CPM Message Store

Candidate Version 2.2 – 26 Sep 2017

Open Mobile Alliance

OMA-TS-CPM_Message_Store-V2_2-20170926-C

Use of this document is subject to all of the terms and conditions of the Use Agreement located at http://www.openmobilealliance.org/UseAgreement.html.

Unless this document is clearly designated as an approved specification, this document is a work in process, is not an approved Open Mobile Alliance™ specification, and is subject to revision or removal without notice.

You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. You may not use this document in any other manner without the prior written permission of the Open Mobile Alliance. The Open Mobile Alliance authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services. The Open Mobile Alliance assumes no responsibility for errors or omissions in this document.

Each Open Mobile Alliance member has agreed to use reasonable endeavors to inform the Open Mobile Alliance in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. However, the members do not have an obligation to conduct IPR searches. The declared Essential IPR is publicly available to members and non-members of the Open Mobile Alliance and may be found on the "OMA IPR Declarations" list at http://www.openmobilealliance.org/ipr.html. The Open Mobile Alliance has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions. Defined terms above are set forth in the schedule to the Open Mobile Alliance Application Form.

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR'S REPRESENTED ON THE "OMA IPR DECLARATIONS" LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

Contents

Figures

Tables

1. Scope

This document provides the technical specifications for the message storage functionality of the CPM Enabler. The document covers the storage of Media Objects, CPM Messages, CPM File Transfer Histories, CPM Session Histories and CPM Conversation Histories in the network and the interactions between client and server components to access the network storage. The technical specifications are designed to fulfil the requirements, architecture and system concepts that are described in [OMA-CPM-RD], [OMA-CPM-AD] and [OMA-CPM-SD] respectively.

As such, these technical specifications provide the formal definitions of the CPM-MSG interface that has been identified in [OMA-CPM-AD]. Also, these technical specifications formally define the expected behaviour of the Message Storage Client and Message Storage Server functional components that have been identified in [OMA-CPM-AD].

2. References

2.1 Normative References

[GSMA-VVM] “Visual Voice Mail Interface Specifications”, Version 1.3, Open Mobile Terminal Platform, OMTP URL:http://www.gsmworld.com/documents/

[OMA-CPM-AD] “Converged IP Messaging Architecture”, Open Mobile Alliance™, OMA-AD-CPM-V2_1, URL:http://www.openmobilealliance.org/

[OMA-CPM-Conv] “Converged IP Messaging Conversation Functions”, Open Mobile Alliance™, OMA-TS-CPM_Conversation_Function-V2_1, URL:http://www.openmobilealliance.org/

[OMA-CPM-RD] “Converged IP Messaging Requirements”, Open Mobile Alliance™, OMA-RD-CPM-V2_1, URL:http://www.openmobilealliance.org/

[OMA-CPM-SD] “Converged IP Messaging System Description”, Open Mobile Alliance™, OMA-TS-CPM_System_Description-V2_1, URL:http://www.openmobilealliance.org/

[OMA-EVVM] “Enhanced Visual Voice Mail”, Version 1.0, Open Mobile Alliance™, OMA-ERP-EVVM-V1_0, URL:http://www.openmobilealliance.org/

[OMA-SCRRULES] “SCR Rules and Procedures”, Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[OMA-SEC-CF] “Security Common Functions Requirements”, Open Mobile Alliance, OMA-RD-SEC_CF-V1.0, URL:http//www.openmobilealliance.org/

[OMA-MMS-ENC] "Multimedia Messaging Service", Open Mobile Alliance™, OMA-MMS-ENC-V1_3, URL:http://www.openmobilealliance.org/

[RFC2119] “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119

[RFC2177] “IMAP IDLE command”, B. Leiba, June 1997, URL:http://www.ietf.org/rfc/rfc2177

[RFC2246] “The TLS Protocol Version 1.0”, T. Dierks et al, January 1999, URL:http://www.ietf.org/rfc/rfc2246

[RFC2387] “The MIME Multipart/Related Content-type”, E. Levinson, August 1998, URL:http://www.ietf.org/rfc/rfc2387

