ETSI TS 132 616 V13.0.0 (2016-02)

Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Telecommunication management;
Configuration Management (CM);
Bulk CM Integration Reference Point (IRP);
Solution Set (SS) definitions
(3GPP TS 32.616 version 13.0.0 Release 13)

Reference
RTS/TSGS-0532616vd00

Keywords
GSM,LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part of a TS-family covering the 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Telecommunication management; as identified below:

32.611: "Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Requirements".

32.612: "Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Information Service (IS)".

32.616: "Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Solution Set (SS) definitions".

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.

1 Scope

The present document specifies the Solution Sets for the IRP whose semantics are specified in Bulk CM IRP: Information Service (3GPP TS 32.612 [9]).

This Solution Set specification is related to 3GPP TS 32.612 V12.0.X.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 32.101: "Telecommunication management; Principles and high level requirements".

[2] 3GPP TS 32.102: "Telecommunication management; Architecture".

[3] 3GPP TS 32.150: "Telecommunication management; Integration Reference Point (IRP) Concept and definitions".

[4] 3GPP TS 32.300: "Telecommunication management; Configuration Management (CM); Name convention for Managed Objects".

[5] 3GPP TS 32.306: "Telecommunication management; Configuration Management (CM); Notification Integration Reference Point (IRP): Solution Set (SS) definitions".

[6] 3GPP TS 32.312: "Telecommunication management; Generic Integration Reference Point (IRP) management; Information Service (IS)".

[7] 3GPP TS 32.316: "Telecommunication management; Generic Integration Reference Point (IRP) management: Solution Set (SS) definitions".

[8] 3GPP TS 32.611: " Technical Specification Group Services and System Aspects; Telecommunication management; Configuration Management (CM); Bulk CM Integration Reference Point (IRP): Requirements ".

[9] 3GPP TS 32.612: "Telecommunication management; Configuration Management (CM); Bulk CM Integration Reference Point (IRP); Information Service (IS)".

[10] 3GPP TS 32.622: "Telecommunication management; Configuration Management (CM); Generic network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[11] 3GPP TS 32.626: "Telecommunication management; Configuration Management (CM); Generic network resources Integration Reference Point (IRP): Solution Set (SS) definitions".

[12] 3GPP TS 32.632: "Telecommunication management; Configuration Management (CM); CN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[13] 3GPP TS 32.636: "Telecommunication management; Configuration Management (CM); Core network resources Integration Reference Point (IRP): Solution Set (SS) definitions".

[14] 3GPP TS 32.642: "Telecommunication management; Configuration Management (CM); UTRAN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[15] 3GPP TS 32.646: "Telecommunication management; Configuration Management (CM); UTRAN network resources Integration Reference Point (IRP): Solution Set (SS) definitions".

[16] 3GPP TS 32.652: "Telecommunication management; Configuration Management (CM); GERAN network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[17] 3GPP TS 32.656: "Telecommunication management; Configuration Management (CM); GERAN network resources Integration Reference Point (IRP): Solution Set (SS) definitions".

[18] 3GPP TS 32.692 "Inventory Management (IM) network resources Integration Reference Point (IRP): Network Resource Model (NRM)".

[19] 3GPP TS 32.696: "Telecommunication management; Inventory Management (IM) Network Resource Model (NRM); Integration Reference Point (IRP): Solution Set (SS) definitions".

[20] 3GPP TS 32.716: "Telecommunication management; Configuration Management (CM); Transport Network (TN) interface Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[21] 3GPP TS 32.736: "IP Multimedia Subsystem (IMS) Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[22] 3GPP TS 32.742: "Telecommunication management; Configuration Management (CM); Signalling Transport Network (STN) Interface Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[23] 3GPP TS 32.746: "Telecommunication management; Configuration Management (CM); Signalling Transport Network (STN) Interface Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[24] 3GPP TS 32.756: " Telecommunication management; Evolved Packet Core (EPC) Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[25] 3GPP TS 32.766: "Telecommunication management; Evolved Universal Terrestrial Radio Access Network (E UTRAN) Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[26] OMG Notification Service, Version 1.0.

[27] W3C REC-xml-20001006: "Extensible Markup Language (XML) 1.0 (Second Edition)".

[28] W3C REC-xmlschema-0-20010502: "XML Schema Part 0: Primer".

[29] W3C REC-xmlschema-1-20010502: "XML Schema Part 1: Structures".

[30] W3C REC-xmlschema-2-20010502: "XML Schema Part 2: Datatypes".

[31] W3C REC-xml-names-19990114: "Namespaces in XML".

[32] W3C SOAP 1.1 specification (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

[33] W3C XPath 1.0 specification (http://www.w3.org/TR/1999/REC-xpath-19991116)

[34] W3C WSDL 1.1 specification (http://www.w3.org/TR/2001/NOTE-wsdl-20010315)

[35] W3C SOAP 1.2 specification (http://www.w3.org/TR/soap12-part1/)

[36] 3GPP TS 32.172: "Telecommunication management; Subscription Management (SuM) Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[37] 3GPP TS 32.176: "Telecommunication management; Subscription Management (SuM) Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[38] 3GPP TS 32.522: "Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[39] 3GPP TS 32.526: "Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

[40] 3GPP TS 32.712: "Telecommunication management; Configuration Management (CM); Transport Network (TN) Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[41] 3GPP TS 32.722: "Telecommunication management; Configuration Management (CM); Repeater network resources Integration Reference Point (IRP); information Service (IS)".

[42] 3GPP TS 32.726: "Telecommunication management; Configuration Management (CM); Repeater network resources Integration Reference Point (IRP): Solution Set (SS) definitions".

[43] 3GPP TS 32.732: "Telecommunication management; IP Multimedia Subsystem (IMS) Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[44] 3GPP TS 32.752: "Telecommunication management; Evolved Packet Core (EPC) Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[45] 3GPP TS 32.762: "Telecommunication management; Evolved Universal Terrestrial Radio Access Network (E UTRAN) Network Resource Model (NRM) Integration Reference Point (IRP): Information Service (IS)".

[46] 3GPP TS 32.772: "Telecommunication management; Home Node B (HNB) Subsystem (HNS); Network Resource Model (NRM); Integration Reference Point (IRP): Information Service (IS)".

[47] 3GPP TS 32.776: "Telecommunication management; Home Node B (HNB) Subsystem (HNS); Network Resource Model (NRM); Integration Reference Point (IRP): Solution Set (SS) definitions".

[48] 3GPP TS 32.782: "Telecommunication management; Home enhanced Node B (HeNB) Subsystem (HeNS); Network Resource Model (NRM); Integration Reference Point (IRP): Information Service (IS)".

[49] 3GPP TS 32.786: "Telecommunication management; Home enhanced Node B (HeNB) Subsystem (HeNS); Network Resource Model (NRM); Integration Reference Point (IRP): Solution Set (SS) definitions".

[50] 3GPP TS 32.796: "Telecommunication management; Generic Radio Access Network (RAN) Network Resource Model (NRM) Integration Reference Point (IRP): Solution Set (SS) definitions".

3 Definitions and abbreviations

3.1 Definitions

For terms and definitions please refer to TS 32.101 [1], TS 32.102 [2], TS 32.150 [3], TS 32.172 [36], TS 32.522 [38], TS 32.611 [8], TS 32.612 [9], TS 32.622 [10], TS 32.632 [12], TS 32.642 [14], TS 32.652 [16], TS 32.692 [18], TS 32.712 [40], 32.722 [41], TS 32.732 [43], TS 32.742 [22], TS 32.752 [44], TS 32.762 [45], TS 32.772 [46] and TS 32.782 [48].

For the purposes of the present document, the following terms and definitions apply.

IRP document version number string (or "IRPVersion"): See 3GPP TS 32.312 [6].

XML file: a file containing an XML document.

XML document: see [27]; in the scope of this specification, an XML document is composed of the succession of an optional XML declaration followed by a root XML element.

XML declaration: see [27]; it specifies the version of XML and the character encoding being used.

XML element: see [27]; an XML element has a type, is identified by a name, may have a set of XML attribute specifications and is either composed of the succession of an XML start-tag followed by the XML content of the XML element followed by an XML end-tag, or composed simply of an XML empty-element tag; each XML element may contain other XML elements.

empty XML element: see [27]; an XML element having an empty XML content; an empty XML element still possibly has a set of XML attribute specifications; an empty XML element is either composed of the succession of an XML start-tag directly followed by an XML end-tag, or composed simply of an XML empty-element tag.

XML content (of an XML element): empty if the XML element is simply composed of an XML empty-element tag; otherwise the part, possibly empty, of the XML element between its XML start-tag and its XML end-tag.

XML start-tag: see [27]; the beginning of a non-empty XML element is marked by an XML start-tag containing the name and the set of XML attribute specifications of the XML element.

XML end-tag: see [27]; the end of a non-empty XML element is marked by an XML end-tag containing the name of the XML element.

XML empty-element tag: see [27]; an empty XML element is composed simply of an empty-element tag containing the name and the set of XML attribute specifications of the XML element.

XML attribute specification: see [27]; an XML attribute specification has a name and a value.

DTD: see [27]; a DTD defines structure and content constraints to be respected by an XML document to be valid with regard to this DTD.

XML schema: see [28], [29] and [30]; more powerful than a DTD, an XML schema defines structure and content constraints to be respected by an XML document to conform with this XML schema; through the use of XML namespaces several XML schemas can be used together by a single XML document; an XML schema is itself also an XML document that shall conform with the XML schema for XML schemas.

XML namespace: see [31]; in the scope of this specification, enables qualifying element and attribute names used in XML documents by associating them with namespaces identified by different XML schemas.

XML complex type: see [28], [29] and [30]; defined in an XML schema; cannot be directly used in an XML document; can be the concrete type or the derivation base type for an XML element type or for another XML complex type; ultimately defines constraints for an XML element on its XML attribute specifications and/or its XML content.

XML element type: see [28], [29] and [30]; declared by an XML schema; can be directly used in an XML document; as the concrete type of an XML element, directly or indirectly defines constraints on its XML attribute specifications and/or its XML content; can also be the concrete type or the derivation base type for another XML element type.

3.2 Abbreviations

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

CORBA	Common Object Request Broker Architecture
CM	Configuration Management
DN	Distinguished Name
DTD	Document Type Definition
EDGE	Enhanced Data for GSM Evolution
GERAN	GSM/EDGE Radio Access Network
GSM	Global System for Mobile communication
IDL	Interface Definition Language (OMG)
IRP	Integration Reference Point
IS	Information Service
MO	Managed Object
MOC	Managed Object Class
NRM	Network Resource Model
OMG	Object Management Group
RDN	Relative Distinguished Name
SS	Solution Set
UMTS	Universal Mobile Telecommunications System
UTRAN	Universal Terrestrial Radio Access Network
WSDL	Web Service Description Language
XML	eXtensible Markup Language

4 Solution Set Definitions

This specification defines the following 3GPP Bulk CM IRP Solution Set Definitions:

Annex A provides the CORBA Solution Set.
Annex B provides the XML Definitions.
Annex C provides the SOAP Solution Set.

Annex A (normative): CORBA Solution Set

This annex specifies the CORBA Solution Set for the IRP whose semantics are specified in 3GPP TS 32.612 [9].

A.1 Architectural features

The overall architectural feature of Bulk CM IRP is specified in 3GPP TS 32.612 [9].
This clause specifies features that are specific to the CORBA SS.

A.1.1 Syntax for Distinguished Names

The syntax of a Distinguished Name is defined in 3GPP TS 32.300 [4].

A.1.2 BulkCMIRPNotifications Interface

OMG CORBA Notification push operation is used to realise the notification of BulkCMIRPNotifications. All the notifications in this interface are implemented using this push_structured_event method.

A.1.2.1 Method push (M)

```
module CosNotifyComm {
        …
        Interface SequencePushConsumer : NotifyPublish {
                void push_structured_events(
                        in CosNotification::EventBatch notifications)
            raises( CosEventComm::Disconnected);
                …
        }; // SequencePushConsumer
        …
}; // CosNotifyComm
```

NOTE 1: The push_structured_events method takes an input parameter of type EventBatch as defined in the OMG CosNotification module (OMG Notification Service [26]). This data type is the same as a sequence of Structured Events. Upon invocation, this parameter will contain a sequence of Structured Events being delivered to IRPManager by IRPAgent to which it is connected.

NOTE 2: The maximum number of events that will be transmitted within a single invocation of this operation is controlled by IRPAgent wide configuration parameter.

NOTE 3: The amount of time the supplier (IRPAgent) of a sequence of Structured Events will accumulate individual events into the sequence before invoking this operation is controlled by IRPAgent wide configuration parameter as well.

NOTE 4: IRPAgent may push EventBatch with only one Structured Event.

A.2 Mapping

A.2.1 General mappings

All MOs are arranged in a **containment** structure, according to the containment relations defined in the NRM. This structure is held internally by the IRPAgent. Externally, the MO containment structure is defined by the semantics in the distinguished name syntax. The distinguished name (DN) for a MO contains the distinguished name of the parent plus the Relative DN for the MO itself.

Associations as defined in the NRM (UML) are in this document mapped to attributes in the MIB. The names of the roles for an association in the NRM are used for defining attribute names in the MIB. When the cardinality for a role is 0..1 or 1..1 the datatype for the attribute is defined as a MO reference. The value of a MO reference contains the distinguished name of the referred MO. When the cardinality for a role allows more than one referred MO instances, the attribute will contain a sequence of MO references (i.e., DNs).

A.2.2 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.

There are 3 qualifications for each row of the following mapping table.
The 3 qualifications correspond to the three IS-defined packages: Controlled Upload & Provisioning, Controlled Upload and Simple Upload.
Not all operations/notifications specified in the following table are required for all 3 packages.
An "-" indicates that the subject operation or notification is not allowed by that corresponding package.

