive notifications are specified in 3GPP TS 32.302 [3]. 3GPP TS 32.302 [3] also specifies a generic parameter information that is commonly found in notifications defined by IRPs. The commonly carried parameter attributes are collectively called `notification Header` in the present document and . The parameter attribute names and their qualifiers are listed in Table 12 .~~

~~Table 12: Notification Header~~

~~Parameter Attributes defined in 3GPP TS 32.302 [3]~~	~~Qualifier for use in this IS~~	~~Comment~~
~~managedObjectClass/ (objectClass([3])~~	~~O~~	~~See [3]~~
~~ManagedObjectInstance/(objectInstance [3])~~	~~O~~	~~See [3]~~
~~`NotificationId`~~	~~O~~	~~See [3]~~
~~`EventTime`~~	~~M~~	~~See [3]~~
~~`systemDN`~~	~~O~~	~~See [3]~~
~~NotificationType~~	~~M~~	~~Indicates the type of notification. The type used for each Bulk CM Notification are specified in Tables 13 and 14~~

~~The following clauses define specific notifications relevant for Bulk CM IRP by extending notify in 32.302 Notification IRP IS [3].~~

~~6.2.3.2~~ 8.1.1 Notification notifySessionStateChanged (M)

8.1.1.1 Definition

The IRPAgent notifies the IRPManager that a state change has occurred on a bulk ~~data~~ configuration data file sessionId operation subscribed to by the IRPManager. E.g. a configuration data file is available for processing after an upload, a download is complete. See clause ~~7~~ 9 for a further description of states.

Parameter Name	Qualifiers	Matching Information	Comment
objectClass	O,F	ManagedEntity.objectClass	Notification header - see [3].
objectInstance	O,F	ManagedEntity.objectInstance.	Notification header - see [3].
notificationId	O	This carries the semantics of notification identifier.	Notification header - see [3].
eventTime	M,F	ManagedEntity.creationTime	Notification header - see [3].
systemDN	O,C,F	IRPAgent.systemDN where the IRPAgent is related to the KernelCmIRP.	Notification header - see [3].
NotificationType	M,F	Mapped to notificationType in [3].	Notification header - see [3]. For this notification it indicates notification type is Notify Session State Changed.
sessionId	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current status is required.
sourceIndicator	O		This parameter, when present, indicates the source of the operation that led to the generation of this notification. It can have one of the following values: • resource operation: The notification was generated in response to an internal operation of the resource; • management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; • unknown: It is not possible to determine the source of the operation.
sessionState	M	ENUM(Upload Failed, Upload Completed, Download Failed, Download Completed, Validation Failed, Validation Completed, Preactivation Failed, Preactivation Partly Realised, Preactivation Completed, Activation Failed, Activation Partly Realised, Activation Completed, Fallback Failed, Fallback Partly Realised, Fallback Completed)	Indicates the state transition that caused the Notification. See clause 9. i.e. Upload Failed, Upload Completed, Download Failed, Download Completed, Validation Failed, Validation Completed, Preactivation Failed, Preactivation Partly Realised, Preactivation Completed, Activation Failed, Activation Partly Realised, Activation Completed, Fallback Failed, Fallback Partly Realised, Fallback Completed. (Note: as per sub-clause 9.2 “in-progress” transition states are not notified)

8.1.1.3 Triggering events

State transitions as defined in clause 9.

Table 13: notifySessionStateChange parameters

Name	Qualifier	Description
notificationHeader	Input, M	See Table 12 Notification Header.
NotificationType of notificationhHeader	Input, M	See Table 12 Notification Header. For this notification it indicates notification type is Notify Session State Changed.
sessionId	Input, M	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which the current status is required.
sourceIndicator	Input, O	This parameter, when present, indicates the source of the operation that led to the generation of this notification. It can have one of the following values: □resource operation: The notification was generated in response to an internal operation of the resource; □management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; □unknown: It is not possible to determine the source of the operation.
sessionState	Input, M	Indicates the state transition that caused the Notification. See clause 7. i.e. Upload Failed, Upload Completed, Download Failed, Download Completed, Activation Failed, Activation Partly Realised, Activation Completed, Fallback Failed, Fallback Partly Realised, Fallback Completed. (Note: as per sub-clause 7.2 "in-progress " transition states are not notified)

8.1.1 Notification 6.2.3.3 NotifyGetSessionLogEnded (M)

8.1.1.1 Definition

The IRPAgent notifies the IRPManager that a requested GetSessionLog for a bulk data configuration file sessionId operation subscribed to by the IRPManager has ended successfully or unsuccessfully.

8.1.1.2 Input parameters

Parameter Name	Qualifiers	Matching Information	Comment
objectClass	O,F	ManagedEntity.objectClass	Notification header - see [3].
objectInstance	O,F	ManagedEntity.objectInstance.	Notification header - see [3].
notificationId	O	This carries the semantics of notification identifier.	Notification header - see [3].
eventTime	M,F	ManagedEntity.creationTime	Notification header - see [3].
systemDN	O,C,F	IRPAgent.systemDN where the IRPAgent is related to the KernelCmIRP.	Notification header - see [3].
NotificationType of notificationHeader	M,F	Mapped to notificationType in [3	Notification header - see [3].For this notification it indicates notification type is Notify Bulk CM Log State.
SessionId	M	String identifying the session	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which Log State is required.
SourceIndicator	O		This parameter, when present, indicates the source of the operation that led to the generation of this notification. It can have one of the following values: • resource operation: The notification was generated in response to an internal operation of the resource; • management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; • unknown: It is not possible to determine the source of the operation.
SessionLogStatus	M	Boolean = GetSessionLog completed successfully or GetSessionLog completed unsucessfully	Indicates event that caused the Notification i.e. GetSessionLog completed successfully, GetSessionLog completed unsucessfully.

8.1.1.3 Triggering event

Attempt to transfer session log to destinations completed successfully or failed. Session state independent, see clause 9.

Table 14 : notifyGetSessionLogEnded parameters

Name	Qualifier	Description
NotificationHeader	Input, M	See Table 12: Notification Header.
NotificationType of notificationHeader	Input, M	See Table 12 Notification Header. For this notification it indicates notification type is Notify Bulk CM Log State.
SessionId	Input, M	Identifies this specific session and process associated with an earlier bulk data operation e.g. upload or download for which Log State is required.
SourceIndicator	Input, O	This parameter, when present, indicates the source of the operation that led to the generation of this notification. It can have one of the following values: □ resource operation: The notification was generated in response to an internal operation of the resource; □ management operation: The notification was generated in response to a management operation applied across the managed object boundary external to the managed object; □ unknown: It is not possible to determine the source of the operation.
SessionLogStatus	Input, M	Indicates event that caused the Notification i.e. GetSessionLog completed successfully, GetSessionLog completed unsucessfully.

79 State Machine

79.1 State Machine Overview

The Bulk CM IRPAgent state machine satisfies the following general requirements and characteristics for Bulk CM IRP:

1) Each configuration session is associated with one state machine. The session is identified by the sessionId. If a session is a started (startSession operation) an instance of the state machine is created. If the session is ended (endSession operation) the instance of the state machine is deleted.

2) Under normal operation without errors the IRPManager is able to supervise a configuration session by just monitoring the state change notifications (notifySessionStateChanged) triggered by the IRPAgent

3) Under abnormal conditions where the IRPManager is not notified of a change, the getSessionStatus operation can be invoked to determine current state of the session. The IRPManager does not need to maintain a history of the state machine.

4) On the IRPAgent there is only one download configuration data file (clause 810) associated with a session at a time.

5) Multi configuration session must be supported by the IRPAgent. E.g. it must be possible to invoke an upload session in parallel with an active activate session.

6) The IRPAgent resolves concurrency problems on a "first come - first serve" basis. E.g. an upload and an activation requested on the same configuration data can not be performed at the same time and in this case the first will be progress to completions and the second request rejected.

7) It must be possible to abort a configuration session within a transition state.

8) The operator/IRPManager decides on whether or not enabling the fallback option is required before requesting an activation or preactivation Enabling the fallback option will maintain the disposition of the configuration before the activation or preactivation . The fallback configuration information is established at point before the first activation or preactivation is started. If there are multiple activation or preactivation attempts during a session only one (first) fallback configuration is maintained.

9) The session log file can be requested in any state. The uploaded log file contains information which is specific to the configuration session.

10) Clause 7.3 defines the valid state machine pre and post conditions for each operation.

~~7~~9.2 State Machine Description

The IRPAgent progresses Bulk CM operations and associated configuration data changes (clause ~~8~~10) within a session according to the state machine defined here. The IRPManager can manage a configuration session using session state change notifications which are triggered by the IRPAgent. Not all state changes defined here are notified to the IRPManager. The transition states (UPLOAD_IN_PROGRESS, DOWNLOAD_IN_PROGRESS, VALIDATION_IN_PROGRESS, PREACTIVATION_IN_PROGRESS, ACTIVATION_IN_PROGRESS) are not notified to the IRPManager as they are not required.

If the IRPManager becomes unaware or needs to confirm the current state of a configuration session it can request this by invoking getSessionStatus operation. It is not required to know the history of the state machine. The getSessionStatus operation will provide the "actual " current status.
An IRPManager may request the status when it detects loss of control, for example because of the following reasons:

1) Session state change notifications are not being received as expected, e.g. because IRPAgent is blocked in a transition state, e.g. ACTIVATION_IN_PROGRESS;

2) IRPManager gets disconnected from the IRPAgent, e.g. session state notification are not received.

The session state notification events are a considered a subset of the state machine (without transition state). The actual configuration state can be requested via getSessionStatus. Because of this common behaviour it is reasonable to define one interface type for the state machine handling which is used in the session state notification and in the getSessionStatus operation.
The IRPManager will only receive notifications if it registered itself at the IRPAgent with the subscribe operation.