[RFC2392] “Content-ID and Message-ID Uniform Resource Locator”, E. Levinson, August 1998, URL:http://www.ietf.org/rfc/rfc2392

[RFC3458] “Message Context for Internet Mail”, E. Burger, January 2003, URL:http://tools.ietf.org/rfc/rfc3458

[RFC3501] “Internet Message Access Protocol - version 4rev1”, M. Crispin, March 2003, URL:http://www.ietf.org/rfc/rfc3501

[RFC3516] “IMAP4 Binary Content Extension”, L. Nerenberg, April 2003, URL:http://www.ietf.org/rfc/rfc3516

[RFC3862] “Common Presence and Instant Messaging (CPIM) : Message Format”, G. Klyne et al, August 2004, URL:http://www.ietf.org/rfc/rfc3862

[RFC4314] “IMAP4 Access Control List (ACL) Extension”, A. Melnikov, December 2005, URL:http://www.ietf.org/rfc/rfc4314

[RFC4315] “Internet Message Access Protocol (IMAP) - UIDPLUS extension”, M. Crispin, December 2005, URL:http://www.ietf.org/rfc/rfc4315

[RFC4467] “Internet Message Access Protocol (IMAP) – URLAUTH Extension”, M. Crispin, May 2006, URL:http://www.ietf.org/rfc/rfc4467

[RFC4575] A Session Initiation Protocol (SIP) Event Package for Conference State, J. Rosenberg, H. Schulzrinne, O. Levin, Ed., August 2006, URL:http://www.ietf.org/rfc/ rfc4575

[RFC5092] “IMAP URL Scheme”, A. Melnikov, Ed. et al, November 2007, URL:http://www.ietf.org/rfc/rfc5092

[RFC5161] “The IMAP ENABLE Extension”, A. Gulbrandsen, Ed. et al, March 2008, URL:http://www.ietf.org/rfc/rfc5161

[RFC5257] “Internet Message Access Protocol - ANNOTATE Extension”, C. Daboo et al, June 2008, URL:http://www.ietf.org/rfc/rfc5257

[RFC5259] “Internet Message Access Protocol -CONVERT extension”, A. Melnikov, Ed. et al, July 2008, URL:http://www.ietf.org/rfc/rfc5259

[RFC5322] “Internet Message Format”, P. Resnick, October 2008. URL:http://www.ietf.org/rfc/rfc5322

[RFC5423] “Internet Message Store Event”, R. Gellens, et al, March 2009, URL:http://www.ietf.org/rfc/rfc5423

[RFC5464] “The IMAP METADATA extension”, C. Daboo, February 2009, URL:http://www.ietf.org/rfc/rfc5464

[RFC5465] “The IMAP NOTIFY Extension”, A. Gulbrandsen, et al, February 2009, URL:http://www.ietf.org/rfc/rfc5465

[RFC5547] “A Session Description Protocol (SDP) Offer/Answer Mechanism for File Transfer” , A. Garcia-Martin, May 2009; URL:http://tools.ietf.org/rfc/rfc5547

[RFC5551] “Lemonade Notifications Architecture’, R. Gellens, August 2009, URL:http://www.ietf.org/rfc/rfc5551

[RFC5788] “IMAP4 Keyword Registry”, A. Melnikov et al, March 2010, URL:http://www.ietf.org/rfc/rfc5788

[RFC5819] “IMAP4 Extension for Returning STATUS Information in Extended LIST”, A. Melnikov et al, March 2010, URL:http://www.ietf.org/rfc/rfc5819

[RFC7162] “IMAP4 Extensions: Quick Flag Changes Resynchronization (CONDSTORE) and Quick Mailbox Resynchronization (QRESYNC) ”, A. Melnikov, May 2014, URL:http://www.ietf.org/rfc/rfc7162