Table A.2.2: Mapping from IM Notification/Operation to SS equivalents

IS Operation/ notification	SS Method	Qualifier
startSession	start_session	M,M,-
endSession	end_session	M,M,-
upload	upload	M,M,M
download	download	M,-,-
activate	activate	M,-,-
getSessionStatus	get_session_status	M,M,-
getSessionIds	get_session_ids	M,M,-
getSessionLog	get_session_log	M,M,-
fallback	fallback	M,-,-
abortSessionOperation	abort_session_operation	M,M,-
getIRPVersion	get_bulk_cm_irp_versions get_controlled_upload_bulk_cm_irp_versions get_simple_upload_bulk_cm_irp_versions	M,-,- -,M,- -,-,M
notifySessionStateChanged	push_structured_event Note that OMG Notification Service OMG Notification Service [26] defines this method. See clause A.1.2.	M,M,M
notifyGetSessionLogEnded	push_structured_event Note that OMG Notification Service OMG Notification Service [26] defines this method. See clause A.1.2.	M,M,-
preactivate	preactivate	O,-,-
validate	validate	O,-,-
getOperationProfile	get_bulk_cm_irp_operation_profile get_controlled_upload_bulk_cm_irp_operation_profile get_simple_upload_bulk_cm_irp_operation_profile	O,-,- -,O,- -,-,O
getNotificationProfile	get_bulk_cm_irp_notification_profile get_controlled_upload_bulk_cm_irp_notification_profile get_simple_upload_bulk_cm_irp_notification_profile	O,-,- -,O,- -,-,O

A.2.3 Operation parameter mapping

Reference Bulk CM IRP; Information Service [9] 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 A.2.3.1: Mapping from IS startSession parameters to SS equivalents

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

Table A.2.3.2: Mapping from IS endSession parameters to SS equivalents

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

Table A.2.3.3: Mapping from IS upload parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionId	BulkCmIRPConstDefs::SessionId session_id	M
uploadDataFileReference	BulkCmIRPConstDefs::FileDestination sink	M
baseObjectInstance	BulkCmIRPConstDefs::DistinguishedName base_object	M
scope, filter	BulkCmIRPConstDefs::SearchControl search_control	M
status	Exceptions: UploadException, UnknownSessionIdException, MaxSessionReachedException, NotValidInCurrentStateException, ConcurrencyException, IllegalDNFormatException, IllegalFilterFormatException, IllegalScopeTypeException, IllegalScopeLevelException, IllegalURLFormatException, 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 A.2.3.4: 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	Exceptions: DownloadException, UnknownSessionIdException, MaxSessionReachedException, NotValidInCurrentStateException, IllegalURLFormatException, 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 A.2.3.5: 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	M
status	Exceptions: ActivateException, UnknownSessionIdException, NotValidInCurrentStateException, ConcurrencyException, IllegalActivationModeException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table A.2.3.6: Mapping from IS fallback parameters to SS equivalents

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

Table A.2.3.7: Mapping from IS abortSessionOperation parameters to SS equivalents

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

Table A.2.3.8: Mapping from IS getSessionIds parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
sessionIdList	Return value of type BulkCmIRPConstDefs::SessionIdList	M
status	Exceptions: GetSessionIdsException, ManagedGenericIRPSystem::InvalidParameter	M

Table A.2.3.9: Mapping from IS getSessionStatus parameters to SS equivalents

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

Table A.2.3.10: Mapping from IS getSessionLog parameters to SS equivalents

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

IS Operation parameter	SS Method parameter	Qualifier
logFileReference	BulkCmIRPConstDefs::FileDestination sink	M
contentType	boolean only_error_info	M
status	Exceptions: GetSessionLogException, UnknownSessionIdException, IllegalURLFormatException, 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 A.2.3.11: Mapping from IS getBulkCmIRPVersion parameters to SS equivalents

IS Operation parameter	SS Method parameter	Qualifier
versionNumberList	Return value of type ManagedGenericIRPConstDefs::VersionNumberSet	M
status	Exceptions: GetBulkCmIRPVersionsException	M

Table A.2.3.12: 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	Exceptions: ValidateException , UnknownSessionIdException, NotValidInCurrentStateException, ConcurrencyException, IllegalActivationModeException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ManagedGenericIRPSystem::OperationNotSupported	M

Table A.2.3.13: 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	Exceptions: PreactivateException, UnknownSessionIdException, NotValidInCurrentStateException, ConcurrencyException, IllegalActivationModeException, IllegalVerificationModeException, ManagedGenericIRPSystem::ParameterNotSupported, ManagedGenericIRPSystem::InvalidParameter, ManagedGenericIRPSystem::OperationNotSupported	M

Table A.2.3.14: 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

IS Operation parameter	SS Method parameter	Qualifier
status	Exceptions: GetBulkCMIRPOperationProfileException, ManagedGenericIRPSystem::OperationNotSupported, ManagedGenericIRPSystem::InvalidParameter	M

Table A.2.3.15: 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

A.2.4 Notification parameter mapping

Reference 3GPP TS 32.612 [9] defines semantics of parameters carried in notifications. The following tables indicate the mapping of these parameters to their OMG CORBA Structured Event (defined in OMG Notification Service [26]) equivalents. The composition of OMG Structured Event, as defined in the OMG Notification Service [26], is:

```
Header
      Fixed Header
           domain_name
           type_name
           event_name
      Variable Header
Body
      filterable_body_fields
      remaining_body
```

The following tables list all OMG Structured Event attributes in the second column. The first column identifies the Bulk CM IRP: IS [9] defined notification parameters.

Table A.2.4.1: 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 3GPP TS 32.312 [6]. 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 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 [4]. These are attributes of Header defined in the IS.
notificationId	One NV pair of remaining_body	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.

IS Parameter	OMG CORBA Structured Event Attribute	Qualifier	Comment
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 remaining_body	M	Name of NV pair is the SESSION_ID of interface NotifyGetSessionLogEnded of module BulkCMIRPNotifications. Value of NV pair is a string.
sourceIndicator	One NV pair of remaining_body	O	Name of NV pair is the SOURCE_INDICATOR of interface NotifyGetSessionLogEnded of module BulkCMIRPNotifications. Value of NV pair is a string.
There is no corresponding IS attribute.	One NV pair of remaining_body	M	Name of NV pair is the ERROR_INFORMATION of interface NotifyGetSessionLogEnded of module BulkCMIRPNotifications. Value of NV pair is a string.

Table A.2.4.2: 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 3GPP TS 32.312 [6]. 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 [4]. These are attributes of Header defined in the IS.
notificationId	One NV pair of remaining_body	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.

IS Parameter	OMG CORBA Structured Event attribute	Qualifier	Comment
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 remaining_body	M	Name of NV pair is the SESSION_ID of interface NotifySessionStateChange of module BulkCMIRPNotifications. Value of NV pair is a string.
sourceIndicator	One NV pair of remaining_body	O	Name of NV pair is the SOURCE_INDICATOR of interface NotifySessionStateChange of module BulkCMIRPNotifications. Value of NV pair is a string.
There is no corresponding IS attribute.	One NV pair of remaining _body	M	Name of NV pair is the ERROR_INFORMATION of interface NotifySessionStateChange of module BulkCMIRPNotifications. Value of NV pair is a string.

A.2.5 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.

A.2.6 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 A.2.6: 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

A.2.7 Package Mapping

The Bulk CM IRP: IS (3GPP TS 32.612 [9]) clause 7 specifies packages of capabilities.
The IS-defined packages are mapped into IDL module constructs. Specifically:

- The operations named in the IS-defined packages Simple Upload, Controlled Upload and Controlled Upload & Provisioning are mapped to methods in SimpleUploadBulkCMIRSystem::SimpleUploadBulkCMIRP, ControlledUploadBulkCMIRPSystem::ControlledUploadBulkCMIRP and BulkCmIRPSystem::BulkCmIRP respectively (see clause A.3.3).

- The notifications named in the IS-defined Simple Upload, Controlled Upload and Controlled Upload & Provisioning are mapped to SS Interfaces defined in IDL module BulkCMIRPNotifications (see clause A.3.4).

A.3 Solution Set definitions

A.3.1 IDL definition structure

Clause A.3.2 defines the constants and types used by the Bulk CM IRP.

Clause A.3.3 defines the operations which are performed by the Bulk CM IRP agent.

Clause A.3.4 defines the notifications which are emitted by the Bulk CM IRP agent.

A.3.2 IDL specification "BulkCmIRPConstDefs.idl"

```
//File: BulkCmIRPConstDefs.idl
#ifndef _BULK_CM_IRP_CONST_DEFS_IDL_
#define _BULK_CM_IRP_CONST_DEFS_IDL_

// This statement must appear after all include statements
#pragma prefix "3gppsa5.org"

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

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

   /*
   This block defines all possible values for sessionState.
   One of these strings appear in the event_name of the
   Structured Event of notifySessionStateChanged notification.
   */
   interface SessionStateChangeNotification
   {
      const string UPLOAD_FAILED = "x1";
      const string UPLOAD_COMPLETED = "x2";
      const string DOWNLOAD_FAILED = "x3";
      const string DOWNLOAD_COMPLETED = "x4";
      const string ACTIVATION_FAILED = "x5";
      const string ACTIVATION_PARTLY_REALISED = "x6";
      const string ACTIVATION_COMPLETED = "x7";
      const string FALLBACK_FAILED = "x8";
      const string FALLBACK_PARTLY_REALISED = "x9";
      const string FALLBACK_COMPLETED = "x10";
      const string VALIDATION_FAILED = "x11";
      const string VALIDATION_COMPLETED = "x12";
      const string PREACTIVATION_FAILED = "x13";
      const string PREACTIVATION_PARTLY_REALISED = "x14";
      const string PREACTIVATION_COMPLETED = "x15";
   };

   /*
   This block defines all possible values for sessionLogStatus
   One of these strings appear in the event_name of the Structured
   Event of notifyGetSessionLogEnded notification.
   */
   interface LogStateNotification
   {
      const string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY = "x1";
      const string GET_SESSION_LOG_COMPLETED_UNSUCESSFULLY = "x2";
   };

   /*
   For each started configuration session a unique identifier is generated
   by the IRPManager. An sessionId can not be used for an upload if it is
   already in use of a download configuration and vice versa.
```

```
*/
typedef string SessionId;

/*
This string field is used in order to provide additional error information
if an operation has failed.
*/
typedef string ErrorInformation;

/*
Defines the different subphases of a configuration session
e.g. thus it is easy to implement a detection of an upload
or a download/activate session.
*/
enum SubPhase {IDLE_PHASE, DOWNLOAD_PHASE, UPLOAD_PHASE, ACTIVATION_PHASE,
               FALLBACK_PHASE, PREACTIVATION_PHASE, VALIDATION_PHASE};

/*
Defines the different substates of a configuration session. This includes
the transition state as well.
*/
enum SubState {COMPLETED, FAILED, PARTLY_REALISED, IN_PROGRESS};

/*
Defines state of a configuration session with the phase and the substate
of the configuration.
*/
struct SessionState
{
   SubPhase sub_phase;
   SubState sub_state;
};

/*
Contains the list of all current sessionIds
*/
typedef sequence <SessionId> SessionIdList;

/*
Specifies a complete destination path (including filename).
*/
typedef string FileDestination;

/*
The format of Distinguished Name is specified in
the Naming Conventions for Managed Objects; TS 32.300.
e.g. "SubNetwork=10001,ManagedElement=400001" identifies a
ManagedElement instance of the object model.
*/
typedef string DistinguishedName;

/*
Used within the upload method to give filter critera
*/
typedef string Filter;

/*
Defines the kind of scope to use in a search together with
SearchControl.level, in a SearchControl value.
SearchControl.level is always >= 0. If a level is bigger than the
depth of the tree there will be no exceptions thrown.
*/
enum ScopeType {BASE_ONLY, BASE_NTH_LEVEL, BASE_SUBTREE, BASE_ALL};

/*
Controls the searching for MOs during upload, and contains:
the type of scope ("type" field),
the level of scope ("level" field),
the filter ("filter_" field),
The type and level fields are mandatory.
The filter field is mandatory (The filter will have to be
set to an empty string if it has no other value).
*/
struct SearchControl
{
   ScopeType type;
   unsigned long level;
   Filter filter_;
```

```
   };

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

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

   /* 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 // _BULK_CM_IRP_CONST_DEFS_IDL_
```