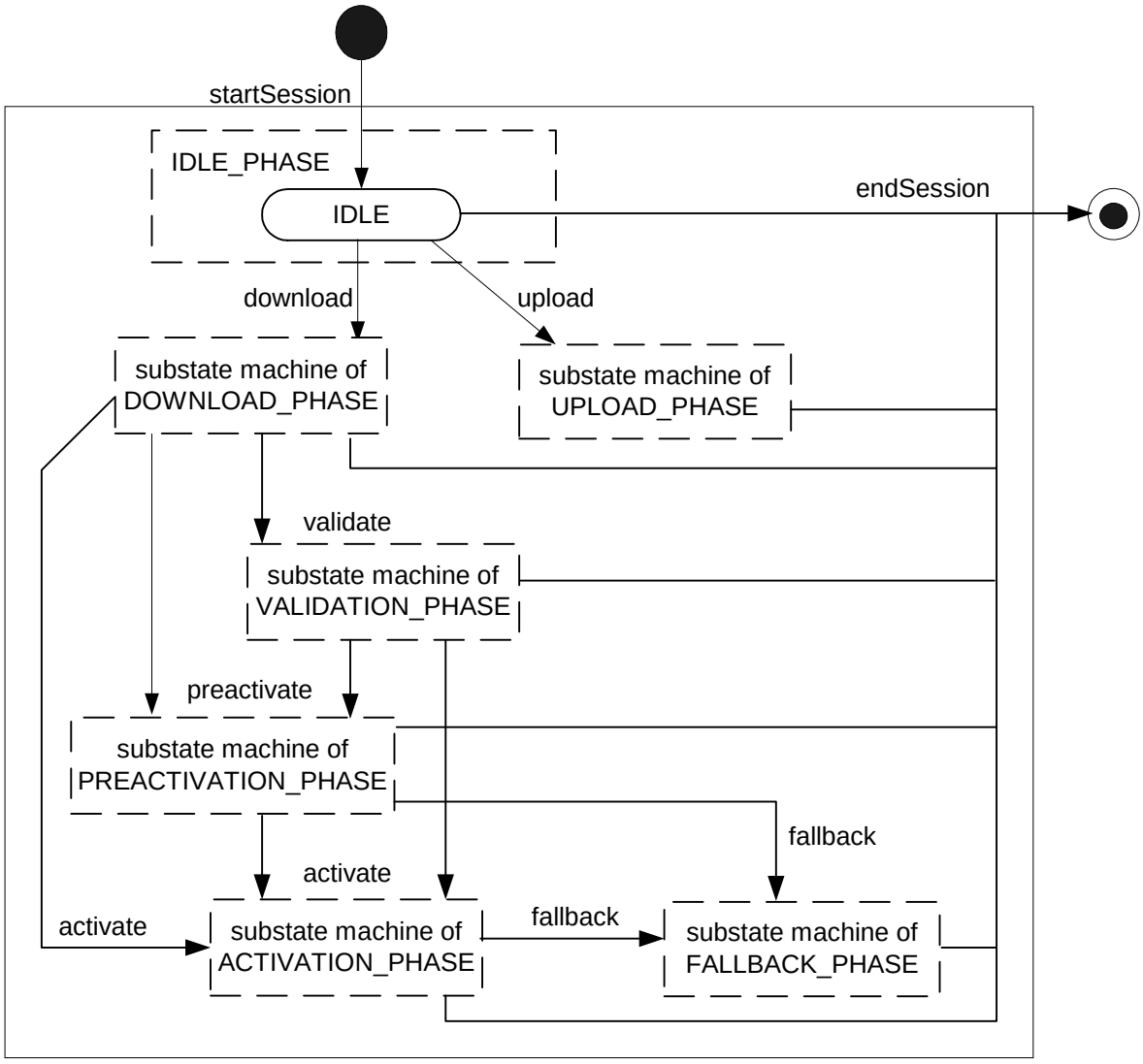
For ease of description the state machine of a configuration session is introduced with the notion of substate machines but state itself are named unique. This kind of notion is not to be interpreted as providing implementation directions.

Within the description of the substate machines it is becoming clear that they have the following state symmetries~~.~~:

~~the~~ The state of the UPLOAD_PHASE, ~~and~~ the DOWNLOAD_PHASE and the VALIDATION_PHASE ~~are the same~~are similar.

~~the~~ The state of the ACTIVATION_PHASE, PREACTIVATION_PHASE and the FALLBACK_PHASE ~~are the same~~ are similar.

The startSession operation creates a state machine. The initial state of the configuration session in the IDLE_PHASE is IDLE. The endSession deletes a state machine which is not in a transition state, more details are defined in the substate machines.


startSession
IDLE_PHASE
IDLE
endSession
download
upload
substate machine of DOWNLOAD_PHASE
substate machine of UPLOAD_PHASE
validate
substate machine of VALIDATION_PHASE
preactivate
substate machine of PREACTIVATION_PHASE
fallback
activate
activate
substate machine of ACTIVATION_PHASE
fallback
substate machine of FALLBACK_PHASE

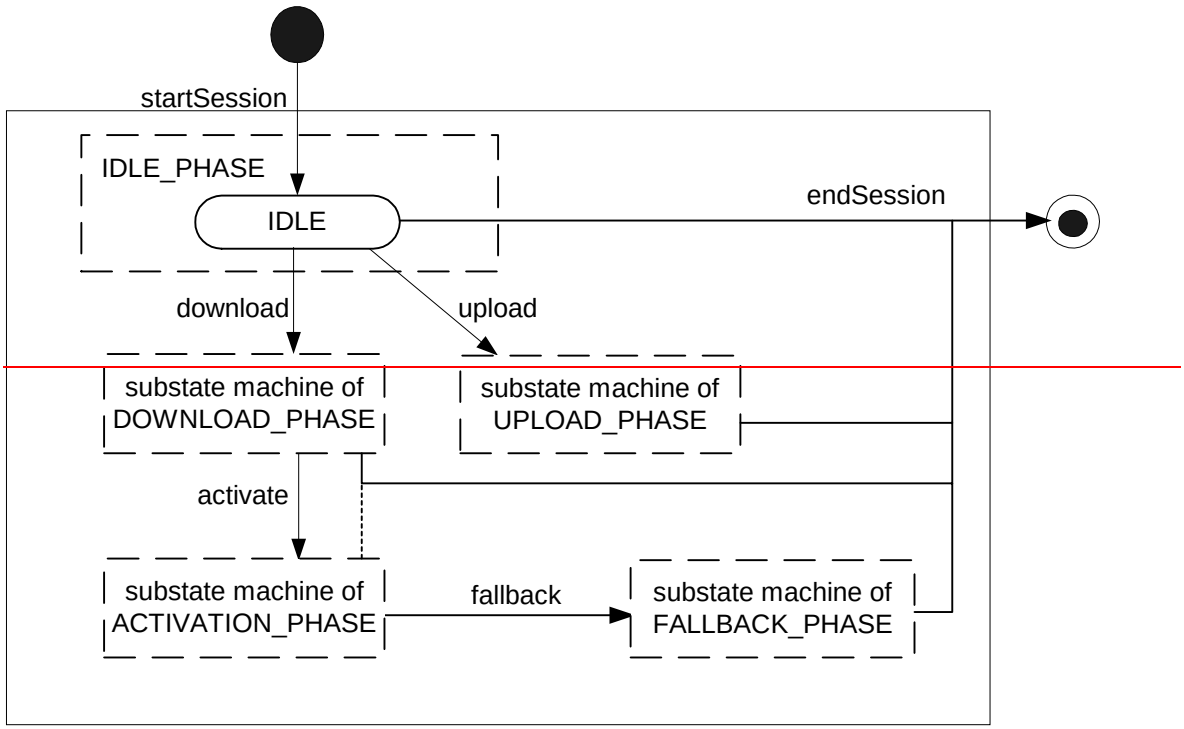


Figure 5: State Machine

The following figures describe~~s~~ the ~~-~~substate machine of a configuration session. The transition states, DOWNLOAD_IN_PROGRESS, UPLOAD_IN_PROGRESS VALIDATION_IN_PROGRESS, PREACTIVATION_IN_PROGRESS and ACTIVATION_IN_PROGRESS, are either left implicit if the IRPAgent finished the processing or explicit via an abortSessionOperation operation from the IRPManager.

In these figures solid transition lines indicate the transition is caused by an external event and dashed transition lines indicate the transition is caused by an internal event or decision as depicted in figure 6.

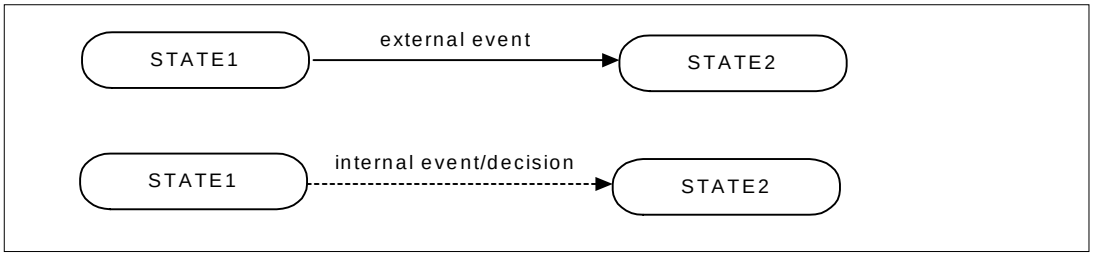


Figure 6: Depicting State Transition Lines for Internal and External Events and Decision

~~7~~9.2.1 Upload Phase

When the upload is triggered the IRP Agent writes the requested configuration data into a configuration data file and copies to the file reference provided by the IRP Manager. If the process succeeds the state UPLOAD_COMPLETED is indicated. If the upload fails a retry can be triggered in state UPLOAD_FAILED.

Once a session is associated with an upload none of the other state changes phases outside of the upload phase, i.e., download, validate, preactivate and activate phases cannot be triggered for the session.

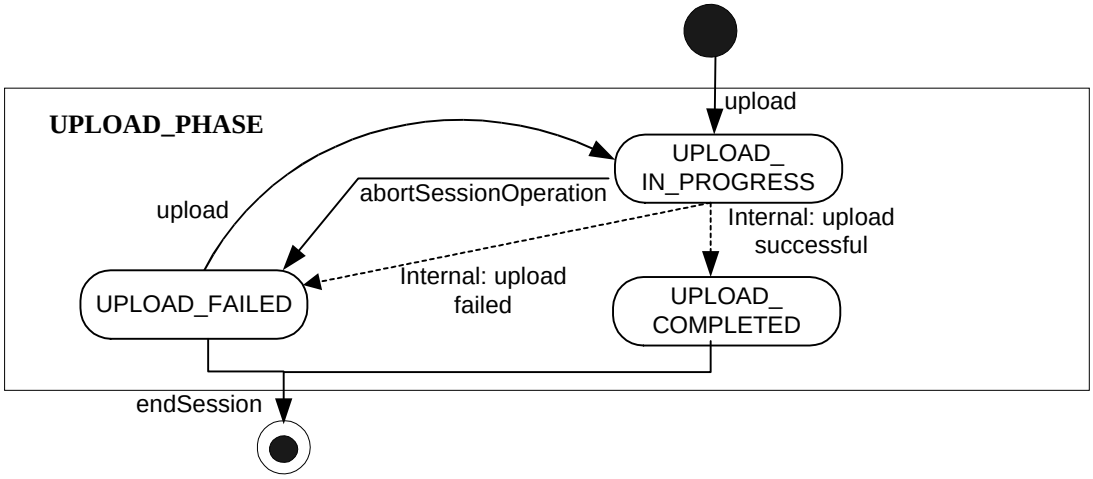
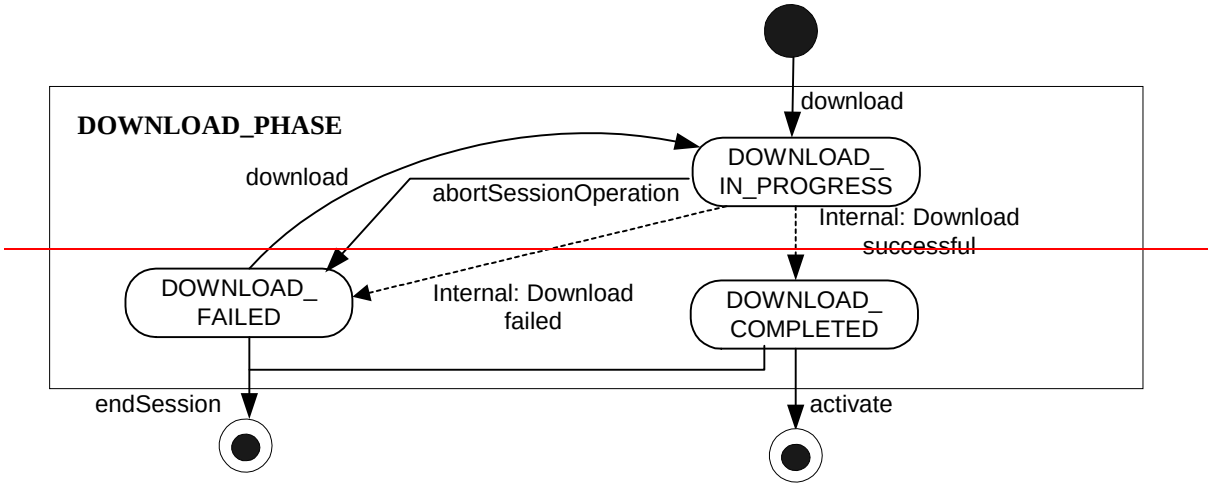


Figure 7: Substate Machine – UPLOAD_PHASE

~~7~~9.2.2 Download Phase

When the download is triggered the IRP Agent copies the configuration data file (clause ~~08~~10) from a given file area. The file is parsed and validated. If valid the state DOWNLOAD_COMPLETED is indicated. If the download fails a retry can be triggered in state DOWNLOAD_FAILED.