2.2 Informative References

[OMADICT] “Dictionary for OMA Specifications”, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Conference Information Object	See [OMA-CPM-SD].
CPM Address	See [OMA-CPM-RD].
CPM Chat Message	See [OMA-CPM-RD].
CPM Conversation	See [OMA-CPM-RD].
CPM Conversation History	See [OMA-CPM-RD].
CPM Conversation Identity	See [OMA-CPM-SD].
CPM File Transfer	See [OMA-CPM-RD].
CPM File Transfer History	See [OMA-CPM-RD].
CPM Message	See [OMA-CPM-RD].
CPM Participating Function	See [OMA-CPM-AD].
CPM Pre-defined Group	See [OMA-CPM-RD].
CPM Session	See [OMA-CPM-RD].
CPM Session History	See [OMA-CPM-RD].
CPM Session Invitation	See [OMA-CPM-RD].
CPM Standalone Message	See [OMA-CPM-RD].
CPM User	See [OMA-CPM-RD].
Group State Object	An IMAP object in the CPM Message Store that is associated to a CPM Group Session and contains a snapshot of the information related to the CPM Group Session and its Participants.
Media Object	See [OMA-CPM-AD].
Message Storage Client	See [OMA-CPM-AD].
Message Storage Server	See [OMA-CPM-AD].
Participant	See [OMADICT].
Principal	See [OMADICT].
UIDVALIDITY	A 32-bit representation of the creation date and time of a folder
Unique IDentifier	A 32-bit value assigned to each stored object and used with the UID validity value to form a 64-bit value that is unique to a stored object and cannot be referred to any other stored object in the folder or any subsequent folder of the message storage [RFC3501].

3.3 Abbreviations

ACL	See [OMADICT]
CPIM	Common Presence and Instant Messaging
CPM	See [OMADICT]
GSO	Group State Object
IMAP	See [OMADICT]
MIME	See [OMADICT]
OMA	See [OMADICT]
PSK	Pre Shared Key
SASL	Simple Authentication and Security Layer
SIO	session info object
TLS	See [OMADICT]
UID	Unique (Message) Identifier
URL	See [OMADICT]

Note: Abbreviations defined in the OMA Dictionary complements this section.

4. Introduction

The CPM message storage functionality allows the storage of CPM Messages, CPM File Transfer Histories, CPM Session Histories, CPM Conversation Histories, and any potential Media objects either stand-alone or attached to CPM Messages and CPM Session Histories in a network-based storage on behalf of CPM Users.

The CPM message storage functionality authenticates and authorizes CPM Users to being able to retrieve, organize, set permissions, receive event notifications, synchronize with CPM Users device's local message storage and manage (e.g., copy, remove, move etc.,) the storage objects that are stored on it. It also allows CPM Users to search the storage objects with key words.

4.1 CPM Version 1.0

The CPM 1.0 offers:

- Storage objects
  - CPM Standalone Messages, CPM File Transfer Histories, CPM Session Histories and CPM Conversation Histories
  - any potential Media objects either stand-alone or attached to CPM Standalone Messages, CPM File Transfer Histories and CPM Session Histories
- User authentication and authorization mechanisms
- Operations
  - folder operations e.g., create, list, set active folder, move, remove, search
  - stored object operations e.g., store, fetch, copy, remove, preview
  - metadata operations on stored objects e.g., update metadata
  - access rights on stored objects e.g., set, get, delete
- generate references to stand-alone or attached to CPM Standalone Messages and CPM Session History objects
- allows fetch of stored Media objects by reference
- synchronize between network-based storage and device's local storage
- Notifications
  - notifications about changes in stored resources

4.2 CPM Version 2.0

Backward compatibility of version 2.0 with CPM Version 1.0 excludes support of session history object as defined in CPM version 1.0. This has been replaced in CPM version 2.0 by the session history folder and session info object.

4.3 CPM Version 2.1

The CPM 2.1 provides

- Clarification that upon a successful Object Store operation the Message Store Server will return the UID of the stored Object to the Message Store Client.
- A definition of how the Message-Correlator shall be set for SMS and MMS message objects
- A new IMAP extension called XLIST-CHANGEDSINCE which combines the benefits of the LIST-STATUS and CONDSTORE/QRESYNC extensions. This extension enables a CPM Client to request only the LIST of the folders that have changed since last synchronization done by the CPM Client. This allows the CPM Client to subsequently issue a SELECT command only on the folders requiring synchronization.
- A new CPM-defined IMAP metadata flag, "Archived", is defined that indicates the associated Message Object is archived.