A.3.3 IDL specification "BulkCMIRPSystem.idl"

```
//File: BulkCMIRPSystem.idl
#ifndef _BULK_CM_IRP_SYSTEM_IDL_
#define _BULK_CM_IRP_SYSTEM_IDL_

#include <BulkCmIRPConstDefs.idl>
#include <ManagedGenericIRPConstDefs.idl>
#include <ManagedGenericIRPSystem.idl>

// This statement must appear after all include statements
#pragma prefix "3gppsa5.org"

/* ## Module: BulkCmIRPSystem
This module implements capabilities of Bulk CM IRP.
================================================================
*/
module BulkCmIRPSystem
{
   /*
   The request cannot be processed due to a situation of concurrency.
   E.g. two concurrent activation requests involving the same ManagedElement
   instance. The semantics carried in reason is outside the scope of this IRP.
   */
   exception ConcurrencyException { string reason; };

   /*
   The provided filter is malformed or invalid. The semantics carried in reason
   is outside the scope of this IRP.
   */
   exception IllegalFilterFormatException { string reason; };

   /*
   The provided Distinguished Name is malformed or invalid. The semantics
   carried in reason is outside the scope of this IRP.
   */
   exception IllegalDNFormatException { string reason; };

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

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

   /*
   The request cannot be processed because no fallback data is available, i.e.
   fallback capability was previously not asked for.
   */
   exception NoFallbackException {};

   /*
   The provided sessionId value is already used for another configuration
   session. The semantics carried in reason is outside the scope of this IRP.
   */
   exception SessionIdInUseException { string reason; };

   /*
   The provided URL is malformed or invalid. The semantics carried in reason is
   outside the scope of this IRP.
   */
   exception IllegalURLFormatException{ string reason; };

   /*
   The provided sessionId value does not identify any existing configuration
   session.
   */
   exception UnknownSessionIdException {};

   /*
   The request cannot be processed because it is not valid in the current state
```

```
of the configuration session.
*/
exception NotValidInCurrentStateException
{
   BulkCmIRPConstDefs::SessionState current_state;
};

/*
The request cannot be processed because the maximum number of simultaneously
running configuration sessions has been reached. The semantics carried in
reason is outside the scope of this IRP.
*/
exception MaxSessionReachedException { string reason; };

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

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

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

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

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

   /*
   Return the list of all supported notifications and their supported
   parameters for a specific BulkCM IRP version.
   */
   ManagedGenericIRPConstDefs::MethodList
      get_bulk_cm_irp_notification_profile
   (
```

```
   in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
)
raises (GetBulkCMIRPNotificationProfileException,
        ManagedGenericIRPSystem::OperationNotSupported,
        ManagedGenericIRPSystem::InvalidParameter);

/*
Uploads a configuration from the subnetwork. The result is put in a
configuration data file in an area specified by the IRPManager.
The MIB of the subnetwork is iterated by means of containment search,
using a SearchControl to control the search and the returned results.
All MOs in the scope constitutes a set that the filter works on.
In case of a concurrent running session the function will
return an exception. If the value of the given baseObject or Filter
does not exist then this asynchronous error condition will be notified.
*/
void upload (
   in BulkCmIRPConstDefs::SessionId session_id,
   in BulkCmIRPConstDefs::FileDestination sink,
   in BulkCmIRPConstDefs::DistinguishedName base_object,
   in BulkCmIRPConstDefs::SearchControl search_control
)
raises (UploadException, UnknownSessionIdException,
        MaxSessionReachedException, NotValidInCurrentStateException,
        ConcurrencyException,
        IllegalDNFormatException, IllegalFilterFormatException,
        IllegalScopeTypeException, IllegalScopeLevelException,
        IllegalURLFormatException,
        ManagedGenericIRPSystem::InvalidParameter);

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

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

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

/*
Creates an instance of the configuration session state machine. The
```

```
IDLE_PHASE & COMPLETED is notified
*/
void start_session (
   in BulkCmIRPConstDefs::SessionId session_id
)
raises (StartSessionException, SessionIdInUseException,
        MaxSessionReachedException,
        ManagedGenericIRPSystem::InvalidParameter);

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

/*
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);

   };

};

module SimpleUploadBulkCMIRPSystem
{
   exception GetSimpleUploadBulkCmIRPVersionsException { string reason; };
   exception GetSimpleUploadBulkCMIRPOperationProfileException
      { string reason; };
   exception GetSimpleUploadBulkCMIRPNotificationProfileException
      { string reason; };

   interface SimpleUploadBulkCMIRP
   {
      /*
      Return the list of all supported Bulk CM IRP versions.
      */
      ManagedGenericIRPConstDefs::VersionNumberSet
         get_simple_upload_bulk_cm_irp_versions (
      )
      raises (GetSimpleUploadBulkCmIRPVersionsException);

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

      /*
      Return the list of all supported notifications and their supported
      parameters for a specific BulkCM IRP version.
      */
      ManagedGenericIRPConstDefs::MethodList
         get_simple_upload_bulk_cm_irp_notification_profile
      (
         in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
      )
      raises (GetSimpleUploadBulkCMIRPNotificationProfileException,
              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 Filter
      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 (
         BulkCmIRPSystem::UploadException,
         BulkCmIRPSystem::UnknownSessionIdException,
         BulkCmIRPSystem::MaxSessionReachedException,
         BulkCmIRPSystem::NotValidInCurrentStateException,
         BulkCmIRPSystem::ConcurrencyException,
         BulkCmIRPSystem::IllegalDNFormatException,
         BulkCmIRPSystem::IllegalFilterFormatException,
         BulkCmIRPSystem::IllegalScopeTypeException,
         BulkCmIRPSystem::IllegalScopeLevelException,
         BulkCmIRPSystem::IllegalURLFormatException,
         ManagedGenericIRPSystem::InvalidParameter);
   };

};  // end of module SimpleUploadBulkCMIRPSystem

module ControlledUploadBulkCMIRPSystem
{
   exception GetControlledUploadBulkCmIRPVersionsException { string reason; };
   exception GetControlledUploadBulkCMIRPOperationProfileException
      { string reason; };
   exception GetControlledUploadBulkCMIRPNotificationProfileException
      { string reason; };

   interface ControlledUploadBulkCMIRP
   {
      /*
      Return the list of all supported Bulk CM IRP versions.
      */
      ManagedGenericIRPConstDefs::VersionNumberSet
         get_controlled_upload_bulk_cm_irp_versions (
      )
      raises (GetControlledUploadBulkCmIRPVersionsException);

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

      /*
      Return the list of all supported notifications and their supported
      parameters for a specific BulkCM IRP version.
      */
      ManagedGenericIRPConstDefs::MethodList
         get_controlled_upload_bulk_cm_irp_notification_profile (
             in ManagedGenericIRPConstDefs::VersionNumber bulk_cm_irp_version
      )
      raises (GetControlledUploadBulkCMIRPNotificationProfileException,
               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 Filter
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 (
   BulkCmIRPSystem::UploadException,
   BulkCmIRPSystem::UnknownSessionIdException,
   BulkCmIRPSystem::MaxSessionReachedException,
   BulkCmIRPSystem::NotValidInCurrentStateException,
   BulkCmIRPSystem::ConcurrencyException,
   BulkCmIRPSystem::IllegalDNFormatException,
   BulkCmIRPSystem::IllegalFilterFormatException,
   BulkCmIRPSystem::IllegalScopeTypeException,
   BulkCmIRPSystem::IllegalScopeLevelException,
   BulkCmIRPSystem::IllegalURLFormatException,
   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 (
   BulkCmIRPSystem::GetSessionLogException,
   BulkCmIRPSystem::UnknownSessionIdException,
   BulkCmIRPSystem::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 (
   BulkCmIRPSystem::StartSessionException,
   BulkCmIRPSystem::SessionIdInUseException,
   BulkCmIRPSystem::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 (
   BulkCmIRPSystem::GetSessionStatusException,
   BulkCmIRPSystem::UnknownSessionIdException,
   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 (
         BulkCmIRPSystem::EndSessionException,
         BulkCmIRPSystem::UnknownSessionIdException,
         BulkCmIRPSystem::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 (
         BulkCmIRPSystem::AbortSessionOperationException,
         BulkCmIRPSystem::UnknownSessionIdException,
         BulkCmIRPSystem::NotValidInCurrentStateException,
         ManagedGenericIRPSystem::InvalidParameter);

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

     };

};  // end of module ControlledUploadBulkCMIRPSystem

#endif // _BULK_CM_IRP_SYSTEM_IDL_
```

A.3.4 IDL specification "BulkCMIRPNotifications.idl"

```
//File: BulkCMNotifications.idl
#ifndef _BULK_CM_IRP_NOTIFICATIONS_IDL_
#define _BULK_CM_IRP_NOTIFICATIONS_IDL_

#include <NotificationIRPNotifications.idl>
#include <BulkCmIRPConstDefs.idl>

// This statement must appear after all include statements
#pragma prefix "3gppsa5.org"

module BulkCMIRPNotifications
{

   interface NotifySessionStateChange: NotificationIRPNotifications::Notify
   {
      // This is the type_name (2nd field) of the fixed header.
      const string EVENT_TYPE =
         BulkCmIRPConstDefs::NotificationType::NOTIFY_SESSION_STATE_CHANGED;

      // ----------------------------
      // One of the strings here is the event_name (3rd field) of the
      // fixed header.
      // The first 2 are relevant for IS-defined packages Simple
      // Upload and Controlled Upload.
      // All are relevant for IS-defined package
      // Controlled Upload & Provisioning.

      const string UPLOAD_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::UPLOAD_FAILED;
      const string UPLOAD_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::UPLOAD_COMPLETED;
      const string DOWNLOAD_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::DOWNLOAD_FAILED;
      const string DOWNLOAD_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::DOWNLOAD_COMPLETED;
      const string ACTIVATION_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::ACTIVATION_FAILED;
      const string ACTIVATION_PARTLY_REALISED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::ACTIVATION_PARTLY_REALISED;
      const string ACTIVATION_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::ACTIVATION_COMPLETED;
      const string FALLBACK_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::FALLBACK_FAILED;
      const string FALLBACK_PARTLY_REALISED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::FALLBACK_PARTLY_REALISED;
      const string FALLBACK_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::FALLBACK_COMPLETED;
      const string VALIDATION_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::VALIDATION_FAILED;
      const string VALIDATION_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::VALIDATION_COMPLETED;
      const string PREACTIVATION_FAILED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::PREACTIVATION_FAILED;
      const string PREACTIVATION_PARTLY_REALISED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::PREACTIVATION_PARTLY_REALISED;
      const string PREACTIVATION_COMPLETED = BulkCmIRPConstDefs::
         SessionStateChangeNotification::PREACTIVATION_COMPLETED;
      // ----------------------------

      const string SESSION_ID =
         BulkCmIRPConstDefs::AttributeNameValue::SESSION_ID;

      const string SOURCE_INDICATOR =
         BulkCmIRPConstDefs::AttributeNameValue::SOURCE_INDICATOR;
   };

   interface NotifyGetSessionLogEnded: NotificationIRPNotifications::Notify
   {
      // This is the type_name (2nd field) of the fixed header.
      const string EVENT_TYPE =
         BulkCmIRPConstDefs::NotificationType::NOTIFY_GET_SESSION_LOG_ENDED;

      // -----------------
      // One of the 2 strings here is the event_name (3rd field) of the
```

```
      // fixed header.
      const string GET_SESSION_LOG_COMPLETED_SUCCESSFULLY =
         BulkCmIRPConstDefs::LogStateNotification::
            GET_SESSION_LOG_COMPLETED_SUCCESSFULLY;
      const string GET_SESSION_LOG_COMPLETED_UNSUCESSFULLY =
         BulkCmIRPConstDefs::LogStateNotification::
            GET_SESSION_LOG_COMPLETED_UNSUCESSFULLY;
      // ----------------------

      const string SESSION_ID =
         BulkCmIRPConstDefs::AttributeNameValue::SESSION_ID;

      const string SOURCE_INDICATOR =
         BulkCmIRPConstDefs::AttributeNameValue::SOURCE_INDICATOR;
   };

};

#endif // _BULK_CM_IRP_NOTIFICATIONS_IDL_
```

Annex B (normative): XML Definitions

This annex contains the XML Definitions for the Bulk CM Integration Reference Point.

B.1 Architectural Features

The overall architectural feature of Bulk CM IRP is specified in 3G TS 32.612 [9]. This clause specifies features that are specific to the XML definitions.

The present document provides the main part of the XML file format definition for the Bulk Configuration Management IRP IS in 3GPP TS 32.612 [9].

The other parts of this XML definition are NRM-specific parts. All NRM IRPs that include SS-level XML definition are in the scope of the Bulk CM IRP.

Bulk CM XML file formats are based on XML [27], XML Schema [28] [29] [30] and XML Namespace [31] standards.

B.1.1 Syntax for Distinguished Names

The syntax of a Distinguished Name is defined in 3GPP TS 32.300 [4].

B.2 Structure and content of configuration data XML files

The present clause defines the file format of configuration data XML files exchanged between an IRPManager and an IRPAgent as part of `upload` and `download` operations of the Bulk CM IRP IS (see [9]).

Upload and download configuration data XML files share a common file format defined by the XML schema in clause B.4.3 and by the following clauses.

Additionally, vendor-specific XML schemas shall be provided to enable configuration data XML files to carry vendor-specific data (see clause B.2.6).

The use of XML schemas enables to ensure configuration data XML files have the proper structure and to some extent the proper content, and in particular to ensure:

- for a given NRM instance, it is properly named/positioned with regard to the global NRM naming tree;
- for a given NRM instance, only attributes of the corresponding NRM class are present;
- for a given NRM attribute, its value is of the proper type.

Location of the XML schemas used for configuration data XML files is outside the scope of this document.

B.2.1 Global structure

The content of a configuration data XML file is the succession of:

- the standard XML declaration with specification of the version of XML and of the character encoding being used (see [27]);
- a `bulkCmConfigDataFile` XML element; this is the root XML element of configuration data XML files.

The definition of the allowed character encoding(s) is outside the scope of this document.

As defined by the following extract of XML schema `configData.xsd` (see clause B.4.3):

```
  <element name="bulkCmConfigDataFile">
    <complexType>
      <sequence>
        <element name="fileHeader">
[…]
        </element>
        <element name="configData" maxOccurs="unbounded">
[…]
        </element>
        <element name="fileFooter">
[…]
        </element>
      </sequence>
    </complexType>
  </element>
```

the XML content of a `bulkCmConfigDataFile` XML element is the succession of:

- a `fileHeader` XML element (see clause B.2.2);
- one or several `configData` XML elements (see clause B.2.3);
- a `fileFooter` XML element (see clause B.2.2).

XML elements `fileHeader` and `fileFooter` are empty XML elements (see clause B.2.2).

The `bulkCmConfigDataFile` XML element shall also have all the XML attribute specifications that declare the XML namespaces (see [31]) used in the XML file.

The following XML namespaces are potentially used in configuration data XML files:

- the default XML namespace is associated with the configuration data files base XML schema `configData.xsd` (see clause B.4.3);
- for each NRM-specific XML schema, a specific XML namespace prefix is defined for the associated XML namespace (see clause B.2.4.1);
- XML namespaces prefixes starting with `vs`, e.g. `vsRHO11`, are reserved for the XML namespaces associated with the vendor-specific XML schemas (see clause B.2.6).

Each `configData` XML element (see clause B.2.3) carries:

- NRM instances with or without their NRM attribute values in a NRM naming tree organized structure together with `modifier` XML attribute specification (see clause B.2.5);
- possibly vendor-specific data (see clause B.2.6).

A `configData` XML element can carry an entire tree of NRM instances with their NRM attribute values and the related vendor-specific data or any subset of it.