Once a ~~configuration is specialised to~~ session is associated with a download/validate/preactivate/activation behaviour then an upload phase ~~can not~~ cannot be triggered within this session.



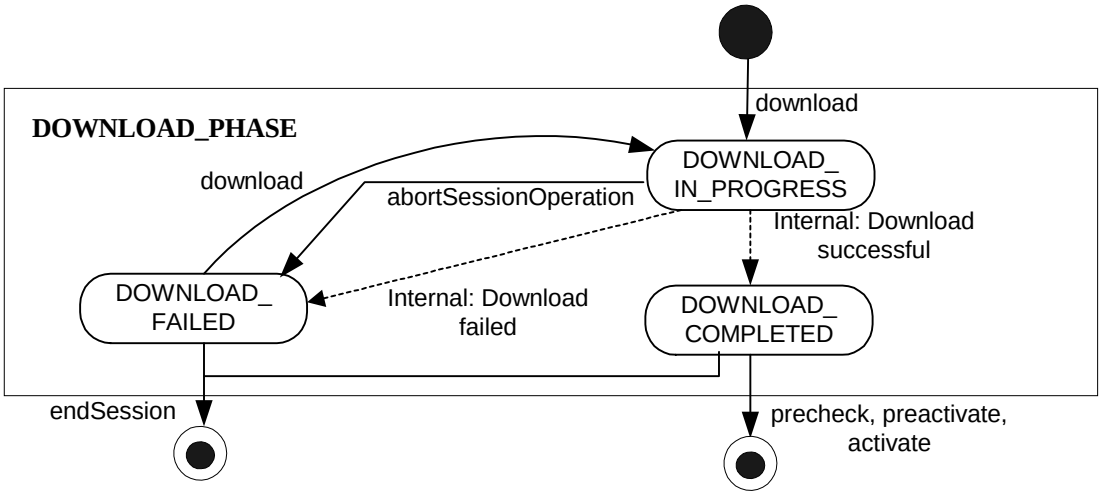


Figure 8: Substate Machine – DOWNLOAD_PHASE

9.2.4 Validation Phase

After a download had been completed the configuration data can be semantically validated before being preactivated or activated into the real subnetwork of an IRPAgent. (see subclause 7.5.6.2). A best effort strategy shall be applied. If validation was successfull the state VALIDATION_COMPLETED is indicated. If the validate fails a retry can be triggered in state VALIDATION_FAILED.

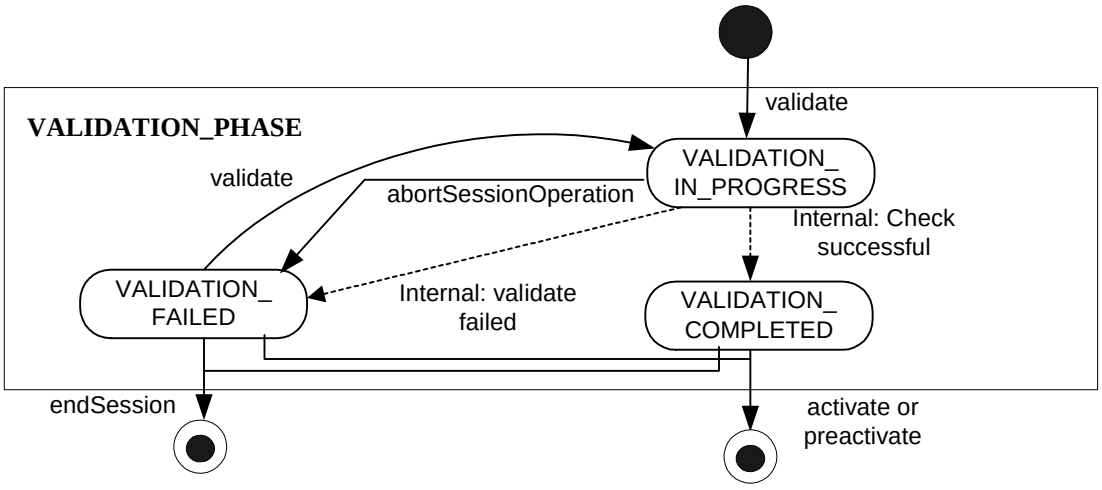


Figure 11: Substate Machine – VALIDATION_PHASE

9.2.5 Preactivation Phase

After a download had been completed and optionally validated the configuration data can be preactivated before being activated into the real subnetwork of an IRPAgent. If the process fully succeeds the preactivation is completed.

For preactivation a best effort strategy shall be employed.

If the IRPAgent is unable to successfully complete all pre-MIB changes that were actioned in the configuration data file (clause 10) the state PREACTIVATION_PARTLY_REALISED is indicated. This state is not an error condition because the preactivation of configuration data changes follows a best effort strategy. If the preactivation fails completely i.e. there are no pre-MIB changes the state PREACTIVATION_FAILED is indicated. A retry of the preactivate can be performed in states PREACTIVATION_PARTLY_REALISED and PREACTIVATION_FAILED. The PREACTIVATION_FAILED state cannot be entered if previously during the session the state had become PREACTIVATION_PARTLY_REALISED. The PREACTIVATION_PARTLY_REALISED state should be re-entered instead. A retry of the preactivate is allowed so that it is possible to recover after transient condition that caused an preactivate to fail or partly realise are no longer present.

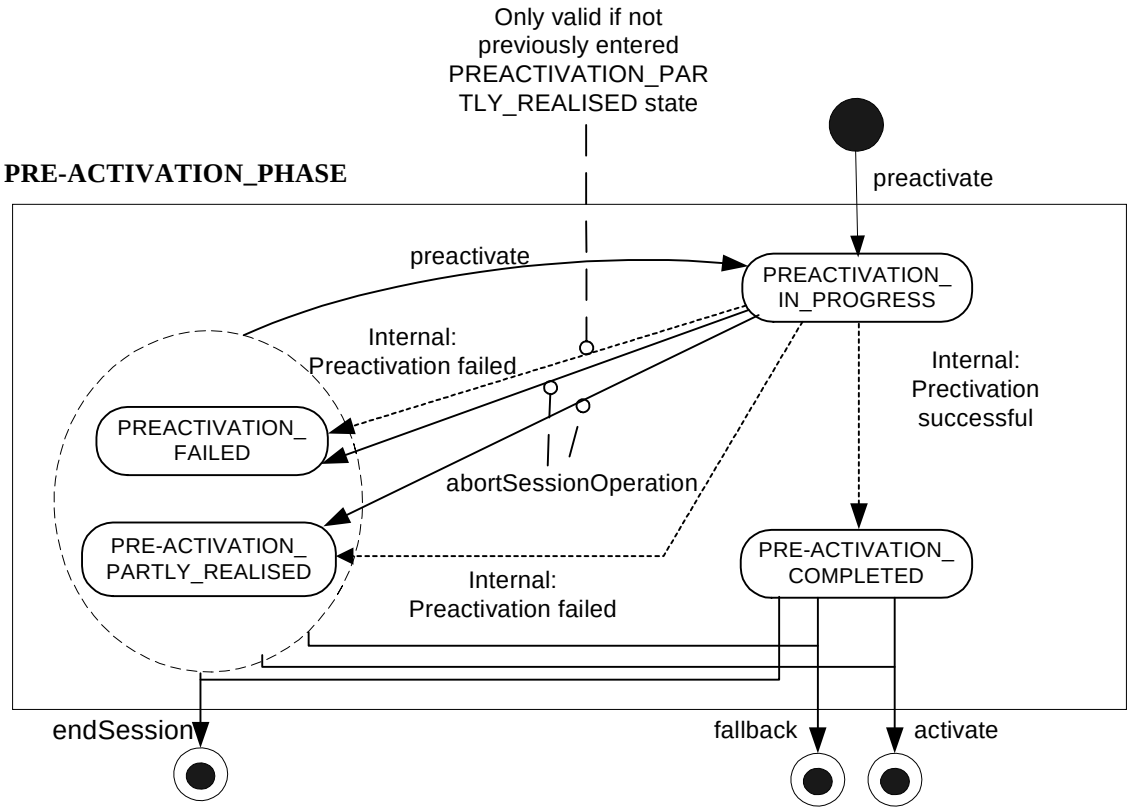


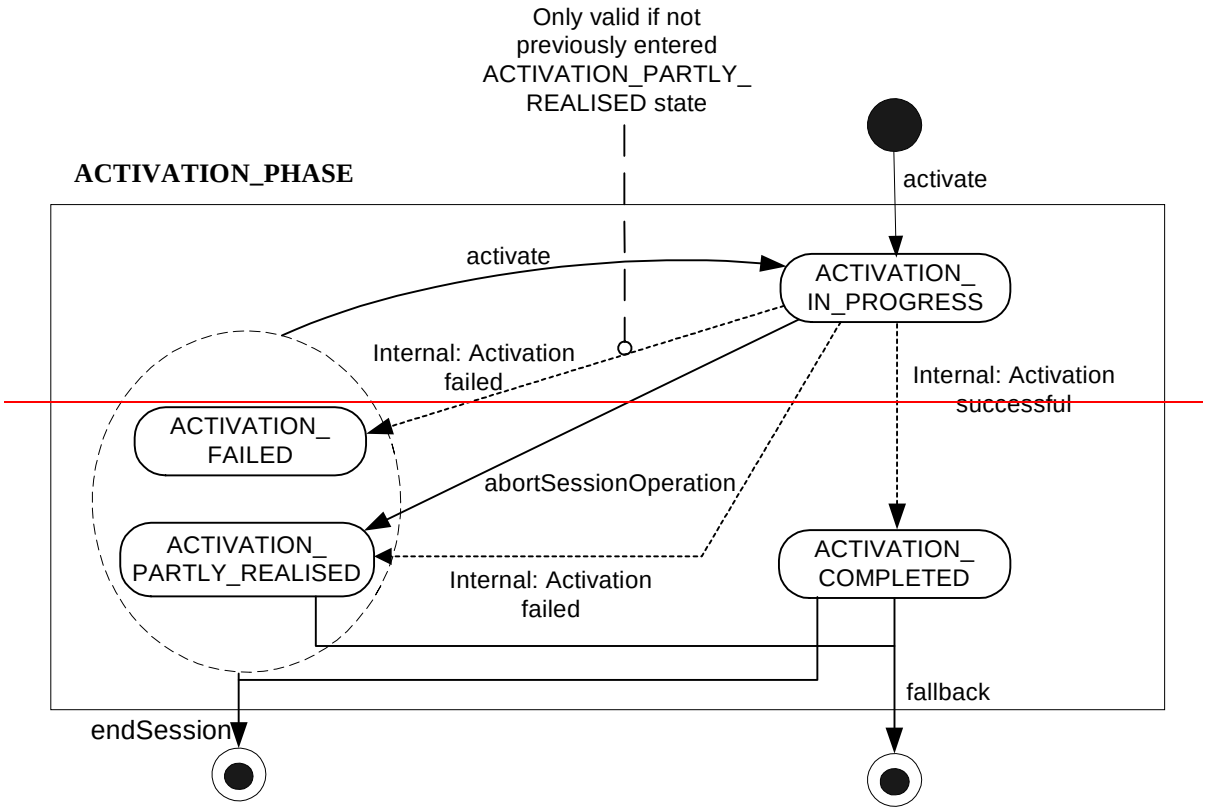
Figure 12: Substate Machine – PREACTIVATION_PHASE

79.2.36 Activation Phase

After a download has been completed and optionally validated and/or preactivated the configuration data can be activated into the real subnetwork of an IRPAgent. If the process fully succeeds the activation is completed.After a download had been completed the configuration can be activated into the real subnetwork of an IRPAgent. If the process fully succeeds the activation is completed.