4.4 CPM Version 2.2

The CPM 2.2 adds:

- Align with new TS: OMA-TS-MessageStore_Using_RestfulAPI-V1_0 on basic Message Object MIME headers such "cc" and "Bcc"
- Add new "Accept-Content" header to Message Object to support GSMA RCS Universal Profile 2.0 use case
- Add new "yourown" and "role" attributes to Group State Object
- Add new "max-user-count" and "policy" elements to Group State Object
- Add new Conference Information Object

5. Common Procedures

5.1 Authorization and Authentication

The IMAPv4 [RFC3501] protocol is used to access the Message Storage Server. This section defines authentication and authorization mechanisms of IMAPv4 used by the Message Storage Server.

5.1.1 Authentication

CPM's message storage functionality supports the two authentication mechanisms listed below, as defined for IMAPv4 [RFC3501];

1. SASL authentication via the AUTHENTICATE command ([RFC3501]); and
2. Username/password in plain text authentication via the LOGIN command ([RFC3501]).

The username/password for the second authentication mechanism is separately managed by the CPM service. The password may be pre-configured by the CPM system when the CPM user subscribes to the CPM service.

In addition to these authentication mechanisms, TLS/PSK-TLS, as defined in [RFC2246] and [OMA-SEC-CF], is optional and complementary to simple authentication-only SASL mechanisms or deployed clear-text password login commands. In this way, IMAPv4 can be immune to eavesdropping and hijacking attacks. Using TLS/PSK-TLS, the Message Storage Client also can authenticate the Message Storage Server by checking the certificate supplied by the Message Storage Server.

5.1.2 Authorization

The Message Storage Server uses standard IMAP4 [RFC3501] functionality to enforce access to the stored resources, extended with the possibility for CPM Users to define access control lists (ACL) for their own stored resources.

The ACL management operations and related standard rights are defined in IMAPv4 ACL extension [RFC4314].

In addition to that, the Message Storage Server also allows the use of the IMAP4 URLAUTH extension, as defined in [RFC4467] and [RFC5092]. This URLAUTH extension provides a means by which a Message Storage Client can use URLs carrying authorization information to access limited data on the Message Storage Server.

5.2 Storage Folder and Objects

Per the description provided in section 5.5 of the CPM System Description [OMA-CPM-SD], a CPM Message Storage Server MAY contain the following items:

- message object,
- session history folder,
- file transfer history object,
- conversation history folder,
- stand-alone Media Object,
- user folder,
- session info object,
- Group State Object,
- Conference Information Object

In this section, these storage objects are specified in terms of their names and identities, which can be used for various CPM message storage operations in accordance with IMAP4 [RFC3501] and its extensions.

5.2.1 Message Object

A message object, matching the message concept described in [RFC3501], is the base object defined in CPM for a Message Storage Server stored item. This implies an MIME object with header attributes (metadata) defined in table below, particularly Content-Type identifying the type and format of the message.

Common content types include:

- Simple types (e.g. Image/jpg, Text/plain)
- MIME multipart (e.g. multipart/mixed) which contains MIME body parts separated by a boundary string. Each body part has its own type and format as identified by the body-part's Content-Type header (e.g. it could be a simple type or a multipart).
- Common Profile Instant Message format type (i.e. Message/CPIM) which encapsulates an arbitrary MIME message content, as defined in [RFC3862]. The type and format of the encapsulated MIME message is identified by its own Content-Type header (e.g. it could be a simple type or a multipart).

For execution of various IMAP commands, the message object is identified by a 3-component identifier consisting of a folder name, a message identification number associated to the message object and a folder validity value as specified below:

1. The folder name is the name of the folder in the Message Storage Server where the message is stored,
2. The message identification number is either a message sequence number or a 32-bit Unique Identifier (UID), which is specified according to [RFC3501],
3. The folder validity value or the Unique IDentifier validity (UIDVALIDITY) is another 32-bit value as defined in [RFC3501] distinguishing folders of the same name from each other.