The following is an example of a configuration data XML file, without presentation of the XML attribute specifications and XML content of `fileHeader`, `configData` and `fileFooter` XML elements (replaced by […]; see clauses B.2.2, B.2.3, B.2.5 and B.2.6):

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
[…]
>
  <fileHeader […]/>
  <configData […]>
[…]
  </configData>
  <configData […]>
[…]
  </configData>
  <fileFooter […]/>
</bulkCmConfigDataFile>
```

B.2.2 XML elements `fileHeader` and `fileFooter`

B.2.2.1 XML element `fileHeader`

As defined by the following extract of XML schema `configData.xsd` (see clause B.4.3):

```
<element name="fileHeader">
  <complexType>
    <attribute name="fileFormatVersion" type="string" use="required"/>
    <attribute name="senderName" type="string" use="optional"/>
    <attribute name="vendorName" type="string" use="optional"/>
  </complexType>
</element>
```

a `fileHeader` XML element:

- has the following XML attribute specifications:
  - a `fileFormatVersion` XML attribute specification; this attribute specification carries the abridged number and version of this 3GPP document (see below); this identifies the version of the file format used for assembling the XML file;
  - a conditional `senderName` XML attribute specification; this attribute specification shall be present only in XML files generated by the IRPAgent; it carries the DN of the IRPAgent that assembled the XML file, i.e. the value of the `systemDN` NRM attribute of the `IRPAgent` NRM instance (see [10]);
  - a conditional `vendorName` XML attribute specification; this attribute specification shall be present only in XML files generated by the IRPAgent; it carries the name of the vendor of the IRPAgent that assembled the XML file;
- and has an empty XML content.

The abridged number and version of a 3GPP document is constructed from its version specific full reference "3GPP […] (yyyy-mm)" by:

- removing the leading "3GPP TS";
- removing everything including and after the version third digit, representing editorial only changes, together with its preceding dot character;
- from the resulting string, removing leading and trailing white space, replacing every multi character white space by a single space character and changing the case of all characters to uppercase.

The following is an example of a `fileHeader` XML element:

```
<fileHeader
  fileFormatVersion="32.616 V10.0"
  senderName="DC=a1.companyNN.com,SubNetwork=1,IRPAgent=1"
  vendorName="Company NN"
/>
```

B.2.2.2 XML element `fileFooter`

As defined by the following extract of XML schema `configData.xsd` (see clause B.4.3):

```
<element name="fileFooter">
  <complexType>
    <attribute name="dateTime" type="dateTime" use="required"/>
  </complexType>
</element>
```

a `fileFooter` XML element:

- has a `dateTime` XML attribute specification; this attribute specification carries the date and time the XML file was assembled;

- and has an empty XML content.

The following is an example of a `fileFooter` XML element:

```
<fileFooter dateTime="2001-05-07T12:00:00+02:00"/>
```

B.2.3 XML element `configData`

As defined by the following extract of XML schema `configData.xsd` (see clause B.4.3):

```
<element name="configData" maxOccurs="unbounded">
  <complexType>
    <choice>
      <element ref="xn:SubNetwork"/>
      <element ref="xn:MeContext"/>
      <element ref="xn:ManagedElement"/>
    </choice>
    <attribute name="dnPrefix" type="string" use="optional"/>
  </complexType>
</element>
```

a `configData` XML element:

- has an optional `dnPrefix` XML attribute specification; this attribute specification carries the DN Prefix information as defined in Annex C of 3GPP TS 32.300 [4];
- and its XML content is an instance of the specific type of XML element (see below) corresponding to one of the NRM classes SubNetwork, MeContext or ManagedElement (see [10]); depending on the System Context of the IRP (see [9]) the used NRM class shall be:
  - in case of System Context A, only SubNetwork NRM class, or;
  - in case of System Context B, only MeContext or ManagedElement NRM class.

This instance of SubNetwork/MeContext/ManagedElement NRM class corresponding specific XML element type is the starting point for a `configData` XML element to possibly contain several NRM instances in a NRM naming tree organized structure (see clause B.2.4.2).

The following is an example of a `configData` XML element:

```
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork […]>
[…]
    </xn:SubNetwork>
  </configData>
```

B.2.4 NRM-specific XML elements

NRM-specific XML element types are generically defined under the mapping rules defined in clause B.2.4.2.

NRM-specific XML element types are explicitly declared by NRM-specific XML schemas as defined in clause B.2.4.1.

B.2.4.1 NRM-specific XML schemas

NRM-specific XML schemas are defined in the NRM-specific parts (see clause 1) of the XML file format definition for the Bulk Configuration Management IRP IS [9].

NRM-specific XML schemas with definition of corresponding XML namespace prefixes (see clause B.2.1) are listed by the following table:

Table B.2.4.1: NRM-specific XML schemas, corresponding 3GPP TSs and XML namespace prefixes

NRM	XML schema	3GPP TS no.	XML namespace prefix
Core Network Resources	coreNrm.xsd	32.636 [13]	cn
Generic Network Resources	genericNrm.xsd	32.626 [11]	xn
GERAN Network Resources	geranNrm.xsd	32.656 [17]	gn
IM Network Resources	inventoryNrm.xsd	32.696 [19]	in
IMS NRM	imsNrm.xsd	32.736 [21]	im
STN Network Resources	stnNrm.xsd	32.746 [23]	stn
TN Network Resources	transportNrm.xsd	32.716 [20]	tn
UTRAN Network Resources	utranNrm.xsd	32.646 [15]	un
E-UTRAN Network Resources	eutranNrm.xsd	32.766 [25]	en
EPC Network Resources	epcNrm.xsd	32.756 [24]	epc
SON Policy Network Resources	sonPolicyNrm.xsd	32.526 [39]	sp
Subscription Management Network Resources	sumNrm.xsd	32.176 [37]	sn
Repeater Network Resources	RepeaterNrm.xsd	32.726 [42]	rn
HNS Network Resources	hnsNrm.xsd	32.776 [47]	hn
HeNS Network Resources	hensNrm.xsd	32.786 [49]	hen
Generic RAN Network Resources	genericRanNrm.xsd	32.796 [50]	gr

Each NRM-specific XML schema explicitly declares NRM-specific XML element types for the related NRM.

Additionally, XML schema `genericNrm.xsd` (see [11]) also provides global XML declarations and definitions for the support of:

- NRM-specific XML element type declaration;
- vendor-specific XML element type declaration (see clause B.2.6).

B.2.4.2 Generic mapping rules

NRM-specific XML element types are generically defined under the following mapping rules:

- to each NRM class corresponds a specific type of XML element having the following characteristics:
  - its name is the name of the NRM class;
  - it derives by extension (see [28], [29] and [30]) the `NrmClass` XML complex type defined in the XML schema `genericNrm.xsd` (see [11]);
  - it has the following XML attribute specifications, inherited from `NrmClass` XML complex type:

- an `id` XML attribute specification; this attribute specification carries the attribute value part of the RDN of the NRM instance carried by the XML element, i.e. the value of the naming attribute of this NRM instance;
- an optional `modifier` XML attribute specification (see clause B.2.5);

- and its XML content is the succession of:
  - an optional `attributes` XML element whose XML content is the succession of:
    - zero or more specific XML elements (see below) corresponding to attributes of the NRM class, each occurring not more than once;
  - zero or more similar specific XML elements corresponding to direct subordinate NRM classes of the NRM class to which the current XML element corresponds;

- to each NRM attribute of each NRM class, except for the following NRM attributes:
  - the naming NRM attribute of each NRM class, whose value is already carried by the `id` XML attribute specification of the specific XML element corresponding to the NRM class;
  - the conditional `dnPrefix` NRM attribute of `SubNetwork`, `MeContext` and `ManagedElement` NRM classes (see [10]), whose value is already carried by the `dnPrefix` XML attribute specification of the `configData` XML element;

  corresponds a specific type of XML element having the following characteristics:
  - its name is constructed from the name of the NRM attribute by removing any contained dash character;
  - and it has an XML content; this XML content carries the value of the NRM attribute.

For example for the `SubNetwork` NRM class (see [10]), the corresponding extract of XML schema `genericNrm.xsd` (see [11]) is the following:

```
  <element name="SubNetwork">
    <complexType>
      <complexContent>
        <extension base="xn:NrmClass">
          <sequence>
            <element name="attributes" minOccurs="0">
              <complexType>
                <all>
                  <element name="userLabel" minOccurs="0"/>
                  <element name="userDefinedNetworkType" minOccurs="0"/>
                </all>
              </complexType>
            </element>
            <choice minOccurs="0" maxOccurs="unbounded">
              <element ref="xn:SubNetwork"/>
              <element ref="xn:ManagedElement"/>
              <element ref="xn:MeContext"/>
              <element ref="xn:ManagementNode"/>
              <element ref="xn:IRPAgent"/>
              <element ref="xn:SubNetworkOptionallyContainedNrmClass"/>
            </choice>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>
```

supported by the following extract of XML schema `genericNrm.xsd` (see [11]):

```
  <complexType name="NrmClass">
    <attribute name="id" type="string" use="required"/>
    <attribute name="modifier" use="optional">
[…]
    </attribute>
  </complexType>
```

Exceptions to the generic mapping rules for the definition of NRM-specific XML element types are listed by the following table:

Table B.2.4.2: Generic mapping rule exceptions

NRM classes / attributes	NRM 3GPP TS no.	Exception description references
vsData attribute of VsDataContainer class	32.622 [10]	clause B.2.6 of the present document and annex A of 3GPP TS 32.626 [11]

The following is an example of a `configData` XML element with regard to NRM-specific XML elements (in **bold**) in a configuration data XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
[…]
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1">
      <xn:attributes>
        <xn:userLabel>Paris SN1</xn:userLabel>
        <xn:userDefinedNetworkType>UMTS</xn:userDefinedNetworkType>
      </xn:attributes>
      <xn:ManagementNode id="1">
        <xn:attributes>
          <xn:userLabel>Paris MN1</xn:userLabel>
          <xn:vendorName>Company NN</xn:vendorName>
          <xn:userDefinedState>commercial</xn:userDefinedState>
          <xn:locationName>Montparnasse</xn:locationName>
        </xn:attributes>
      </xn:ManagementNode>
      <xn:ManagedElement id="1">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
          <xn:userLabel>Paris RN1</xn:userLabel>
          <xn:vendorName>Company NN</xn:vendorName>
          <xn:userDefinedState>commercial</xn:userDefinedState>
          <xn:locationName>Champ de Mars</xn:locationName>
        </xn:attributes>
      </xn:ManagedElement>
      <xn:ManagedElement id="2">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
          <xn:userLabel>Paris RN2</xn:userLabel>
          <xn:vendorName>Company NN</xn:vendorName>
          <xn:userDefinedState>commercial</xn:userDefinedState>
          <xn:locationName>Concorde</xn:locationName>
        </xn:attributes>
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

B.2.5 XML attribute specification `modifier`

As defined by the following extract of XML schema `genericNrm.xsd` (see [11]):