For activation a best effort strategy shall be employed.

If the IRPAgent is unable to successfully complete all MIB changes and corresponding changes in the network elements that were actioned in the configuration data file (clause ~~8~~10) the state ACTIVATION_PARTLY_REALISED is indicated. This state is not an error condition because the activation of configuration data changes follows a best effort strategy. If the activate fails completely i.e. there are no MIB changes or corresponding changes in the network elements, the state ACTIVATION_FAILED is indicated. A retry of the activate can be performed in states ACTIVATION_PARTLY_REALISED and ACTIVATION_FAILED. The ACTIVATION_FAILED state cannot be entered if previously during the session the state had become ACTIVATION_PARTLY_REALISED. The ACTIVATION_PARTLY_REALISED state should be re-entered instead. A retry of the activate is allowed so that it is possible to recover after transient condition that caused an activate to fail or partly realise are no longer present.



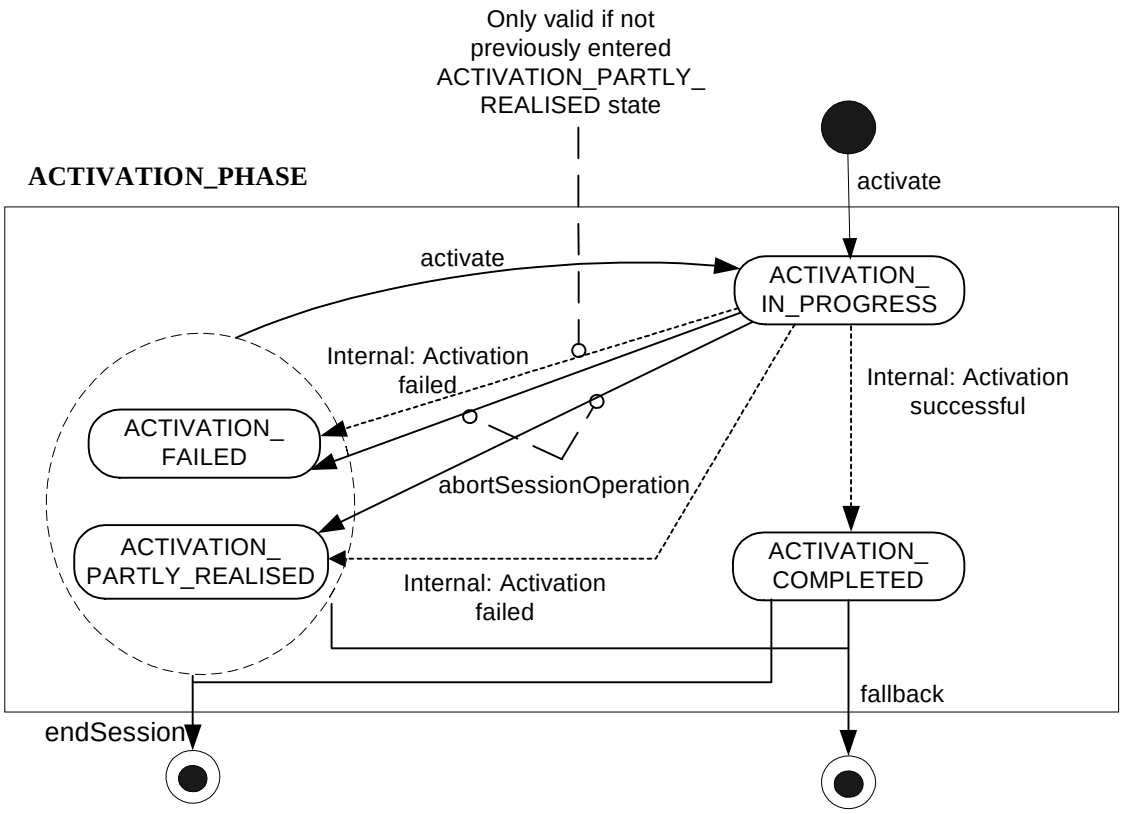


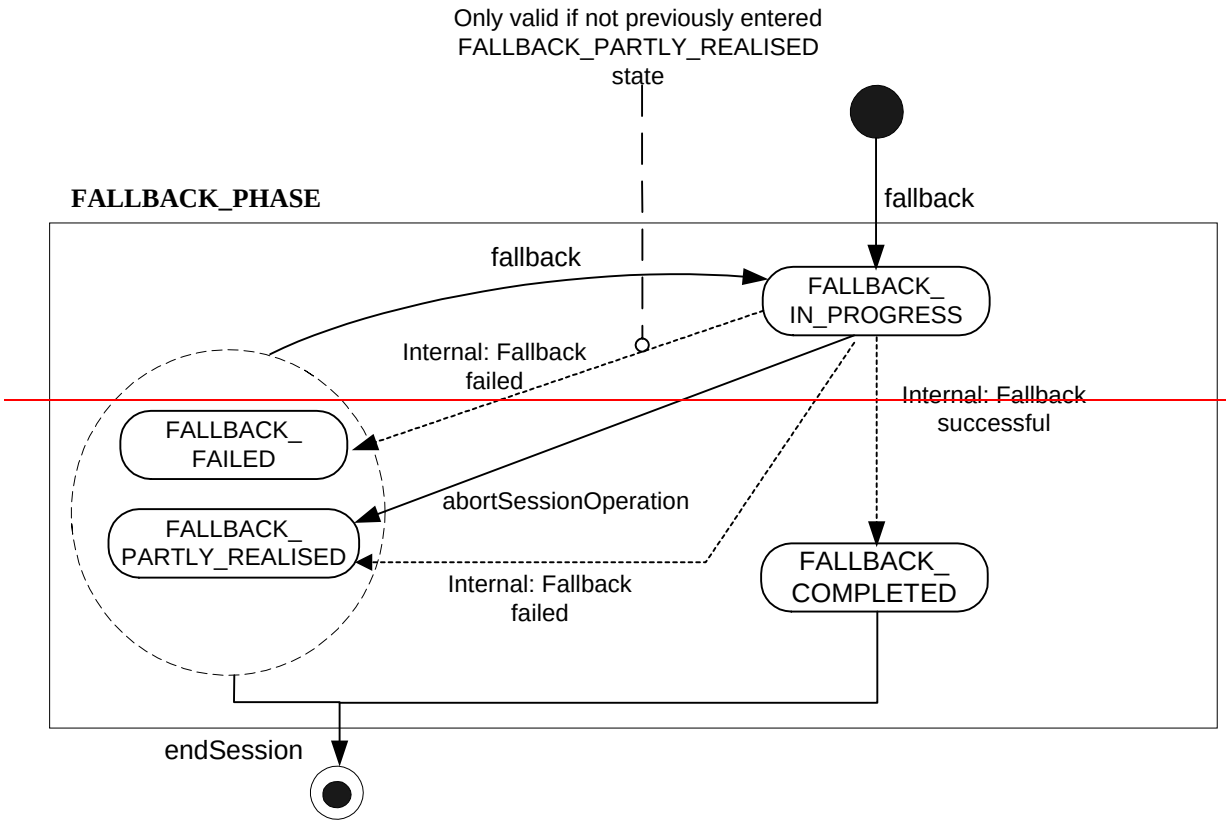
Figure 9: Substate Machine – ACTIVATION_PHASE

~~7~~9.2.~~4~~7 Fallback Phase

If an activate or preactivate operation was requested with the fallback option enabled and was successfully or partially completed then a fallback operation can be requested. If the process of a fallback fully succeeds then the related MIB and subnetwork is reverted back to its former configuration prior to first configuration data file preactivation or activation of a session.

For fallback a best effort strategy shall be employed.

In case that not all MIB changes and corresponding changes in the network elements that were actioned in configuration data file (clause 8) were successfully reverted back the state FALLBACK_PARTLY_REALISED is indicated. This state is not an error condition as the fallback to the former configuration follows a best effort strategy. If the fallback fails completely i.e. no MIB changes or corresponding changes in the network elements can be reverted back then the state FALLBACK_FAILED is indicated. A retry of fallback can be performed in the states FALLBACK_PARTLY_REALISED and FALLBACK_FAILED. The FALLBACK_FAILED state cannot be entered if previously during the session the state had become FALLBACK_PARTLY_REALISED. The FALLBACK_PARTLY_REALISED state should be re-entered instead. A retry of the fallback is allowed so that it is possible to recover after transient condition that caused a fallback to fail or partly realise are no longer present.



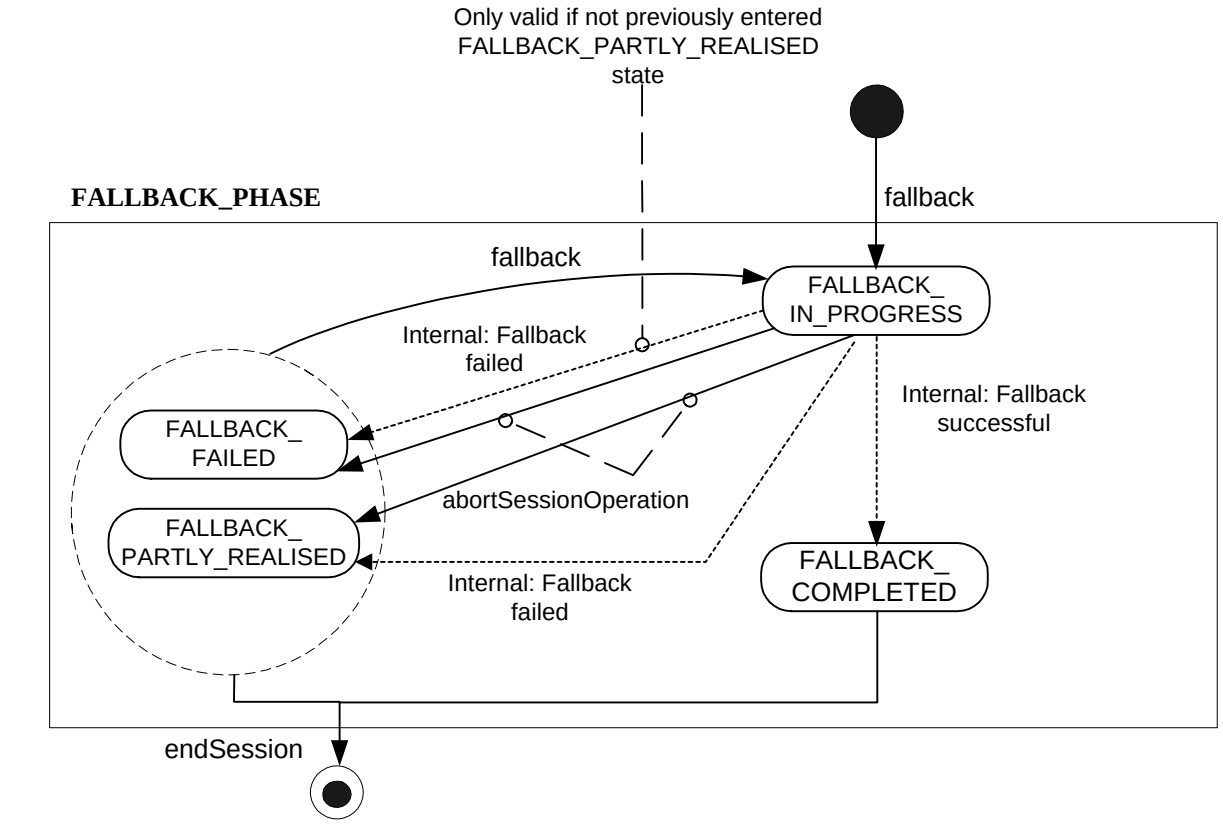


Figure 10: Substate Machine – FALLBACK_PHASE

~~7~~9.3 State Machine Pre and Post Conditions Tables

For each operation Table 15 identifies the state machine pre and post conditions.