The IMDN disposition notifications, sent or received, for any of the CPM Standalone Messages, CPM Chat Messages or CPM File Transfers are stored as message objects in the same Message Store folder as the original message they are associated with.

CPM Standalone Messages, CPM Chat Messages, disposition notifications, SMS Messages and MMS Messages sent or received are stored using the message object definition. The CPM File Transfer object definition is also based on, and extends, the message object.

The message object is a message formatted according to [RFC5322] with the MIME headers and clarifications given in Table 1 below.

Two examples of a chat message objects are shown in Appendix D and Appendix F.

Internet Message Format [RFC5322] header	Internet Message Format Parameter status	Content
From	Mandatory	Set to the address of the initiator of the CPM request or response, or legacy message. For the CPM request or response, it is retrieved from the authenticated originator’s CPM Address in the SIP request, as follows: – Set to the value of the Referred-By header field if it is present in the SIP request; or – Set to the value of the P-Asserted-Identity received in the SIP request or response. – For CPM Chat Messages exchanged in a 1-1 CPM Session it is set to the authenticated CPM Address of the CPM Chat Messages originator, as retrieved from the SIP request or response; – For CPM Chat Messages exchanged within a CPM Group Session, it is set to the address of the originator of the message as retrieved from the CPIM From header. For a legacy message (e.g. SMS, MMS) it is set to the authenticated address of the originating party: – To comply with the MIME From header syntax, a domain part needs to be added to the telephone number. The appended domain part SHALL be a configurable String defined by the service provider, allowing the CPM Message Store Client to determine that the URI is not a real SIP URI, but is rather based on a telephone number only. (Example: +15145551234@phony.com) If the initiator withholds its details, (e.g. use of anonymous URI) then the “anonymous” String SHALL be provided.

To	Mandatory	Set to the address of the recipient of the CPM request or response, or of the legacy message. For the CPM request or response, it is retrieved from the authenticated recipient’s CPM Address in the SIP request or SIP 200 “OK” response to the SIP request, with the following clarifications: – For CPM Chat Messages exchanged within a CPM Group Session, it is set to the address of the recipient of the CPM Chat Message, as retrieved from the CPIM To header of the MSRP message. o In a CPM Chat Message sent by a Participant to the other CPM Group Session Participants, this is the address of the CPM Group Session Identity; o in a CPM Chat Message received by a Participant in the CPM Group Session, it is the address of the Participant. For a legacy message (e.g. SMS, MMS) it is set to the authenticated address of the recipient: – To comply with the MIME To header syntax, a domain part needs to be added to the telephone number. The appended domain part SHALL be a configurable String defined by the service provider, allowing the CPM Message Store Client to determine that the URI is not a real SIP URI, but is rather based on a telephone number only. (Example: +15145551234@phony.com)
CC	Optional	Used only in MMS objects.
BCC	Optional	Used only in MMS objects.
Date	Mandatory	Set to, in order of preference: – the Date of the SIP MESSAGE or SIP INVITE request if available, otherwise, – the date and time when the recording of the CPM request was started.
Subject	Optional	Set only if “Subject” header is set in the SIP request. Note: it is not present in CPM Chat Messages (included in the session info object instead).

P-Asserted-Service	Optional	Includes the CPM Feature Tag present in the SIP request. It SHALL be present in message objects containing: – CPM Standalone Messages, – IMDN notifications received via SIP MESSAGE – File Transfer History objects. Note: it is not present in: – CPM Chat Messages (as it is included in the session infoobject instead) – Group State Object (as it is included in the session info object instead) – Conference Information Object (as it is included in the session info object instead) – legacy messages (containing SMS or MMS).
Conversation-ID	Mandatory	Set to the Conversation-ID of the SIP MESSAGE or SIP INVITE request. NOTE: this is a new field, extending the Internet Message Format in accordance with 3.6.8 of [RFC5322]. The ABNF for this field is described in Appendix C
Contribution-ID	Mandatory	Set to the Contribution-ID of the SIP MESSAGE or SIP INVITE request. NOTE: this is a new field, extending the Internet Message Format in accordance with 3.6.8 of [RFC5322]. The ABNF for this field is described in Appendix C
InReplyTo-Contribution-ID	Optional	Set to the InReplyTo -Contribution-ID of the SIP MESSAGE or SIP INVITE request. NOTE: this is a new field, extending the Internet Message Format in accordance with 3.6.8 of [RFC5322]. The ABNF for this field is described in Appendix C NOTE: it is not present in CPM Chat Messages (included in the session info object instead).
Accept-Contact	Mandatory	Occurrence: 1..* Each occurrence of this MIME header SHALL be set to the value of an Accept-Contact header field that was present in the SIP request associated to this object.