```
<attribute name="modifier" use="optional">
  <simpleType>
    <restriction base="string">
      <enumeration value="create"/>
      <enumeration value="delete"/>
      <enumeration value="update"/>
    </restriction>
  </simpleType>
</attribute>
```

the value of the optional `modifier` XML attribute specification of the specific XML elements corresponding to the classes of the NRM is one of the following: `create`, `delete`, or `update`.

The semantic carried by a `modifier` XML attribute specification applies only to the NRM instance corresponding to the containing XML element and not to any explicit or implicit subordinate NRM instances of this NRM instance.

The following rules apply for the `modifier` XML attribute specification:

- in upload XML configuration files, no `modifier` XML attribute specification should be present; on the contrary those are to be considered as meaningless and shall be ignored;
- in download XML configuration files:
  - if an XML element carrying an NRM instance has a `modifier` XML attribute specification of value `create`, then all directly or indirectly contained XML element carrying NRM instances, if any, shall also have a `modifier` XML attribute specification of value `create`;
  - if an XML element carrying an NRM instance has a `modifier` XML attribute specification of value `delete`, then all directly or indirectly contained XML element carrying NRM instances, if any, shall also have a `modifier` XML attribute specification of value `delete`;
  - if an XML element carrying an NRM instance has a `modifier` XML attribute specification of value `update`, then all directly contained XML element carrying NRM instances, if any, may also have a `modifier` XML attribute specification, this one being of either value `create`, `delete`, or `update`;
  - if an XML element carrying an NRM instance has no `modifier` XML attribute specification or a `modifier` XML attribute specification of value `delete`, then it shall not directly contain an `attributes` XML element.

A tree of XML elements corresponding to a tree of NRM instances with all XML elements having a `modifier` XML attribute specification of value `create` is considered to be in accordance with the following rule from Bulk CM IRP IS 3GPP TS 32.612 [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."

In such a tree of NRM instances, the XML element carrying a given NRM instance does not accurately appear before XML elements carrying subordinate NRM instances. The latter XML elements rather appear as the last part of the XML content of the former XML element.

Nevertheless, XML parsing of such a tree of NRM instances can still enable the above Bulk CM IRP IS rule to be fully respected. Example of an XML parsing enabling such compliance is one effectively actioning the creation of each NRM instance when having parsed the XML start-tag of the XML element carrying the NRM instance and, if any, the contained `attributes` XML element.

A tree of XML elements corresponding to a tree of NRM instances with all XML elements having a `modifier` XML attribute specification of value `delete` is considered to be in accordance with the following rule from Bulk CM IRP IS 3GPP TS 32.612 [9]:

> "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."

In such a tree of NRM instances, the XML elements carrying subordinate NRM instances do not appear before the XML element carrying the parent NRM instance. The former XML elements rather appear as the XML content of the latter XML element.

Nevertheless, XML parsing of such a tree of NRM instances can still enable the above Bulk CM IRP IS rule to be fully respected. Example of an XML parsing enabling such compliance is one effectively actioning the delete of each NRM instance when parsing the XML end-tag of the XML element carrying the NRM instance.

The following are examples of legal `configData` XML element with regard to `modifier` XML attribute specification (in **bold**) in configuration data XML files:

- example 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
[…]
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1" modifier="create">
      <xn:attributes>
        <xn:userLabel>Paris SN1</xn:userLabel>
        <xn:userDefinedNetworkType>UMTS</xn:userDefinedNetworkType>
      </xn:attributes>
      <xn:ManagementNode id="1" modifier="create">
        <xn:attributes>
          <xn:userLabel>Paris MN1</xn:userLabel>
[…]
          <xn:locationName>Montparnasse</xn:locationName>
        </xn:attributes>
      </xn:ManagementNode>
      <xn:ManagedElement id="1" modifier="create">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
[…]
          <xn:locationName>Champ de Mars</xn:locationName>
        </xn:attributes>
      </xn:ManagedElement>
      <xn:ManagedElement id="2" modifier="create">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
[…]
          <xn:locationName>Concorde</xn:locationName>
        </xn:attributes>
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

- example 2:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
[…]
```

```
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1">
      <xn:ManagedElement id="1" modifier="create">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
[…]
          <xn:locationName>Champ de Mars</xn:locationName>
        </xn:attributes>
      </xn:ManagedElement>
      <xn:ManagedElement id="2" modifier="create">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
[…]
          <xn:locationName>Concorde</xn:locationName>
        </xn:attributes>
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

- example 3:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
[…]
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1" modifier="delete">
      <xn:ManagementNode id="1" modifier="delete">
      </xn:ManagementNode>
      <xn:ManagedElement id="1" modifier="delete">
      </xn:ManagedElement>
      <xn:ManagedElement id="2" modifier="delete">
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

- example 4:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
[…]
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1">
      <xn:ManagedElement id="1" modifier="delete">
      </xn:ManagedElement>
      <xn:ManagedElement id="2" modifier="delete">
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

- example 5:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
```

```
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
  xmlns:un=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.646#utranNrm"
[…]
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1" modifier="update">
      <xn:attributes>
        <xn:userLabel>Paris SN1</xn:userLabel>
      </xn:attributes>
      <xn:ManagementNode id="1" modifier="update">
        <xn:attributes>
          <xn:userLabel>Paris MN1</xn:userLabel>
        </xn:attributes>
      </xn:ManagementNode>
      <xn:ManagedElement id="1" modifier="delete">
        <un:RncFunction id="1" modifier="delete">
        </un:RncFunction>
      </xn:ManagedElement>
      <xn:ManagedElement id="2" modifier="create">
        <xn:attributes>
          <xn:managedElementType>RNC</xn:managedElementType>
[…]
          <xn:locationName>Concorde</xn:locationName>
        </xn:attributes>
        <un:RncFunction id="2" modifier="create">
          <un:attributes>
            <un:userLabel>Paris RF2</un:userLabel>
[…]
            <un:rncId>2</un:rncId>
          </un:attributes>
        </un:RncFunction>
      </xn:ManagedElement>
      <xn:ManagedElement id="3">
        <un:RncFunction id="3" modifier="update">
          <un:attributes>
            <un:userLabel>Paris RF3</un:userLabel>
          </un:attributes>
        </un:RncFunction>
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

B.2.6 XML elements VsDataContainer, vsData and vsDataFormatVersion

As all XML element types corresponding to NRM classes (see clause B.2.4.2), the `VsDataContainer` XML element type, explicitly declared in 3GPP TS 32.626 [11], corresponds to the `VsDataContainer` NRM class defined in 3GPP TS 32.622 [10].

Contained in an `attributes` XML element type, itself contained in a `VsDataContainer` XML element, as all XML element types corresponding to NRM attributes (see clause B.2.4.2), the `vsData` and `vsDataFormatVersion` XML element types, explicitly declared in 3GPP TS 32.626 [11], correspond to the `vsData` and `vsDataFormatVersion` NRM attributes defined in 3GPP TS 32.622 [10].

As an exception to the generic mapping rules for the definition of NRM-specific XML element types (see clause B.2.4.2), the `vsData` XML element type has an empty XML content.

Each vendor-specific XML schema shall declare one ore more vendor-specific XML element types that:

- have a name starting with `vsData`, e.g. `vsDataRHO`;
- derive by extension (see [28], [29] and [30]) the `vsData` XML element type declared by the XML schema `genericNrm.xsd` (see [11]);
- are designated as members of the substitution group (see [28], [29] and [30]) headed by the `vsData` XML element type.

Beyond the above statement, the definition of vendor-specific XML schemas is outside the scope of this document.

The XML content of those vendor-specific XML elements carry vendor-specific data.

The XML content of the `vsDataFormatVersion` XML element shall be the filename, without the ".xsd" file extension and without any path specification, of the vendor-specific XML schema used for the related `VsDataContainer` XML element.

See clause B.4.4 for an example of a vendor-specific XML schema.

The following is an example of a vendor-specific XML element (in **bold**) deriving and extending the `vsData` XML element in a configuration data XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmConfigDataFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
  xmlns:un=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.646#utranNrm"
  xmlns:vsRHO11="http://www.companyNN.com/xmlschemas/NNRncHandOver.1.1"
[…]
>
[…]
  <configData dnPrefix="DC=a1.companyNN.com">
    <xn:SubNetwork id="1">
      <xn:ManagedElement id="1">
        <un:RncFunction id="1">
          <xn:VsDataContainer id="1">
            <xn:attributes>
              <xn:vsDataType>RncHandOver</xn:vsDataType>
              <xn:vsDataFormatVersion>NNRncHandOver.1.1</xn:vsDataFormatVersion>
              <vsRHO11:vsDataRHO>
                <vsRHO11:abcMin>12</vsRHO11:abcMin>
                <vsRHO11:abcMax>34</vsRHO11:abcMax>
              </vsRHO11:vsDataRHO>
            </xn:attributes>
          </xn:VsDataContainer>
        </un:RncFunction>
      </xn:ManagedElement>
    </xn:SubNetwork>
  </configData>
[…]
</bulkCmConfigDataFile>
```

B.3 Structure and content of session log XML files

The present clause defines the file format of session log XML files exchanged between an IRPManager and an IRPAgent as part of `getSessionLog` operation of the Bulk CM IRP IS (see [9]).

This file format is defined by the XML schema in clause B.4.5 and by the following clauses.

The use of an XML schema enables to ensure session log XML files have the proper structure and to some extent the proper content.

Location of the XML schemas used for session log XML files is outside the scope of this document.

B.3.1 Global structure

The content of a session log XML file is the succession of:

- the standard XML declaration with specification of the version of XML and of the character encoding being used (see [27]);
- a `bulkCmSessionLogFile` XML element; this is the root XML element of session log XML files.

The definition of the allowed character encoding(s) is outside the scope of this document.

As defined by the following extract of XML schema `sessionLog.xsd` (see clause B.4.5):

```
  <element name "bulkCmSessionLogFile">
    <complexType>
      <sequence>
        <element name="fileHeader">
[…]
        </element>
        <element name="activity" maxOccurs="unbounded">
[…]
        </element>
        <element name="fileFooter">
[…]
        </element>
      </sequence>
    </complexType>
  </element>
```

the XML content of a `bulkCmSessionLogFile` XML element is the succession of:

- a `fileHeader` XML element (see clause B.3.2);
- one or several `activity` XML elements (see clause B.3.3);
- a `fileFooter` XML element (see clause B.3.2).

XML elements `fileHeader` and `fileFooter` are empty XML elements (see clause B.3.2).

The `bulkCmSessionLogFile` XML element shall also have all the XML attribute specifications that declare the XML namespaces (see [31]) used in the XML file.

Only the default XML namespace is used in session log XML files. It is associated with the session log file XML schema `sessionLog.xsd` (see clause B.4.5).

The following is an example of a session log XML file, without presentation of the XML attribute specifications and XML content of `fileHeader`, `activity` and `fileFooter` XML elements (replaced by […]; see clauses B.3.2 and B.3.3):

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmSessionLogFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#sessionLog"
```

```
[…]
>
  <fileHeader […]/>
  <activity […]>
[…]
  </activity>
  <activity […]>
[…]
  </activity>
  <fileFooter […]/>
</bulkCmSessionLogFile>
```

B.3.2 XML elements `fileHeader` and `fileFooter`

The XML elements `fileHeader` and `fileFooter` for session log XML files have the same definition, structure and content as the XML elements `fileHeader` and `fileFooter` for configuration data XML files (see clause B.2.2).

B.3.3 XML element `activity`

As defined by the following extract of XML schema `sessionLog.xsd` (see clause B.4.5):

```
        <element name="activity" minOccurs="0" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="log" maxOccurs="unbounded">
[…]
              </element>
            </sequence>
            <attribute name="dateTime" type="dateTime" use="required"/>
            <attribute name="type" use="required">
              <simpleType>
                <restriction base="string">
                  <enumeration value="upload"/>
                  <enumeration value="download"/>
                  <enumeration value="validate"/>
                  <enumeration value="preactivate"/>
                  <enumeration value="activate"/>
                  <enumeration value="fallback"/>
                </restriction>
              </simpleType>
            </attribute>
          </complexType>
        </element>
```

an `activity` XML element:

- has the following XML attribute specifications:
  - a `dateTime` XML attribute specification; this attribute specification carries the date and time the Bulk CM activity was started;
  - a `type` XML attribute specification; this attribute specification carries the type of the Bulk CM activity triggered by the IRPManager, `upload`, `download`, `validate`, `preactivate`, `activate` or `fallback`;
- and its XML content is the succession of one or several `log` XML elements.

As defined by the following extract of XML schema `sessionLog.xsd` (see clause B.4.5):

```
            <element name="log" maxOccurs="unbounded">
              <complexType>
                <simpleContent>
                  <extension base="string">
                    <attribute name="time" type="time" use="required"/>
                    <attribute name="type" use="required">
                      <simpleType>
                        <restriction base="string">
                          <enumeration value="informative"/>
```

```
                    <enumeration value="error"/>
                  </restriction>
                </simpleType>
              </attribute>
              <attribute name="dn" type="string" use="optional"/>
              <attribute name="modifier" use="optional">
                <simpleType>
                  <restriction base="string">
                    <enumeration value="create"/>
                    <enumeration value="delete"/>
                    <enumeration value="update"/>
                  </restriction>
                </simpleType>
              </attribute>
            </extension>
          </simpleContent>
        </complexType>
      </element>
```

a `log` XML element:

- has the following XML attribute specifications:
  - a `time` XML attribute specification; this attribute specification carries the time the logged Bulk CM internal event occurred;
  - a `type` XML attribute specification; this attribute specification carries the type of the logged Bulk CM internal event, being either `informative` or `error`;
  - an optional `dn` XML attribute specification; this attribute specification carries the DN of the NRM instance associated with the logged Bulk CM internal event, if any;
  - an optional `modifier` XML attribute specification; this attribute specification carries the value of the `modifier` (see clause B.2.5) associated with the NRM instance, if any;
- and it has an XML content; this XML content carries the description of the logged Bulk CM internal event.

The following is an example of an `activity` XML element (in **bold**) in a session log XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<bulkCmSessionLogFile
  xmlns=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#sessionLog"
[…]
>
[…]
  <activity dateTime="2001-05-07T12:00:00+02:00" type="download">
    <log time="12:00:01+02:00" type="informative">
      Download requested with:
        downloadDataFileReference="ftp://a1.companyNN.com/data/upld123.xml"
    </log>
    <log time="12:00:02+02:00" type="error"
      dn="DC=a1.companyNN.com,SubNetwork=1"
      modifier="update"
    >
      No such instance
    </log>
  </activity>
[…]
</bulkCmSessionLogFile>
```

B.4 Solution Set definitions

B.4.1 XML definition structure

Clause B.4.2 provides a graphical representation of the XML elements.

Clause B.4.3 provides the base schema for configuration data XML files.

Clause B.4.4 provides an example of vendor-specific schema for configuration data XML files.

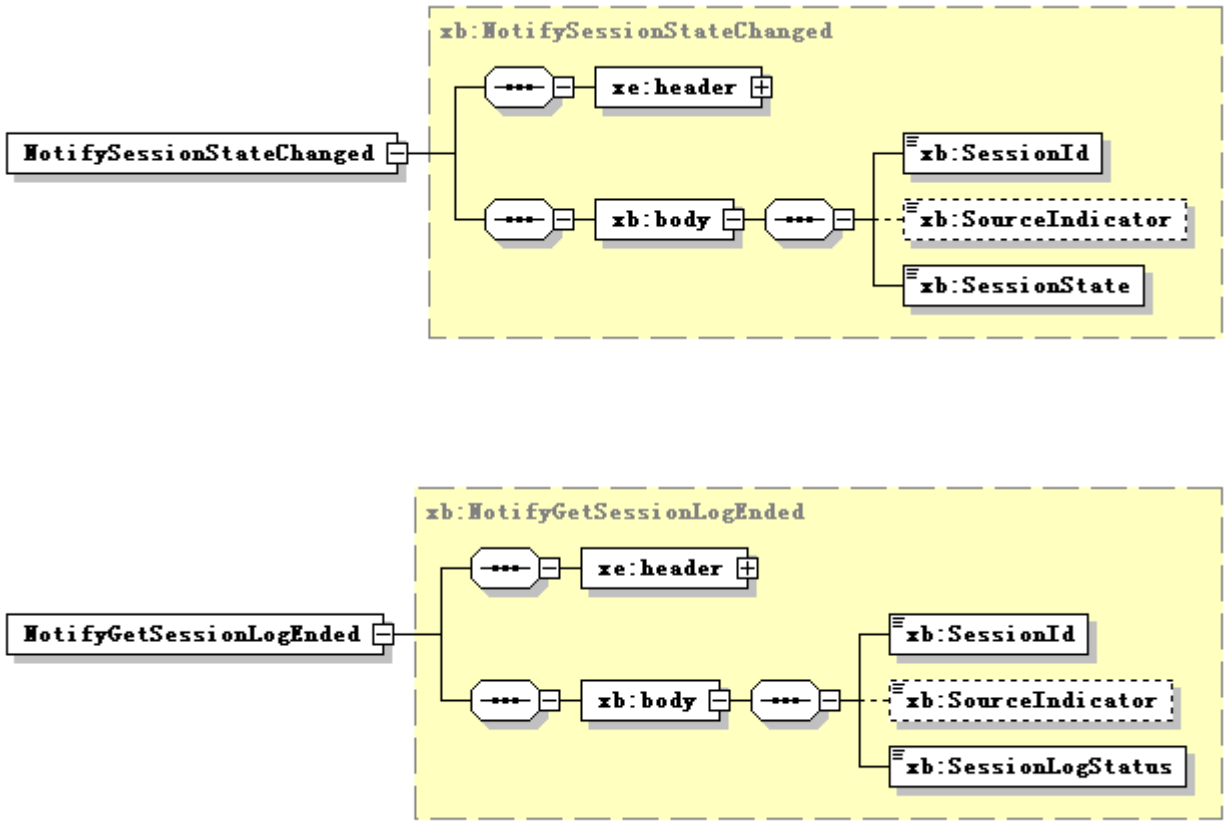
Clause B.4.5 provides the schema for session log XML files.

Clause B.4.6 provides the BulkCMIRP Notification XML Schema.

B.4.2 Graphical Representation

This clause provides XML definitions of Notification Log IRP notifications as defined in [9].

The structure of the BulkCMIRP Notification XML Definitions is shown in graphical depiction below:



The use of XML schema key word "sequence" to support IS-defined set (not sequence) is for the purpose of XML processor efficiency. This shall not imply the use of "sequence" in other technology.