Table 1~~5~~: State Machine Pre and Post Conditions

Operation	Pre-condition	Post Condition
startSession	No state – input sessionId provided by an IRPManager is not already in use in the IRPAgent by this or any other IRPManager	State = IDLE
endSession	not in a Transition status i.e. state <>. *_IN_PROGRESS	sessionId is released - No state.
upload	State = IDLE or UPLOAD_FAILED	Initially while operation is being performed: State= UPLOAD_IN_PROGRESS Finally when operation has completed: State = UPLOAD_COMPLETED or UPLOAD_FAILED
download	State = IDLE or DOWNLOAD_FAILED	Initially while operation is being performed: State= DOWNLOAD_IN_PROGRESS Finally when operation has completed: State = DOWNLOAD_COMPLETED or DOWNLOAD_FAILED
validate	State = DOWNLOAD_COMPLETED or VALIDATION_FAILED	Initially while operation is being performed: State= VALIDATION_IN_PROGRESS Finally when operation has completed: State = VALIDATION_COMPLETED or VALIDATION_FAILED
preactivate	State = DOWNLOAD_COMPLETED or VALIDATION_COMPLETED or PREACTIVATION_PARTLY_REALISED or PREACTIVATION_FAILED	Initially while operation is being performed: State= PREACTIVATION_IN_PROGRESS Finally when operation has completed: State = PREACTIVATION_COMPLETED or PREACTIVATION_PARTLY_REALISED or PREACTIVATION_FAILED
activate	State = DOWNLOAD_COMPLETED or VALIDATION_COMPLETED or ACTIVATION_PARTLY_REALISED or ACTIVATION_FAILED or PREACTIVATION_COMPLETED or PREACTIVATION_PARTLY_REALISED or PREACTIVATION_FAILED	Initially while operation is being performed: State= ACTIVATION_IN_PROGRESS Finally when operation has completed: State = ACTIVATION_COMPLETED or ACTIVATION_PARTLY_REALISED or ACTIVATION_FAILED
fallback	State = PREACTIVATION_COMPLETED or PREACTIVATION_PARTLY_REALISED or ACTIVATION_COMPLETED or ACTIVATION_PARTLY_REALISED or FALLBACK_PARTLY_REALISED or FALLBACK_FAILED or FALLBACK_PARTLY_REALISED or FALLBACK_FAILED	Initially while operation is being performed: State= FALLBACK_IN_PROGRESS Finally when operation has completed: State = FALLBACK_COMPLETED or FALLBACK_PARTLY_REALISED or FALLBACK_FAILED
abortSessionOperation	State = UPLOAD_IN_PROGRESS or DOWNLOAD_IN_PROGRESS or VALIDATION_IN_PROGRESS or PREACTIVATION_IN_PROGRESS or ACTIVATION_IN_PROGRESS or FALLBACK_IN_PROGRESS	State = UPLOAD_FAILED or DOWNLOAD_FAILED or VALIDATE_FAILED or PREACTIVATION_PARTLY_REALISED or PREACTIVATION_FAILED or ACTIVATION_PARTLY_REALISED or ACTIVATION_FAILED or FALLBACK_PARTLY_REALISED or FALLBACK_FAILED
getSessionIds	N/A – State Machine independent	N/A
getSessionStatus	None	None
getSessionLog	None	None
getBulkCmIRPversion	N/A – State Machine independent	N/A

~~8~~10 Bulk Configuration Data File

The overall management of Bulk CM is controlled by the operations in subclause ~~6.2.2~~7. Unitary management information is aggregated into a configuration data file for bulk CM operations. The file can be used for active and passive CM.

Bulk configuration data files consist of one or more blocks. Each block contains one or more object containment trees defined by a standardised language, for example XML. The basic building block (node) of this tree is a specifically-typed MO. This MO is identified by an ID attribute (the Naming attribute used in the RDN), and contains (1) data associated with the MO, and (2) zero or more children nodes. The structure and content of the MO data is constrained by the possible types of contained objects for the CM NRM that is being managed by Bulk CM IRP IS.

The file structure is the same for both upload and download bulk CM operations, apart that for active bulk CM operations, as well as containing MO data the blocks also specify the management actions (sub-operations) associated with each MOs item in the file. The following management actions (sub-operations) on MOs are supported for active bulk CM:

- Create MO. (sub-clause ~~8~~10.1.1)
- Delete MO. (sub-clause ~~8~~10.1.2)
- Change one or more existing MO attribute values. (sub-clause ~~8~~10.1.3)

The rules for ordering management actions in the configuration data file are defined in sub-clause ~~8~~10.2.

~~8~~10.1 Bulk Configuration Data Management Actions – Sub-operations

By the nature of active Bulk CM IRP, in the download bulk configuration file all sub-operation parameters identified in the following sub-clauses ~~8~~10.1.1 – ~~8~~10.1.3 are "input " only. Bulk CM IRP:IS will not generate any explicit notifications or responses for each sub-operation. The resulting session log and output(s) from the associated Bulk CM operations will record and convey the overall result of the sub-operations in the bulk configuration data file. The IRPAgent can record the outcome of relevant sub-operations in the session log. The IRPManager can subsequently get the session log (sub-clause ~~6.2.2.10~~7.3.6) if it is required to make a detailed analysis.

It should be noted other IRPs can generate notifications as a result of Bulk CM: IS sub-operations if an IRPAgent implements Basic CM IRP. The rules and definitions for these notifications are beyond the scope of this document. The NRMs identified in sub-clause 6.~~3~~ 4 and references [4], [5] and [6] give further details of which MOCs may generate Basic CM IRP notifications as a consequence of the sub-operations defined here.

~~8~~10.1.1 bulkCmCreateMo (Create MO Sub-operation) (M)

The IRPManager associates this sub-operation with an MOI in the configuration data file to request the IRPAgent to create the MOI.

~~Table 16~~ bulkCmCreateMo parameters

Name	Qualifier	Description
objectClass	Input, M	Identifies the NRM MOC within the scope of sub-clause 6.~~3~~ 4 that is to be created.
objectInstance	Input, M	Identifies the NRM MOC instance that is to be created.
attributeList	Input, O	Empty, or one or more attribute name and value pairs valid for the MOC. See sub-clause 6.~~3~~4. If the list is not empty the indicated attributes will be set to their indicated values when the object is created.

~~8~~10.1.2 bulkCmDeleteMo (Delete MO Sub-operation) (M)

The IRPManager associates this sub-operation with an MOI in the configuration data file to request the IRPAgent to delete the MOI.

~~Table 17~~ bulkCmDeleteMo parameters

Name	Qualifier	Description
objectClass	Input, M	Identifies the NRM MOC within the scope of sub-clause 6.3 that is to be deleted.
objectInstance	Input, M	Identifies the NRM MOC instance that is to be deleted.

~~8~~10.1.3 bulkCmChangeMo (Change MO Sub-operation) (M)

The IRPManager associates this sub-operation with an MOI in the configuration data file to request the IRPAgent to change/set one or more attributes of the MOI.

~~Table 18:~~ bulkCmChangeMo parameters

Name	Qualifier	Description
objectClass	Input, M	Identifies the NRM MOC within the scope of sub-clause 6.~~3~~ 4 that the attributes are to be changed.
objectInstance	Input, M	Identifies the NRM MOC instance for which the attributes are to be changed.
attributeList	Input, M	One or more attribute name and value pairs valid for the MOC. See sub-clause 6.3. The indicated attributes of the MOC instance will be changed/set to their indicated values.

~~8~~10.2 Rules for ordering Management Actions (Sub-operations) in Configuration Data Files

~~8~~10.2.1 Download files

1. The IRP Manager shall enter the management actions into the configuration data file in the order they are to be interpreted and actioned by the IRPAgent following its sequentially step-by-step single pass operation. The IRPManager has overall responsibility for ensuring the correct order of action is given according to the rules in this sub-clause.

2. The IRPAgent shall interpret the management actions in the configuration data file sequentially step-by-step in a single pass operation. The IRPManager has overall responsibility for ensuring the correct order of action is given.

3. The permitted order shall follow NRM hierarchy subtree(s) of the Managed Object instances pertaining to the configuration data file.

4. All delete MOs actions shall precede any Create MOs actions.

5. This document does not specify any limitations on the ordering of change MO attribute actions other than the impacted if the impacted MO does not already exist it needs to be created by a prior create action. The choice of standardised language may recommend or specify some additional constraints e.g. for reasons of efficiency or for compliance with language syntax. Such recommendation and constraints are beyond the scope of this document

6. All necessary MO changes supported by Bulk CM IRP interface-N need to be fully specified in a configuration data file to maintain consistency within the NRM MIB subtree being operated on. (e.g. if an object is to be deleted, all relations and associations shall be removed).

7. All relations to an MO instance shall be removed prior to deleting an MO instance.

8. When part or whole NRM subtree is to be deleted, in the configuration data file the IRPManager shall first action delete of all associated child instances contained in the NRM subtree before actioning delete of MO parents instances i.e. delete actions on MO instances shall be specified in a recursive manner following the NRM hierarchy subtree from the lowest MO instances to the highest MO instances the IRPManager requires to be deleted. (The IRPAgent will not support autonomous deletion of all MO instance contained in a NRM subtree identified by a single delete action of the highest MO instance of the subtree).

9. When part or a whole NRM subtree is to be created, in the configuration data file the IRPManager shall first action the create action of parents MO instances before actioning the create of any child MO instances

contained in the NRM subtree i.e. create actions on MO instances shall be specified in recursive manner following the NRM hierarchy subtree from the highest MO instances to the lowest MO instances the IRPManager requires to be created.

~~8~~10.2.2 Upload files

1. No rules are identified i.e. it is not necessary that they be part of the scope of this document. They may be implementation specific and specified in other document as part of a specific solution.

Annex A (informative): Scenarios

~~Draft~~ Ssupporting background informational only.

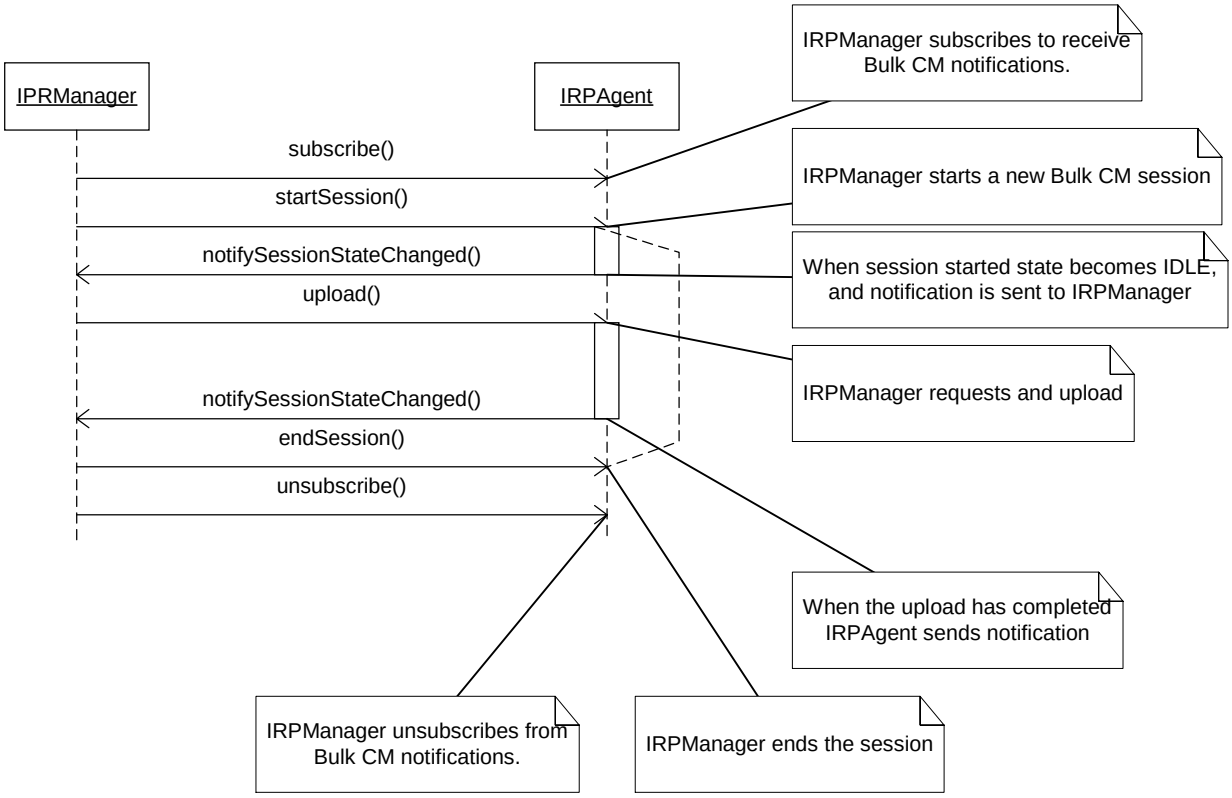
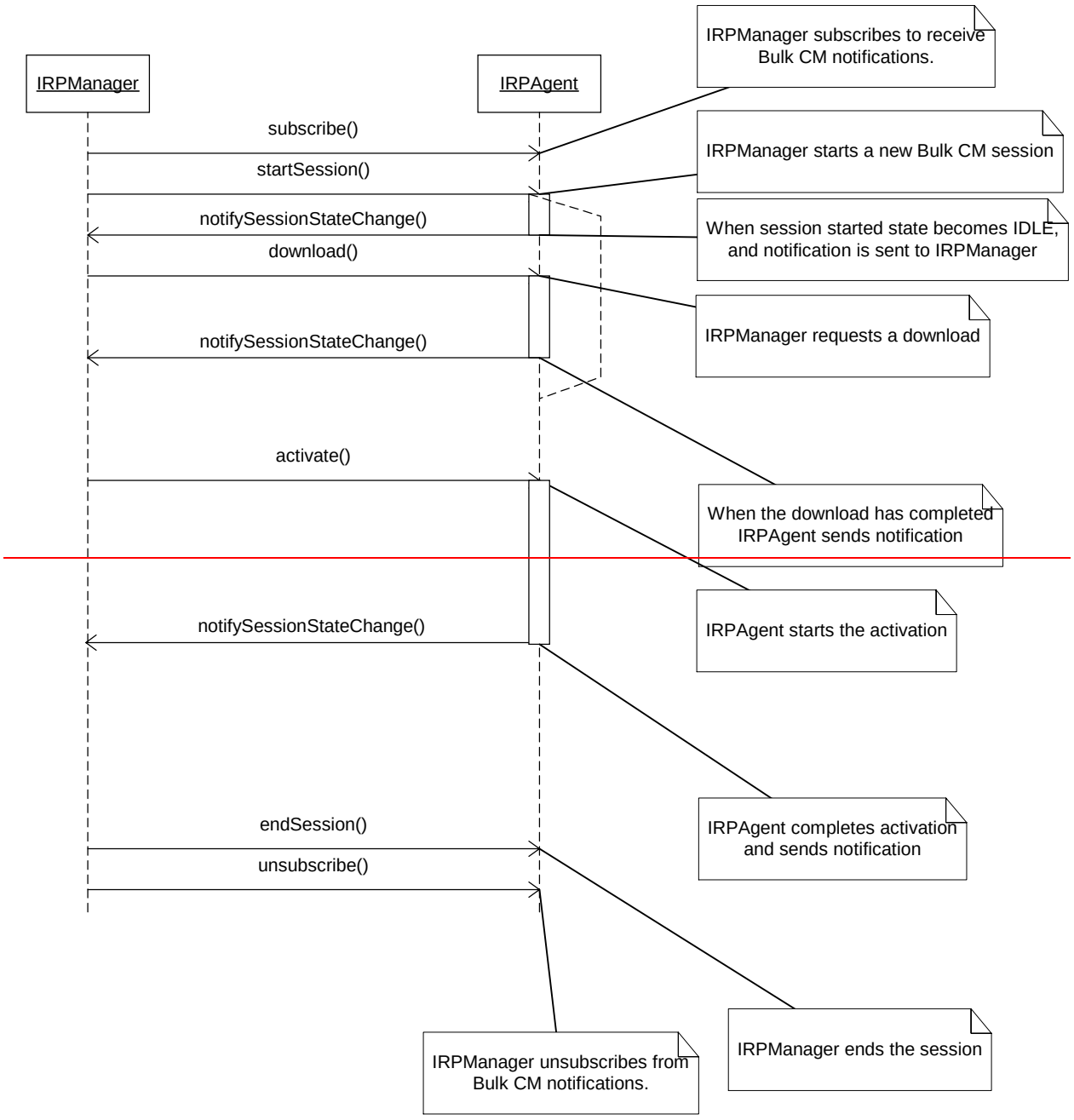


Figure A.1: Example 1: Successful Upload Session


IRPManager
IRPAgent
IRPManager subscribes to receive Bulk CM notifications.
subscribe()
IRPManager starts a new Bulk CM session
startSession()
notifySessionStateChange()
When session started state becomes IDLE, and notification is sent to IRPManager
download()
IRPManager requests a download
notifySessionStateChange()
activate()
When the download has completed IRPAgent sends notification
IRPAgent starts the activation
notifySessionStateChange()
IRPAgent completes activation and sends notification
endSession()
unsubscribe()
IRPManager unsubscribes from Bulk CM notifications.
IRPManager ends the session

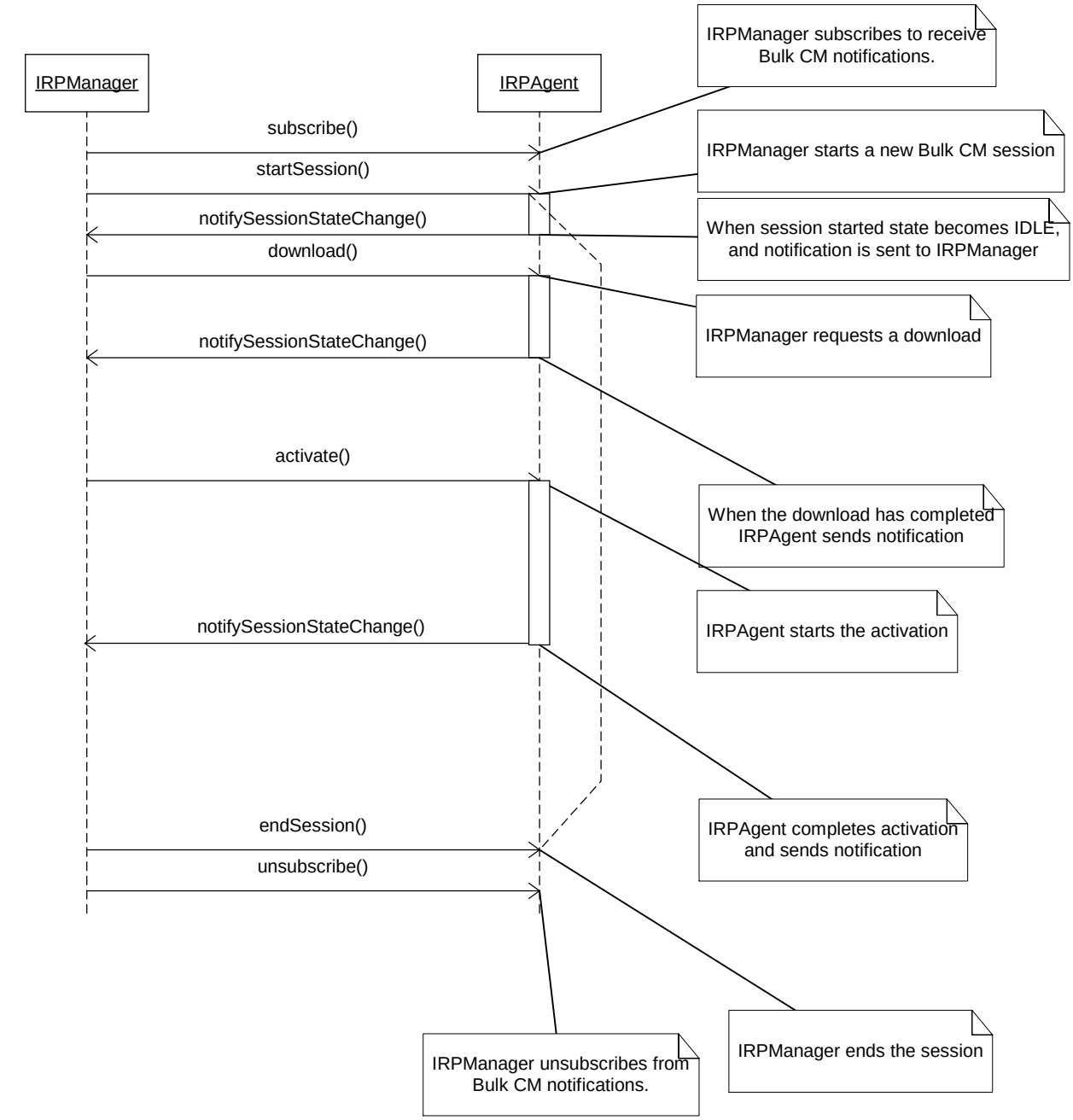


Figure A.2: Example 2: Successful Download and Activation without validation and preactivation.~~Activate~~