B.4.3 XML Schema 'configData.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  3GPP TS 32.616 Bulk CM IRP
  Configuration data file base XML schema
  configData.xsd
-->

<schema
  targetNamespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#configData"

  elementFormDefault="qualified"

  xmlns="http://www.w3.org/2001/XMLSchema"

  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"

  xmlns:cn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.636#coreNrm"

  xmlns:un=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.646#utranNrm"

  xmlns:gn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.656#geranNrm"

  xmlns:stn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.746#stnNrm"

  xmlns:in=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.696#inventoryNrm"

  xmlns:tn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.716#transportNrm"

  xmlns:im=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.736#imsNrm"

  xmlns:rn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.726#repeaterNrm"

  xmlns:epc=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.756#epcNrm"

  xmlns:en=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.766#eutranNrm"

  xmlns:sp=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.526#sonPolicyNrm"

  xmlns:sn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.176#sumNrm"

  xmlns:hn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.776#hnsNrm"

  xmlns:hen=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.786#hensNrm"

  xmlns:gr=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.796#genericRanNrm"

>

   <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.636#coreNrm"
  />
```

```
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.646#utranNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.656#geranNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.746#stnNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.696#inventoryNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.716#transportNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.736#imsNrm"  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.726#repeaterNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.756#epcNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.766#eutranNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.526#sonPolicyNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.176#sumNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.726#repeaterNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.776#hnsNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.786#hensNrm"
  />
  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.796#genericRanNrm"
  />


  <!-- Configuration data file root XML element -->

  <element name="bulkCmConfigDataFile">
    <complexType>
      <sequence>
        <element name="fileHeader">
          <complexType>
            <attribute name="fileFormatVersion" type="string" use="required"/>
            <attribute name="senderName" type="string" use="optional"/>
            <attribute name="vendorName" type="string" use="optional"/>
          </complexType>
        </element>
        <element name="configData" maxOccurs="unbounded">
          <complexType>
            <choice>
              <element ref="xn:SubNetwork"/>
              <element ref="xn:MeContext"/>
```

```
              <element ref="xn:ManagedElement"/>
            </choice>
            <attribute name="dnPrefix" type="string" use="optional"/>
          </complexType>
        </element>
        <element name="fileFooter">
          <complexType>
            <attribute name="dateTime" type="dateTime" use="required"/>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>

</schema>
```

B.4.4 Example XML Schema 'NNRncHandOver.1.1.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  Configuration data file vendor-specific XML schema example
  NNRncHandOver.1.1.xsd
-->

<schema
  targetNamespace="http://www.companyNN.com/xmlschemas/NNRncHandOver.1.1"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xn=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
>

  <import
    namespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
  />

  <!-- RncHandOver version 1.1 company NN vendor-specific data -->

  <element name="vsDataRHO" substitutionGroup="xn:vsData">
    <complexType>
      <complexContent>
        <extension base="xn:vsData">
          <all>
            <element name="abcMin" minOccurs="0"/>
            <element name="abcMax" minOccurs="0"/>
          </all>
        </extension>
      </complexContent>
    </complexType>
  </element>

</schema>
```

B.4.5 XML Schema 'sessionLog.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  3GPP TS 32.616 Bulk CM IRP
  Session log file XML schema
  sessionLog.xsd
-->

<schema
  targetNamespace=
"http://www.3gpp.org/ftp/specs/archive/32_series/32.616#sessionLog"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
>

  <!-- Session log file root XML element -->

  <element name="bulkCmSessionLogFile">
    <complexType>
      <sequence>
        <element name="fileHeader">
          <complexType>
            <attribute name="fileFormatVersion" type="string" use="required"/>
            <attribute name="senderName" type="string" use="optional"/>
            <attribute name="vendorName" type="string" use="optional"/>
          </complexType>
        </element>
        <element name="activity" minOccurs="0" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="log" maxOccurs="unbounded">
                <complexType>
                  <simpleContent>
                    <extension base="string">
                      <attribute name="time" type="time" use="required"/>
                      <attribute name="type" use="required">
                        <simpleType>
                          <restriction base="string">
                            <enumeration value="informative"/>
                            <enumeration value="error"/>
                          </restriction>
                        </simpleType>
                      </attribute>
                      <attribute name="dn" type="string" use="optional"/>
                      <attribute name="modifier" use="optional">
                        <simpleType>
                          <restriction base="string">
                            <enumeration value="create"/>
                            <enumeration value="delete"/>
                            <enumeration value="update"/>
                          </restriction>
                        </simpleType>
                      </attribute>
                    </extension>
                  </simpleContent>
                </complexType>
              </element>
            </sequence>
            <attribute name="dateTime" type="dateTime" use="required"/>
            <attribute name="type" use="required">
              <simpleType>
                <restriction base="string">
                  <enumeration value="upload"/>
                  <enumeration value="download"/>
                  <enumeration value="validate"/>
                  <enumeration value="preactivate"/>
                  <enumeration value="activate"/>
                  <enumeration value="fallback"/>
                </restriction>
              </simpleType>
            </attribute>
          </complexType>
        </element>
        <element name="fileFooter">
```

```
          <complexType>
            <attribute name="dateTime" type="dateTime" use="required"/>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>

</schema>
```

B.4.6 XML Schema 'bulkCMIRPNotif.xsd'

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    3GPP TS 32.616 BulkCMIRP Notification XML Schema
    bulkCMIRPNotif.xsd
-->
<schema xmlns:xb="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#bulkCMIRPNotif"
xmlns:xe="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#notification"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#bulkCMIRPNotif"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <import namespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#notification"/>
    <simpleType name="SessionId">
        <restriction base="string">
            <minLength value="1"/>
        </restriction>
    </simpleType>
    <simpleType name="SourceIndicator">
        <restriction base="string">
            <enumeration value="Resource Operation"/>
            <enumeration value="management Operation"/>
            <enumeration value="unknown"/>
        </restriction>
    </simpleType>
    <simpleType name="SessionState">
        <restriction base="string">
            <enumeration value="Upload Failed"/>
            <enumeration value="Upload Completed"/>
            <enumeration value="Download Failed"/>
            <enumeration value="Download Completed"/>
            <enumeration value="Validation Failed"/>
            <enumeration value="Validation Completed"/>
            <enumeration value="Preactivation Failed"/>
            <enumeration value="Preactivation Partly Realised"/>
            <enumeration value="Preactivation Completed"/>
            <enumeration value="Activation Failed"/>
            <enumeration value="Activation Partly Realised"/>
            <enumeration value="Activation Completed"/>
            <enumeration value="Fallback Failed"/>
            <enumeration value="Fallback Partly Realised"/>
            <enumeration value="Fallback Completed"/>
        </restriction>
    </simpleType>
    <simpleType name="SessionLogStatus">
        <restriction base="string">
            <enumeration value="GetSessionLog completed successfully"/>
            <enumeration value="GetSessionLog completed unsuccessfully"/>
        </restriction>
    </simpleType>
    <complexType name="NotifySessionStateChanged">
        <complexContent>
            <extension base="xe:Notification">
                <sequence>
                    <element name="body">
                        <complexType>
                            <sequence>
                                <element name="SessionId" type="xb:SessionId"/>
                                <element name="SessionState" type="xb:SessionState"/>
                                <element name="SourceIndicator" type="xb:SourceIndicator"
minOccurs="0"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="NotifyGetSessionLogEnded">
        <complexContent>
            <extension base="xe:Notification">
                <sequence>
                    <element name="body">
                        <complexType>
                            <sequence>
                                <element name="SessionId" type="xb:SessionId"/>
```

```
                                <element name="SessionLogStatus" type="xb:SessionLogStatus"/>
                                <element name="SourceIndicator" type="xb:SourceIndicator"
minOccurs="0"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="NotifySessionStateChanged" type="xb:NotifySessionStateChanged"/>
    <element name="NotifyGetSessionLogEnded" type="xb:NotifyGetSessionLogEnded"/>
</schema>
```

Annex C (normative): SOAP Solution Set

This annex specifies the SOAP Solution Set for the IRP whose semantics are specified in 3GPP TS 32.612 [9].

C.1 Architectural features

The overall architectural feature of the Bulk CM IRP is specified in 3GPP TS 32.612 [9]. This clause specifies features that are specific to the SOAP Solution Set.

The Bulk CM IRP SOAP SS uses the Notification IRP SOAP SS of 3GPP TS 32.306 [5]. The IRPAgent shall support the push interface model, which means that the IRPAgent sends Bulk CM notifications to the IRPManager as soon as new events occur. The IRPManager does not need to check ("pull") for events.

C.1.1 Syntax for Distinguished Names

The syntax of a Distinguished Name is defined in 3GPP TS 32.300 [4].

C.1.2 Supported W3C specifications

The SOAP 1.1 specification [32] and WSDL 1.1 specification [34] are supported.

The SOAP 1.2 specification [35] is supported optionally.

This specification uses "document" style in WSDL file.

This specification uses "literal" encoding style in WSDL file.

C.1.3 Prefixes and namespaces

This specification uses a number of namespace prefixes throughout that are listed in Table C.1.3.1.

Table C.1.3.1: Prefixes and Namespaces used in this specification

PREFIX	NAMESPACE
(no prefix)	http://schemas.xmlsoap.org/wsdl/
soap	http://schemas.xmlsoap.org/wsdl/soap/
bulkCMIRPSystem	http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRPSystem
bulkCMIRPData	http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRPData
xn	http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm
genericIRPSystem	http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRPSystem
ntfIRPNtfSystem	http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfSystem

C.1.4 Filter language

The filter language used in the SS is the XPath Language (see W3C XPath 1.0 specification [33]). IRPAgents may throw a FilterComplexityLimit fault when a given filter is too complex.

C.2 Mapping

C.2.1 Operation and notification mapping

The Bulk CM IRP IS (3GPP TS 32.612 [9]) defines the operations and their semantics.

Table C.2.1 maps the operations defined in the Bulk CM IRP IS to their equivalent types, messages, port type operation, and binding operation in this Solution Set (SS).

Table C.2.1 also maps the notifications of the Bulk CM IRP IS, as well as inherited operations.

Table C.2.1 also qualifies if an operation is Mandatory (M) or Optional (O).

Table C.2.1: Mapping from IS Operation to SS Equivalents

IS Operation in 3GPP TS 32.612 [9]	SS: Operation for WSDL port type and WSDL binding	SS: Port of BulkCMIRPService	Qualifier
startSession	startSession (note 1)	BulkCMIRPPort	M
endSession	endSession (note 1)	BulkCMIRPPort	M
abortSessionOperation	abortSessionOperation (note 1)	BulkCMIRPPort	M
getSessionIds	getSessionIds (note 1)	BulkCMIRPPort	M
getSessionStatus	getSessionStatus (note 1)	BulkCMIRPPort	M
getSessionLog	getSessionLog (note 1)	BulkCMIRPPort	M
upload	upload (note 1)	BulkCMIRPPort	M
download	download (note 1)	BulkCMIRPPort	M
validate	validate (note 1)	BulkCMIRPPort	O
preactivate	preactivate (note 1)	BulkCMIRPPort	O
activate	activate (note 1)	BulkCMIRPPort	M
fallback	fallback (note 1)	BulkCMIRPPort	M
notifySessionStateChanged	notify (note 2)	NotificationIRPNtfPort	M
notifyGetSessionLogEnded	notify (note 2)	NotificationIRPNtfPort	M
getIRPVersion (note 3)	See TS 32.316 [7]	GenericIRPPort	M
getOperationProfile (note 3)	See TS 32.316 [7]	GenericIRPPort	O
getNotificationProfile (note 3)	See TS 32.316 [7]	GenericIRPPort	O
NOTE 1: The operation is under the port type bulkCMIRPSystem:BulkCMIRPPortType and under the binding bulkCMIRPSystem:BulkCMIRPBinding.			
NOTE 2: The IS equivalent maps to an XML definition specified in Annex B, and this being an input parameter to the operation notify under the port type ntfIRPNtfSystem:NotificationIRPNtf and under the binding ntfIRPNtfSystem:NotificationIRPNtf of 3GPP TS 32.306 [5]. This binding is linked to a port of the BulkCMIRPService as indicated in the table above.			
NOTE 3: The IS operation is inherited from the ManagedGenericIRP IOC specified in 3GPP TS 32.312 [6]. This inheritance is by the BulkCMIRP IOC of 3GPP TS 32.612 [9] inheriting from the ManagedGenericIRP IOC. The corresponding binding is linked to a port of the BulkCMIRPService as indicated in the table above.			

C.2.2 Operation parameter mapping

The Bulk CM IRP IS (3GPP TS 32.612 [9]) defines semantics of parameters carried in operations. The tables below show the mapping of these parameters, as per operation, to their equivalents defined in this SS.

C.2.2.1 Operation `startSession`

C.2.2.1.1 Input parameters

Table C.2.2.1.1: Mapping from IS startSession input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M

C.2.2.1.2 Output parameters

Table C.2.2.1.2: Mapping from IS startSession output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.1.3 Fault definition

Table C.2.2.1.3: Mapping from IS startSession exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.2 Operation `endSession`

C.2.2.2.1 Input parameters

Table C.2.2.2.1: Mapping from IS endSession input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M

C.2.2.2.2 Output parameters

Table C.2.2.2.2: Mapping from IS endSession output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.2.3 Fault definition

Table C.2.2.2.3: Mapping from IS endSession exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.3 Operation abortSessionOperation

C.2.2.3.1 Input parameters

Table C.2.2.3.1: Mapping from IS abortSessionOperation input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M

C.2.2.3.2 Output parameters

Table C.2.2.3.2: Mapping from IS abortSessionOperation output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.3.3 Fault definition

Table C.2.2.3.3: Mapping from IS abortSessionOperation exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.4 Operation `getSessionIds`

C.2.2.4.1 Input parameters

None.