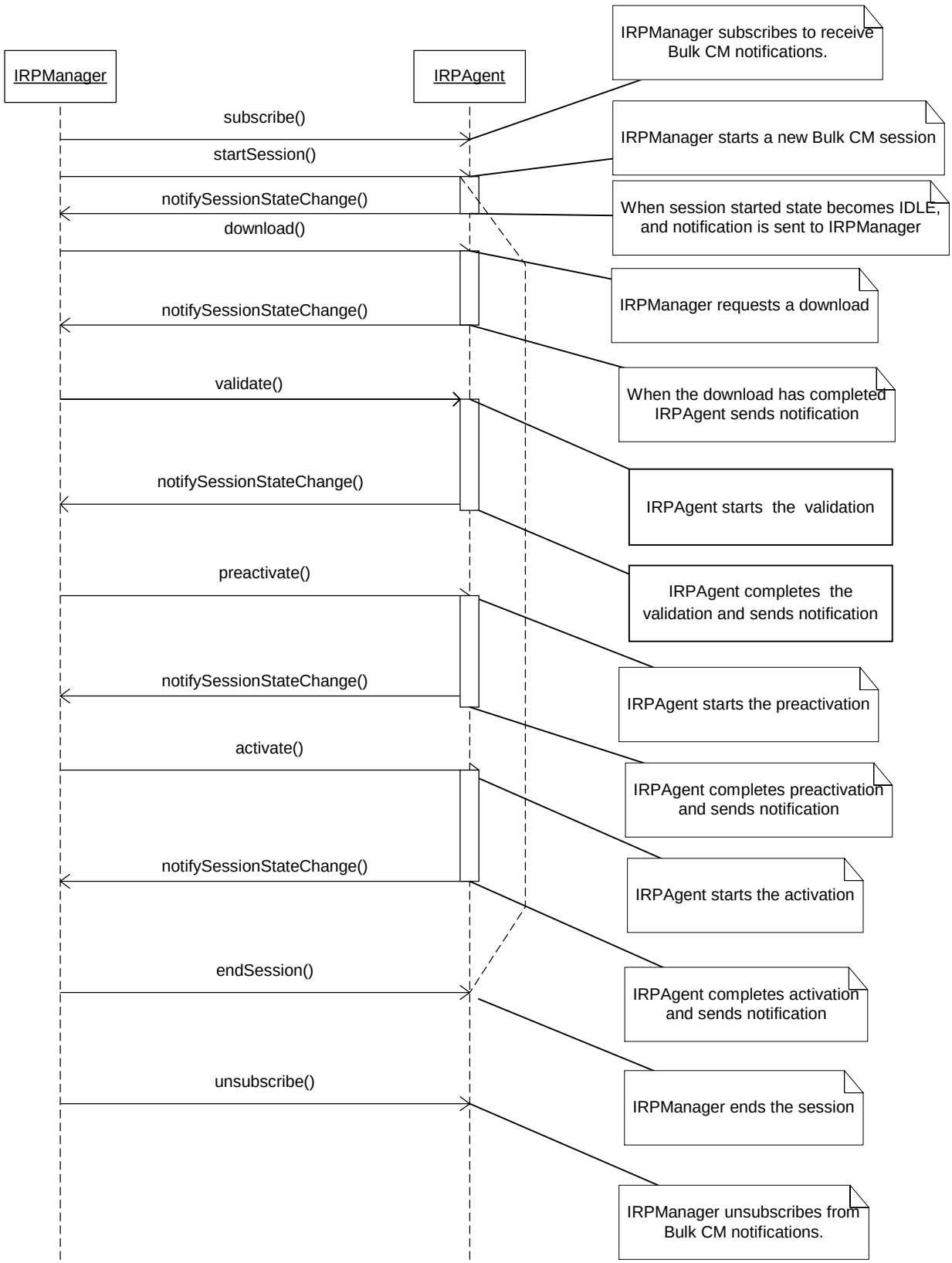


Figure A.3: Example 3: Successful Download and Activation with validation and preactivation.

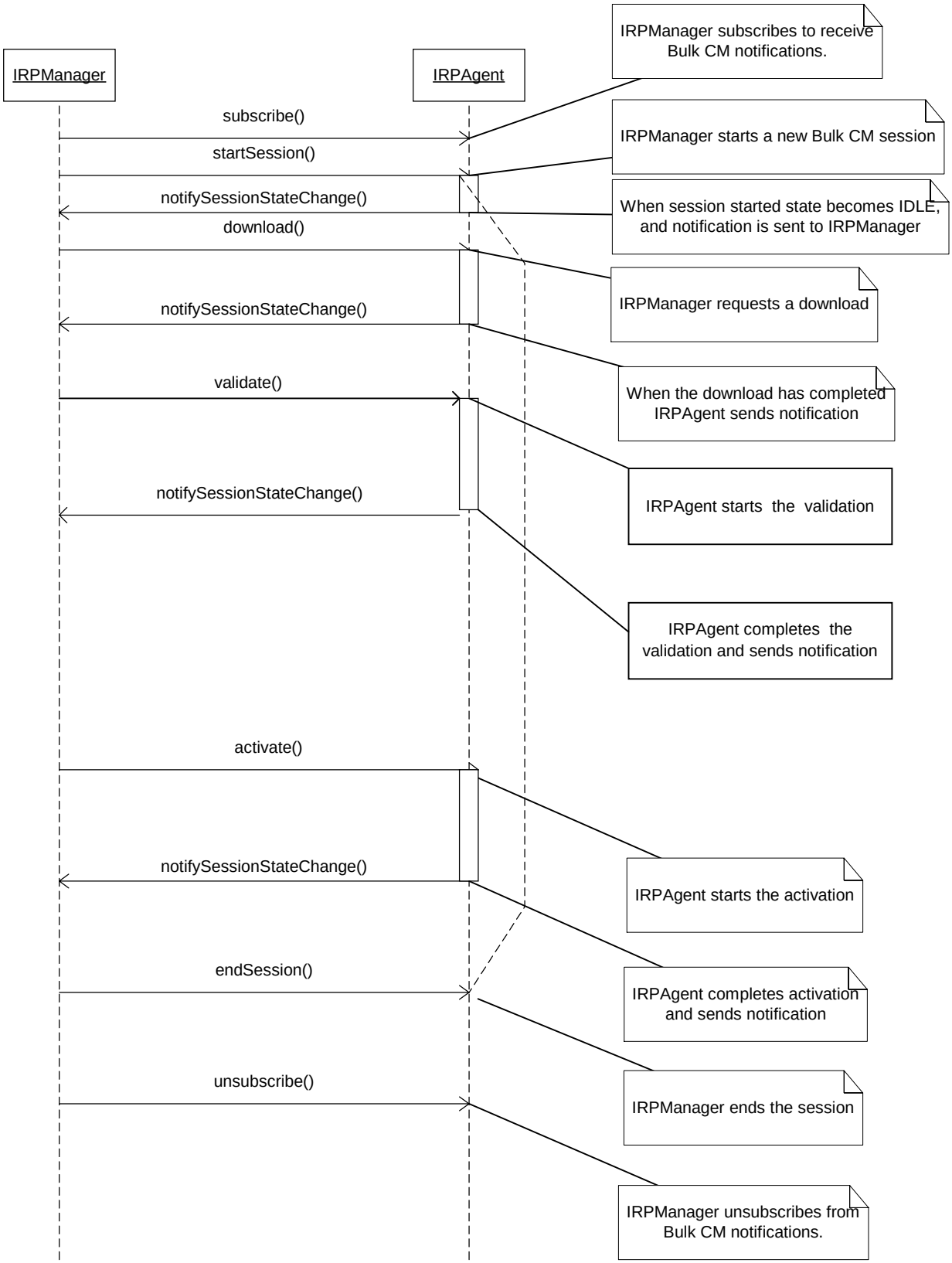


Figure A.4: Example 4: Successful Download and Activation with Validation.

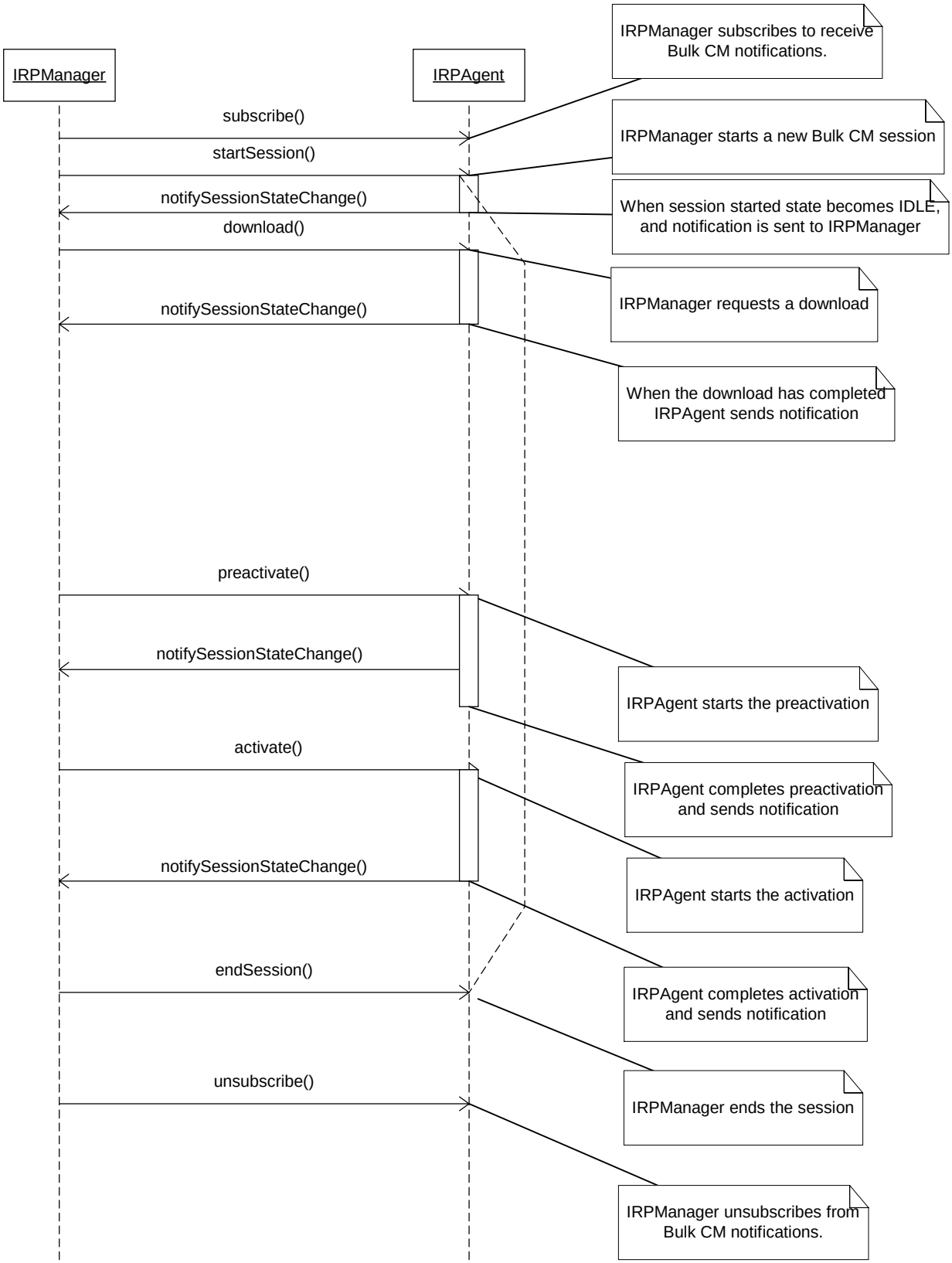


Figure A.5: Example 5: Successful Download and Activation with Preactivation.

Annex B (informative): Bulk CM Application and Operation Principles

B1 Key Characteristics

- Bulk CM operations are not transaction based.
- The state machine does not allow looping. Can only progress forward through main states.
- If any errors are found in the configuration data, it shall not be possible to fix the configuration data. A new session should be started with new corrected configuration data being downloaded..
- Non-transitional interface (editors note: clarify)
- Sessions may be run in parallel. There shall not be any exclusion of specified changes between pararellel sessions. (Editors note: a “lock” option may be added, subject to further contribution).
- (Editors note: include some example exceptions)
-

3GPP TSG-SA5 (Telecom Management) Meeting #30 **S5-026734**
Tampere, Finland, 19 – 23 August 2002

CR-Form-v7

CHANGE REQUEST

⌘ **32.613** **CR** **005** ⌘ **rev** **-** ⌘ Current version: **4.2.0** ⌘

*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:	⌘	Add Bulk CM IRP CORBA Solution Set Enhancements Rel-5
Source:	⌘	S5
Work item code:	⌘	OAM-NIM
Date:	⌘	23/08/2002
Category:	⌘	**C**
Release:	⌘	REL-5

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:	⌘	Support of Validation and Preactivation Modes
Summary of change:	⌘	Add new operations validation and preactivate. Add Mode parameters. Other changes suggested by #25, #26, #27, #27quad, #28, #29 and email discussion.
Consequences if not approved:	⌘	Validation and Preactivation Modes not supported.

Clauses affected:	⌘	Sub-clause Introduction, 3.3, 4.2, Table 1,2,3,4,5,6,7,8,9,10,11,13,14,1516,17,18, 4.5, 4.6, Annex A, Annex B.

Other specs affected: ⌘

Y	N	
	X	Other core specifications
	X	Test specifications
	X	O&M Specifications

⌘

Other comments: ⌘

S5-026688	Rel-5 CR 32611 Additional Bulk CM IRP requirements for Rel-5 - Parent CR
S5-026733	Rel-5 CR 32612 Bulk CM IS - R5 Enhancements and New Methodology - Child CR
S5-026734	Rel-5 CR 32613 Bulk CM CORBA SS - R5 Enhancements - Grandchild CR

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.
2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

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

R99 Old no.	Old (R99) specification title	Rel-4 New no.	New (Rel-4) specification title
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	**Notification IRP**: Requirements
32.106-2	Notification IRP: IS	32.302	Notification IRP: Information Service
32.106-3	Notification IRP: CORBA SS	32.303	Notification IRP: CORBA SS
32.106-4	Notification IRP: CMIP SS	32.304	Notification IRP: CMIP SS
32.106-8	Name convention for Managed Objects	32.300	**Name Convention for Managed Objects**
32.106-1	<Basic CM IRP IS requirements from 32.106-1 and 32.106-5>	32.601	**Basic CM IRP**: Requirements
32.106-5	Basic CM IRP IM (Intro & IS part)	32.602	Basic CM IRP: Information Service
32.106-6	Basic CM IRP CORBA SS (IS related part)	32.603	Basic CM IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (IS related part)	32.604	Basic CM IRP: CMIP SS
32.106-8	Name convention for Managed Objects	32.300	**Name Convention for Managed Objects**
-	-	32.611	**Bulk CM IRP**: Requirements
-	-	32.612	Bulk CM IRP: Information Service
-	-	32.613	Bulk CM IRP: CORBA SS
-	-	32.614	Bulk CM IRP: CMIP SS
		32.615	Bulk CM IRP: XML file format definition
32.106-1	<Basic CM IRP Generic NRM requirements from 32.106-1 and 32.106-5>	32.621	**Generic Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (Generic NRM part)	32.622	Generic Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (Generic NRM related part)	32.623	Generic Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (Generic NRM related part)	32.624	Generic Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP CN NRM requirements from 32.106-1 and 32.106-5>	32.631	**Core Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (CN NRM part)	32.632	Core Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (CN NRM related part)	32.633	Core Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (CN NRM related part)	32.634	Core Network Resources IRP: CMIP SS
32.106-1	<Basic CM IRP UTRAN NRM requirements from 32.106-1 and 32.106-5>	32.641	**UTRAN Network Resources IRP**: Requirements
32.106-5	Basic CM IRP IM (UTRAN NRM part)	32.642	UTRAN Network Resources IRP: NRM
32.106-6	Basic CM IRP CORBA SS (UTRAN NRM related part)	32.643	UTRAN Network Resources IRP: CORBA SS
32.106-7	Basic CM IRP CMIP SS (UTRAN NRM related part)	32.644	UTRAN Network Resources IRP: CMIP SS
		32.651	**GERAN Network Resources IRP**: Requirements
		32.652	GERAN Network Resources IRP: NRM
		32.653	GERAN Network Resources IRP: CORBA SS
		32.654	GERAN Network Resources IRP: CMIP SS

:

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 32.612 [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 32.622 [4], UTRAN Network Resources IRP: NRM 3GPP TS 32.642 [11], GERAN Network Resources IRP: NRM 3GPP TS 32.652 [12].

This Solution Set specification is related to 3GPP TS 32.612 V5.0.X.

:

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”.

Take the 3GPP document number on the front page of this specification, such as “3GPP TS 32.613 V5.0.0 (2002-09)”. Discard the leading “3GPP TS ”. Discard all characters after and including the last period. Eliminate leading and trailing spaces. Reduce multiple consecutive spaces with one space. Express the resultant in a string. Capitalised the string. For example, if the 3GPP document version number is “3GPP TS 32. 613 V5.0.0 (2002-09)”, then the IRP document version number shall be “32.613 V5.0”.

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.

:

:

4.2 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
get~~BulkCm~~IRPVersion	get_bulk_CM_IRP_versions	M
notifySessionStateChanged	push_structured_event Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 5.1	M
notifyGetSessionLogEnded	push_structured_event Note that OMG Notification Service OMG Notification Service [1] defines this method. See clause 5.1.	M
preactivate	preactivate	O
validate	validate	O
getOperationProfile	get_bulk_CM_IRP_operation_profile	O
getNotificationProfile	get_bulk_CM_IRP_notification_profile	O

4.3 Operation Parameter Mapping

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

Table 2: Mapping from IS startSession parameters to SS equivalents

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

Table 3: Mapping from IS endSession parameters to SS equivalents

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

Table 4: Mapping from IS `upload parameters` to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
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, exception ManagedGenericIRPSystem::InvalidParameter	M

NOTE : The IllegalURLFormatException does not imply that the transfer protocol used must be a URL. The transfer protocol is dependant on the file format definition, i.e. in the case of XML, FileDestination will be a URL.

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

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
downloadDataFileReference	BulkCmIRPConstDefs::FileDestination source	M
status	exception DownloadException, exception UnknownSessionIdException, exception MaxSessionReachedException, exception NotValidInCurrentStateException, exception IllegalURLFormatException, exception ManagedGenericIRPSystem::InvalidParameter	M

NOTE : The IllegalURLFormatException does not imply that the transfer protocol used must be a URL. The transfer protocol is dependant on the file format definition, i.e. in the case of XML, FileDestination will be a URL.

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

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

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

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

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

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

Table 10: Mapping from IS `getSessionStatus` parameters to SS equivalents

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

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

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
logFileReference	BulkCmIRPConstDefs::FileDestination sink	M
contentType	boolean only_error_info	M
status	exception GetSessionLogException, exception UnknownSessionIdException, ~~exception ConcurrencyException,~~ exception IllegalURLFormatException, exception ManagedGenericIRPSystem::InvalidParameter	M

NOTE : The IllegalURLFormatException does not imply that the transfer protocol used must be a URL. The transfer protocol is dependant on the file format definition, i.e. in the case of XML, FileDestination will be a URL.

Table 13: Mapping from IS `validate` parameters to SS equivalents

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

Table 14: Mapping from IS `preactivate` parameters to SS equivalents

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

Table 15: Mapping from IS `getOperationProfile` parameters to SS equivalents

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

Table 16: Mapping from IS `getNotificationProfile` parameters to SS equivalents

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

Table 173: Mapping from IS `notifyGetSessionLogEnded` parameters to SS equivalents

IS Parameter	OMG CORBA Structured Event Attribute	Qualifier	Comment
There is no corresponding IS attribute.	domain_name	M	It carries the IRP document version number string. See sub-clause 3.3. It indicates the syntax and semantics of the Structured Event as defined by this specification.
notificationType	type_name	M	It carries the string NOTIFY_GET_SESSION_LOG_ENDED.
sessionLogStatus	event_name	M	It carries either the string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY or GET_SESSION_LOG_COMPLETED_UNSUCCESSFULLY. In the case of the latter, the NV pair indicating ERROR_INFORMATION may be present.
There is no corresponding IS 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 184: Mapping from IS `notifySessionStateChanged` parameters to SS equivalents

IS Parameter	OMG CORBA Structured Event attribute	Qualifier	Comment
There is no corresponding IS attribute	domain_name	M	It carries the IRP document version number string. See sub-clause 3.3. It indicates the syntax and semantics of the Structured Event as defined by this specification.
notificationType	type_name	M	It carries the string NOTIFY_SESSION_STATE_CHANGED. This is an attribute of Header defined in the IS.

sessionState	event_name	M	It carries one of the following:
			`UPLOAD_FAILED`
			`UPLOAD_COMPLETED,`
			`DOWNLOAD_FAILED,`
			`DOWNLOAD_COMPLETED,`
			`ACTIVATION_FAILED,`
			`ACTIVATION_PARTLY_REALISED,`
			`ACTIVATION_COMPLETED,`
			`FALLBACK_FAILED,`
			`FALLBACK_PARTLY_REALISED,`
			`FALLBACK_COMPLETED,`
			`VALIDATION_FAILED,`
			`VALIDATION_COMPLETED,`
			`PREACTIVATION_FAILED,`
			`PREACTIVATION_PARTLY_REALISED,`
			`PREACTIVATION_COMPLETED`
			
			In the case of XXX_FAILED and XXX_PARTLY_REALISED, the NV pair indicating ERROR_INFORMATION may be present.

There is no corresponding IS attribute	Variable Header		
managedObjectClass, managedObjectInstance	One NV pair of filterable_body_fields	M	NV stands for name-value pair. Order arrangement of NV pairs is not significant. The name of NV-pair is always encoded in string. Name of NV pair is the `MANAGED_OBJECT_INSTANCE` of `interface AttributeNameValue` of `module NotificationIRPConstDefs.` Value of NV pair is a string. See encoding of this string in [5]. These are attributes of Header defined in the IS.
notificationId	One NV pair of 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.

4.5 Two modes of operations

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

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

4.6 Mapping from IS State Names to SS equivalents

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

Table 195: Mapping from IS State Names to SS equivalents

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

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 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";
   };

●●●

   /*
   The format of Distinguished Name is specified in
   the Naming Conventions for Managed Objects; 3G TS 32.300 Annex H.
   e.g. "g3SubNetwork=10001,g3ManagedElement=400001" identifies an
```

```
   G3ManagedElement instance of the object model.
   */
   typedef string DistinguishedName;

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

●●●

   /*
   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 mandatory (defined byThe 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;     // optional parameter
   };

   /*
   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
```

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 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,
                ~~ConcurrencyException,~~ IllegalURLFormatException,
                ManagedGenericIRPSystem::InvalidParameter);

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

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

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

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

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

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

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

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

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

   };
};

#endif
```