C.2.2.4.2 Output parameters

Table C.2.2.4.2: Mapping from IS getSessionIds output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionIdList	sessionIdList	M
status	status	M

C.2.2.4.3 Fault definition

Table C.2.2.4.3: Mapping from IS getSessionIds exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.5 Operation `getSessionStatus`

C.2.2.5.1 Input parameters

Table C.2.2.5.1: Mapping from IS getSessionStatus input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M

C.2.2.5.2 Output parameters

Table C.2.2.5.2: Mapping from IS getSessionStatus output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionState	sessionState	M
status	status	M

C.2.2.5.3 Fault definition

Table C.2.2.5.3: Mapping from IS getSessionStatus exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.6 Operation `getSessionLog`

C.2.2.6.1 Input parameters

Table C.2.2.6.1: Mapping from IS getSessionLog input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M
logFileReference	logFileReference	M
contentType	contentType	M

C.2.2.6.2 Output parameters

Table C.2.2.6.2: Mapping from IS getSessionLog output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.6.3 Fault definition

Table C.2.2.6.3: Mapping from IS getSessionLog exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.7 Operation upload

C.2.2.7.1 Input parameters

Table C.2.2.7.1: Mapping from IS upload input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M
uploadDataFileReference	uploadDataFileReference	M
baseObjectInstance	baseObjectInstance	M
scope	scope	M
filter	filter	M

C.2.2.7.2 Output parameters

Table C.2.2.7.2: Mapping from IS upload output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.7.3 Fault definition

Table C.2.2.7.3: Mapping from IS upload exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.8 Operation download

C.2.2.8.1 Input parameters

Table C.2.2.8.1: Mapping from IS download input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M
downloadDataFileReference	downloadDataFileReference	M

C.2.2.8.2 Output parameters

Table C.2.2.8.2: Mapping from IS download output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.8.3 Fault definition

Table C.2.2.8.3: Mapping from IS download exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.9 Operation `validate`

C.2.2.9.1 Input parameters

Table C.2.2.9.1: Mapping from IS validate input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M
activationMode	activationMode	O

C.2.2.9.2 Output parameters

Table C.2.2.9.2: Mapping from IS validate output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.9.3 Fault definition

Table C.2.2.9.3: Mapping from IS validate exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.10 Operation `preactivate`

C.2.2.10.1 Input parameters

Table C.2.2.10.1: Mapping from IS preactivate input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M
verificationMode	verificationMode	O
activationMode	activationMode	O
fallbackEnabled	fallbackEnabled	M

C.2.2.10.2 Output parameters

Table C.2.2.10.2: Mapping from IS preactivate output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.10.3 Fault definition

Table C.2.2.10.3: Mapping from IS preactivate exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.11 Operation `activate`

C.2.2.11.1 Input parameters

Table C.2.2.11.1: Mapping from IS activate input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M
activationMode	activationMode	O
fallbackEnabled	fallbackEnabled	M

C.2.2.11.2 Output parameters

Table C.2.2.11.2: Mapping from IS activate output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.11.3 Fault definition

Table C.2.2.11.3: Mapping from IS activate exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.2.2.12 Operation `fallback`

C.2.2.12.1 Input parameters

Table C.2.2.12.1: Mapping from IS fallback input parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding input message under corresponding port type operation as indicated in Table C.2.1	Qualifier
sessionId	sessionId	M

C.2.2.12.2 Output parameters

Table C.2.2.12.2: Mapping from IS fallback output parameters to SS equivalents

IS Operation parameter	SS WSDL type sub-element used in corresponding output message under corresponding port type operation as indicated in Table C.2.1	Qualifier
status	status	M

C.2.2.12.3 Fault definition

Table C.2.2.12.3: Mapping from IS fallback exceptions to SS equivalents

Assertion name	SS WSDL type enumeration value used in corresponding fault message under corresponding port type operation as indicated in Table C.2.1	Qualifier
operation_failed	OperationFailed	M

C.3 Solution Set definitions

C.3.1 WSDL definition structure

Clause C.3.2 provides a graphical representation of the Bulk CM IRP service.

Clause C.3.3 defines the services which are supported by the Bulk CM IRP agent.

C.3.2 Graphical Representation

The WSDL structure is depicted in Figure C.3.2 below, depicting port type, binding and service. The port type contains port type operations, which again contains input, output and fault messages. The binding contains binding operations, which have the same name as the port type operations. The binding connects to a port inside the service.

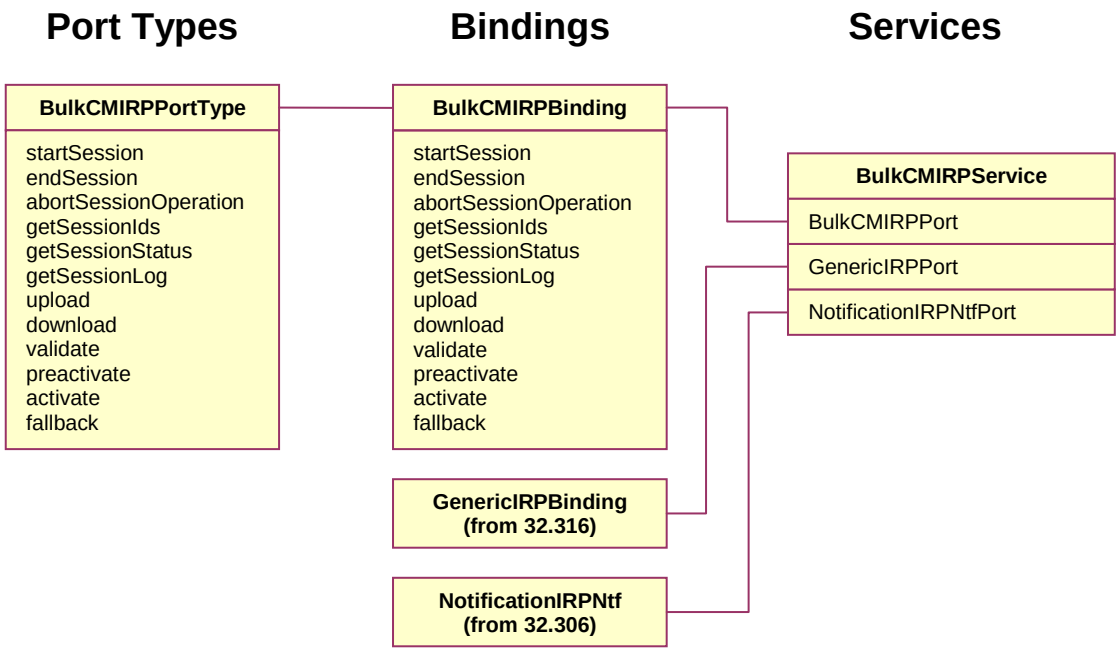


Figure C.3.2: Bulk CM IRP SOAP Solution Set WSDL structure

C.3.3 WSDL specification 'BulkCMIRPSystem.wsdl'

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    3GPP TS 32.616 Bulk CM IRP SOAP Solution Set
-->
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:bulkCMIRPSystem="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRPSystem"
xmlns:bulkCMIRPData="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRPData"
xmlns:xn="http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"
xmlns:genericIRPSystem="http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRPSystem"
xmlns:ntfIRPNtfSystem="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfSyst
em" targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRPSystem">
    <import
namespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtfSystem"/>
    <import namespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRPSystem"/>
    <types>
        <schema
targetNamespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRPData"
xmlns="http://www.w3.org/2001/XMLSchema">
            <import namespace="http://www.3gpp.org/ftp/specs/archive/32_series/32.626#genericNrm"/>
            <!-- The following types are defined for the Bulk CM IRP operations -->
            <simpleType name="OperationStatusTwo">
                <restriction base="string">
                    <enumeration value="OperationSucceeded"/>
                    <enumeration value="OperationFailed"/>
                </restriction>
            </simpleType>
            <complexType name="SessionIdList">
                <sequence>
                    <element name="sessionId" type="string" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
            <simpleType name="SessionState">
                <restriction base="string">
                    <enumeration value="Idle"/>
                    <enumeration value="UploadInProgress"/>
                    <enumeration value="UploadFailed"/>
                    <enumeration value="UploadCompleted"/>
                    <enumeration value="DownloadInProgress"/>
                    <enumeration value="DownloadFailed"/>
                    <enumeration value="DownloadCompleted"/>
                    <enumeration value="ValidationInProgress"/>
                    <enumeration value="ValidationFailed"/>
                    <enumeration value="ValidationCompleted"/>
                    <enumeration value="PreactivationInProgress"/>
                    <enumeration value="PreactivationFailed"/>
                    <enumeration value="PreactivationPartlyRealised"/>
                    <enumeration value="PreactivationCompleted"/>
                    <enumeration value="ActivationInProgress"/>
                    <enumeration value="ActivationFailed"/>
                    <enumeration value="ActivationPartlyRealised"/>
                    <enumeration value="ActivationCompleted"/>
                    <enumeration value="FallbackInProgress"/>
                    <enumeration value="FallbackFailed"/>
                    <enumeration value="FallbackPartlyRealised"/>
                    <enumeration value="FallbackCompleted"/>
                </restriction>
            </simpleType>
            <simpleType name="ContentType">
                <restriction base="string">
                    <enumeration value="CompleteLog"/>
                    <enumeration value="ErrorsOnly"/>
                </restriction>
            </simpleType>
            <simpleType name="ValueIsNotRelevant">
                <restriction base="string">
                    <enumeration value="ValueIsNotRelevant"/>
                </restriction>
            </simpleType>
            <complexType name="Scope">
            <complexType name="Scope">
                <sequence>
                    <element name="level" type="nonNegativeInteger"/>
                    <choice>
                        <element name="baseOnly" type="bulkCMIRPData:ValueIsNotRelevant"/>
                            <element name="baseNthLevel" type="nonNegativeInteger"/>
                        <element name="baseSubtree" type="nonNegativeInteger"/>
                        <element name="baseAll" type="bulkCMIRPData:ValueIsNotRelevant"/>
                    </choice>
                </sequence>
            </complexType>
            <simpleType name="ActivationMode">
                <restriction base="string">
                    <enumeration value="LeastServiceImpact"/>
```

```
            <enumeration value="LeastElapseTime"/>
            <enumeration value="NoIndication"/>
        </restriction>
    </simpleType>
    <simpleType name="VerificationMode">
        <restriction base="string">
            <enumeration value="FullChecking"/>
            <enumeration value="LimitedChecking"/>
        </restriction>
    </simpleType>
    <simpleType name="RequiredOrNot">
        <restriction base="string">
            <enumeration value="Required"/>
            <enumeration value="NotRequired"/>
        </restriction>
    </simpleType>
    <!--  startSession Request-->
    <element name="startSession">
        <complexType>
            <sequence>
                <element name="sessionId" type="string"/>
            </sequence>
        </complexType>
    </element>
    <!--  startSession Response -->
    <element name="startSessionResponse">
        <complexType>
            <sequence>
                <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
            </sequence>
        </complexType>
    </element>
    <!--  startSession Fault -->
    <element name="startSessionFault">
        <simpleType>
            <restriction base="string">
                <enumeration value="OperationFailed"/>
            </restriction>
        </simpleType>
    </element>
    <!-- endSession Request-->
    <element name="endSession">
        <complexType>
            <sequence>
                <element name="sessionId" type="string"/>
            </sequence>
        </complexType>
    </element>
    <!-- endSession Response -->
    <element name="endSessionResponse">
        <complexType>
            <sequence>
                <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
            </sequence>
        </complexType>
    </element>
    <!-- endSession Fault -->
    <element name="endSessionFault">
        <simpleType>
            <restriction base="string">
                <enumeration value="OperationFailed"/>
            </restriction>
        </simpleType>
    </element>
    <!-- abortSessionOperation Request-->
    <element name="abortSessionOperation">
        <complexType>
            <sequence>
                <element name="sessionId" type="string"/>
            </sequence>
        </complexType>
    </element>
    <!-- abortSessionOperation Response -->
    <element name="abortSessionOperationResponse">
        <complexType>
            <sequence>
                <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
            </sequence>
        </complexType>
    </element>
    <!-- abortSessionOperation Fault -->
    <element name="abortSessionOperationFault">
        <simpleType>
            <restriction base="string">
                <enumeration value="OperationFailed"/>
            </restriction>
        </simpleType>
    </element>
    <!-- getSessionIds Request-->
```

```
<element name="getSessionIds"/>
<!-- getSessionIds Response -->
<element name="getSessionIdsResponse">
    <complexType>
        <sequence>
            <element name="sessionIdList" type="bulkCMIRPData:SessionIdList"/>
            <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
        </sequence>
    </complexType>
</element>
<!-- No fault message is defined for getSessionIds -->
<!-- getSessionStatus Request-->
<element name="getSessionStatus">
    <complexType>
        <sequence>
            <element name="sessionId" type="string"/>
        </sequence>
    </complexType>
</element>
<!-- getSessionStatus Response -->
<element name="getSessionStatusResponse">
    <complexType>
        <sequence>
            <element name="sessionState" type="bulkCMIRPData:SessionState"/>
            <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
        </sequence>
    </complexType>
</element>
<!-- getSessionStatus Fault -->
<element name="getSessionStatusFault">
    <simpleType>
        <restriction base="string">
            <enumeration value="OperationFailed"/>
        </restriction>
    </simpleType>
</element>
<!-- getSessionLog Request-->
<element name="getSessionLog">
    <complexType>
        <sequence>
            <element name="sessionId" type="string"/>
            <element name="logFileReference" type="string"/>
            <element name="contentType" type="bulkCMIRPData:ContentType"/>
        </sequence>
    </complexType>
</element>
<!-- getSessionLog Response -->
<element name="getSessionLogResponse">
    <complexType>
        <sequence>
            <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
        </sequence>
    </complexType>
</element>
<!-- getSessionStatus Fault -->
<element name="getSessionLogFault">
    <simpleType>
        <restriction base="string">
            <enumeration value="OperationFailed"/>
        </restriction>
    </simpleType>
</element>
<!-- upload Request-->
<element name="upload">
    <complexType>
        <sequence>
            <element name="sessionId" type="string"/>
            <element name="uploadDataFileReference" type="anyURI"/>
            <element name="baseObjectInstance" type="xn:dn"/>
            <element name="scope" type="bulkCMIRPData:Scope"/>
            <element name="filter" type="string"/>
        </sequence>
    </complexType>
</element>
<!-- upload Response -->
<element name="uploadResponse">
    <complexType>
        <sequence>
            <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
        </sequence>
    </complexType>
</element>
<!-- upload Fault -->
<element name="uploadFault">
    <simpleType>
        <restriction base="string">
            <enumeration value="OperationFailed"/>
        </restriction>
    </simpleType>
```

```
            </element>
            <!-- download Request-->
            <element name="download">
                <complexType>
                    <sequence>
                        <element name="sessionId" type="string"/>
                        <element name="downloadDataFileReference" type="anyURI"/>
                    </sequence>
                </complexType>
            </element>
            <!-- download Response -->
            <element name="downloadResponse">
                <complexType>
                    <sequence>
                        <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
                    </sequence>
                </complexType>
            </element>
            <!-- download Fault -->
            <element name="downloadFault">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="OperationFailed"/>
                    </restriction>
                </simpleType>
            </element>
            <!-- validate Request-->
            <element name="validate">
                <complexType>
                    <sequence>
                        <element name="sessionId" type="string"/>
                        <element name="activationMode" type="bulkCMIRPData:ActivationMode"
minOccurs="0"/>
                    </sequence>
                </complexType>
            </element>
            <!-- validate Response -->
            <element name="validateResponse">
                <complexType>
                    <sequence>
                        <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
                    </sequence>
                </complexType>
            </element>
            <!-- validate Fault -->
            <element name="validateFault">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="OperationFailed"/>
                    </restriction>
                </simpleType>
            </element>
            <!-- preactivate Request-->
            <element name="preactivate">
                <complexType>
                    <sequence>
                        <element name="sessionId" type="string"/>
                        <element name="verificationMode" type="bulkCMIRPData:VerificationMode"
minOccurs="0"/>
                        <element name="activationMode" type="bulkCMIRPData:ActivationMode"
minOccurs="0"/>
                        <element name="fallbackEnabled" type="bulkCMIRPData:RequiredOrNot"/>
                    </sequence>
                </complexType>
            </element>
            <!-- preactivate Response -->
            <element name="preactivateResponse">
                <complexType>
                    <sequence>
                        <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
                    </sequence>
                </complexType>
            </element>
            <!-- preactivate Fault -->
            <element name="preactivateFault">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="OperationFailed"/>
                    </restriction>
                </simpleType>
            </element>
            <!-- activate Request-->
            <element name="activate">
                <complexType>
                    <sequence>
                        <element name="sessionId" type="string"/>
                        <element name="activationMode" type="bulkCMIRPData:ActivationMode"
minOccurs="0"/>
                        <element name="fallbackEnabled" type="bulkCMIRPData:RequiredOrNot"/>
```

```
                </sequence>
            </complexType>
        </element>
        <!-- activate Response -->
        <element name="activateResponse">
            <complexType>
                <sequence>
                    <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
                </sequence>
            </complexType>
        </element>
        <!-- activate Fault -->
        <element name="activateFault">
            <simpleType>
                <restriction base="string">
                    <enumeration value="OperationFailed"/>
                </restriction>
            </simpleType>
        </element>
        <!-- fallback Request-->
        <element name="fallback">
            <complexType>
                <sequence>
                    <element name="sessionId" type="string"/>
                </sequence>
            </complexType>
        </element>
        <!-- fallback Response -->
        <element name="fallbackResponse">
            <complexType>
                <sequence>
                    <element name="status" type="bulkCMIRPData:OperationStatusTwo"/>
                </sequence>
            </complexType>
        </element>
        <!-- fallback Fault -->
        <element name="fallbackFault">
            <simpleType>
                <restriction base="string">
                    <enumeration value="OperationFailed"/>
                </restriction>
            </simpleType>
        </element>
    </schema>
</types>
<message name="startSession">
    <part name="parameter" element="bulkCMIRPData:startSession"/>
</message>
<message name="startSessionResponse">
    <part name="parameter" element="bulkCMIRPData:startSessionResponse"/>
</message>
<message name="startSessionFault">
    <part name="parameter" element="bulkCMIRPData:startSessionFault"/>
</message>
<message name="endSession">
    <part name="parameter" element="bulkCMIRPData:endSession"/>
</message>
<message name="endSessionResponse">
    <part name="parameter" element="bulkCMIRPData:endSessionResponse"/>
</message>
<message name="endSessionFault">
    <part name="parameter" element="bulkCMIRPData:endSessionFault"/>
</message>
<message name="abortSessionOperation">
    <part name="parameter" element="bulkCMIRPData:abortSessionOperation"/>
</message>
<message name="abortSessionOperationResponse">
    <part name="parameter" element="bulkCMIRPData:abortSessionOperationResponse"/>
</message>
<message name="abortSessionOperationFault">
    <part name="parameter" element="bulkCMIRPData:abortSessionOperationFault"/>
</message>
<message name="getSessionIds">
    <part name="parameter" element="bulkCMIRPData:getSessionIds"/>
</message>
<message name="getSessionIdsResponse">
    <part name="parameter" element="bulkCMIRPData:getSessionIdsResponse"/>
</message>
<message name="getSessionStatus">
    <part name="parameter" element="bulkCMIRPData:getSessionStatus"/>
</message>
<message name="getSessionStatusResponse">
    <part name="parameter" element="bulkCMIRPData:getSessionStatusResponse"/>
</message>
<message name="getSessionStatusFault">
    <part name="parameter" element="bulkCMIRPData:getSessionStatusFault"/>
</message>
<message name="getSessionLog">
    <part name="parameter" element="bulkCMIRPData:getSessionLog"/>
```

```
    </message>
    <message name="getSessionLogResponse">
        <part name="parameter" element="bulkCMIRPData:getSessionLogResponse"/>
    </message>
    <message name="getSessionLogFault">
        <part name="parameter" element="bulkCMIRPData:getSessionLogFault"/>
    </message>
    <message name="upload">
        <part name="parameter" element="bulkCMIRPData:upload"/>
    </message>
    <message name="uploadResponse">
        <part name="parameter" element="bulkCMIRPData:uploadResponse"/>
    </message>
    <message name="uploadFault">
        <part name="parameter" element="bulkCMIRPData:uploadFault"/>
    </message>
    <message name="download">
        <part name="parameter" element="bulkCMIRPData:download"/>
    </message>
    <message name="downloadResponse">
        <part name="parameter" element="bulkCMIRPData:downloadResponse"/>
    </message>
    <message name="downloadFault">
        <part name="parameter" element="bulkCMIRPData:downloadFault"/>
    </message>
    <message name="validate">
        <part name="parameter" element="bulkCMIRPData:validate"/>
    </message>
    <message name="validateResponse">
        <part name="parameter" element="bulkCMIRPData:validateResponse"/>
    </message>
    <message name="validateFault">
        <part name="parameter" element="bulkCMIRPData:validateFault"/>
    </message>
    <message name="preactivate">
        <part name="parameter" element="bulkCMIRPData:preactivate"/>
    </message>
    <message name="preactivateResponse">
        <part name="parameter" element="bulkCMIRPData:preactivateResponse"/>
    </message>
    <message name="preactivateFault">
        <part name="parameter" element="bulkCMIRPData:preactivateFault"/>
    </message>
    <message name="activate">
        <part name="parameter" element="bulkCMIRPData:activate"/>
    </message>
    <message name="activateResponse">
        <part name="parameter" element="bulkCMIRPData:activateResponse"/>
    </message>
    <message name="activateFault">
        <part name="parameter" element="bulkCMIRPData:activateFault"/>
    </message>
    <message name="fallback">
        <part name="parameter" element="bulkCMIRPData:fallback"/>
    </message>
    <message name="fallbackResponse">
        <part name="parameter" element="bulkCMIRPData:fallbackResponse"/>
    </message>
    <message name="fallbackFault">
        <part name="parameter" element="bulkCMIRPData:fallbackFault"/>
    </message>
    <portType name="BulkCMIRPPortType">
        <operation name="startSession">
            <input message="bulkCMIRPSystem:startSession"/>
            <output message="bulkCMIRPSystem:startSessionResponse"/>
            <fault name="startSessionFault" message="bulkCMIRPSystem:startSessionFault"/>
        </operation>
        <operation name="endSession">
            <input message="bulkCMIRPSystem:endSession"/>
            <output message="bulkCMIRPSystem:endSessionResponse"/>
            <fault name="endSessionFault" message="bulkCMIRPSystem:endSessionFault"/>
        </operation>
        <operation name="abortSessionOperation">
            <input message="bulkCMIRPSystem:abortSessionOperation"/>
            <output message="bulkCMIRPSystem:abortSessionOperationResponse"/>
            <fault name="abortSessionOperationFault"
message="bulkCMIRPSystem:abortSessionOperationFault"/>
        </operation>
        <operation name="getSessionIds">
            <input message="bulkCMIRPSystem:getSessionIds"/>
            <output message="bulkCMIRPSystem:getSessionIdsResponse"/>
        </operation>
        <operation name="getSessionStatus">
            <input message="bulkCMIRPSystem:getSessionStatus"/>
            <output message="bulkCMIRPSystem:getSessionStatusResponse"/>
            <fault name="getSessionStatusFault" message="bulkCMIRPSystem:getSessionStatusFault"/>
        </operation>
        <operation name="getSessionLog">
            <input message="bulkCMIRPSystem:getSessionLog"/>
```

```
            <output message="bulkCMIRPSystem:getSessionLogResponse"/>
            <fault name="getSessionLogFault" message="bulkCMIRPSystem:getSessionLogFault"/>
        </operation>
        <operation name="upload">
            <input message="bulkCMIRPSystem:upload"/>
            <output message="bulkCMIRPSystem:uploadResponse"/>
            <fault name="uploadFault" message="bulkCMIRPSystem:uploadFault"/>
        </operation>
        <operation name="download">
            <input message="bulkCMIRPSystem:download"/>
            <output message="bulkCMIRPSystem:downloadResponse"/>
            <fault name="downloadFault" message="bulkCMIRPSystem:downloadFault"/>
        </operation>
        <operation name="validate">
            <input message="bulkCMIRPSystem:validate"/>
            <output message="bulkCMIRPSystem:validateResponse"/>
            <fault name="validateFault" message="bulkCMIRPSystem:validateFault"/>
        </operation>
        <operation name="preactivate">
            <input message="bulkCMIRPSystem:preactivate"/>
            <output message="bulkCMIRPSystem:preactivateResponse"/>
            <fault name="preactivateFault" message="bulkCMIRPSystem:preactivateFault"/>
        </operation>
        <operation name="activate">
            <input message="bulkCMIRPSystem:activate"/>
            <output message="bulkCMIRPSystem:activateResponse"/>
            <fault name="activateFault" message="bulkCMIRPSystem:activateFault"/>
        </operation>
        <operation name="fallback">
            <input message="bulkCMIRPSystem:fallback"/>
            <output message="bulkCMIRPSystem:fallbackResponse"/>
            <fault name="fallbackFault" message="bulkCMIRPSystem:fallbackFault"/>
        </operation>
    </portType>
    <binding name="BulkCMIRPBinding" type="bulkCMIRPSystem:BulkCMIRPPortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="startSession">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#startSession" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="startSessionFault">
                <soap:fault name="startSessionFault" use="literal"/>
            </fault>
        </operation>
        <operation name="endSession">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#endSession" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="endSessionFault">
                <soap:fault name="endSessionFault" use="literal"/>
            </fault>
        </operation>
        <operation name="abortSessionOperation">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#abortSessionOperation"
style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="abortSessionOperationFault">
                <soap:fault name="abortSessionOperationFault" use="literal"/>
            </fault>
        </operation>
        <operation name="getSessionIds">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#getSessionIds" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <!-- No fault message for this operation -->
        </operation>
        <operation name="getSessionStatus">
```

```
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#getSessionStatus"
style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="getSessionStatusFault">
                <soap:fault name="getSessionStatusFault" use="literal"/>
            </fault>
        </operation>
        <operation name="getSessionLog">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#getSessionLog" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="getSessionLogFault">
                <soap:fault name="getSessionLogFault" use="literal"/>
            </fault>
        </operation>
        <operation name="upload">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#upload" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="uploadFault">
                <soap:fault name="uploadFault" use="literal"/>
            </fault>
        </operation>
        <operation name="download">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#download" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="downloadFault">
                <soap:fault name="downloadFault" use="literal"/>
            </fault>
        </operation>
        <operation name="validate">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#validate" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="validateFault">
                <soap:fault name="validateFault" use="literal"/>
            </fault>
        </operation>
        <operation name="preactivate">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#preactivate" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="preactivateFault">
                <soap:fault name="preactivateFault" use="literal"/>
            </fault>
        </operation>
        <operation name="activate">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#activate" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="activateFault">
                <soap:fault name="activateFault" use="literal"/>
```

```
            </fault>
        </operation>
        <operation name="fallback">
            <soap:operation
soapAction="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#fallback" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="fallbackFault">
                <soap:fault name="fallbackFault" use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="BulkCMIRPService">
        <port name="BulkCMIRPPort" binding="bulkCMIRPSystem:BulkCMIRPBinding">
            <soap:address
location="http://www.3gpp.org/ftp/specs/archive/32_series/32.616#BulkCMIRP"/>
        </port>
        <port name="GenericIRPPort" binding="genericIRPSystem:GenericIRPBinding">
            <soap:address
location="http://www.3gpp.org/ftp/specs/archive/32_series/32.316#GenericIRP"/>
        </port>
        <port name="NotificationIRPNtfPort" binding="ntfIRPNtfSystem:NotificationIRPNtf">
            <soap:address
location="http://www.3gpp.org/ftp/specs/archive/32_series/32.306#NotificationIRPNtf"/>
        </port>
    </service>
</definitions>
```

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2010-09	SA#49	SP-100517	--	--	Presentation to SA for Information and Approval	---	1.0.0
2010-10	--	--	--	--	Publication	1.0.0	10.0.0
2010-12	SA#50	SP-100833	001	--	Correcting Bulk CM IRP SS - Align with 32.612 IS	10.0.0	10.1.0
2012-09	SA#57	-	-	-	Automatic upgrade from previous Release version 10.1.0	10.1.0	11.0.0
2013-12	SA#62	SP-130613	003	-	Add missing support for generic RAN NRM	11.0.0	11.1.0
2014-09	SA#65	SP-140559	004	-	Update the link from Solution Set to Information Service due to the end of Release 12	11.1.0	12.0.0
2016-01	-	-	-	-	Update to Rel-13 version (MCC)	12.0.0	**13.0.0**

History

Document history		
V13.0.0	February 2016	Publication