IMDN-Message-ID	Mandatory	For messages received via MSRP, it is set to the imdn.Message-ID header field value described in sect. 6.3 of [RFC5438] when available in the CPM request (e.g., within a CPM Standalone Large Message Mode, or CPM Session, or CPM File Transfer). For SMS Messages, MMS Messages, CPM Pager Mode Messages without an imdn. Message-ID, session info objects, Group State Objects and Conference Info Objects a unique value needs to be generated and assigned by the storing entity (e.g. CPM Participating Function, CPM Message Store Client), with similar definition and uniqueness requirements as described in sect. 6.3 of [RFC5438]. NOTE: this is a new field, extending the Internet Message Format in accordance with 3.6.8 of [RFC5322]. The ABNF for this field is described in Appendix C
Message-Correlator	Optional	For correlation of legacy messages such as SMS and MMS. For SMS message objects the value SHALL contain a fragment or the full text contained in the user data of the short message, depending on the length of the message. For MMS message objects the value SHALL be set to the Message-ID field value as defined in [OMA-MMS-ENC]. The type of value of the Message-Correlator SHALL correspond to the context indicated in the value of the Message-Context header. NOTE: this is a new field, extending the Internet Message Format in accordance with 3.6.8 of [RFC5322]. The ABNF for this field is described in Appendix C.
Message-Context	Optional	This MIME header field is defined in [RFC3458]. It is only present when a CPM Message was sent or delivered via legacy messaging such as SMS or MMS. It SHALL be present if a Message-Correlator header is present. In this case, this MIME header is set to the value “pager-message” for SMS and to “multimedia-message” for MMS. For received messages, the value is set when the message is delivered to a primary device: o via SMS , the value is “pager-message”; o via MMS, the value is set to “multimedia-message”. For originated messages, when the legacy message was sent: o via SMS, the value is set to “pager-message”; o via MMS, the value is set to “multimedia-message”.
Content-Type	Mandatory	Set to the following value: “Message/CPIM” for objects including a CPM Standalone Message, a legacy message or an IMDN message. Other Content-Types than “Message/CPIM” MAY be stored inside the “Message/CPIM” wrapper, some examples are listed above in this section.

Message-body	Mandatory	Contains the CPM Message content as received. NOTE: for message objects storing private messages exchanged during a CPM Group Session, if the actual recipient list for each message is recorded, this MAY be recorded in the CPIM "To" headers of the MSRP messages.

Table 1: MIME headers for the Message Object.

5.2.2 File Transfer History Object

The File Transfer History object is a special type of a message object. The definition and identity specification of the message object in section 5.2.1. "*Message Object*" SHALL be applicable to the File Transfer History object, with the differences described in Table 2 below.

The content inserted in the MIME headers of this message is retrieved from the CPM File Transfer invitation.

An example of a File Transfer History object is shown in Appendix E.

The MIME headers defined for the message object apply also for the CPM File Transfer History object, with the following differences for the MIME header fields described in Table 2 below:

Internet Message Format [RFC5322] header	Internet Message Format Parameter status	Content
P-Asserted-Service	Mandatory	Mandatory MIME header field for a File Transfer History object.
Message-Context	Not applicable	Not present for CPM File Transfer History objects.
Message-Correlator	Not applicable	Not present for CPM File Transfer History objects.
Cc	Not applicable	
Bcc	Not applicable	
Content-Type	Mandatory	Set to the following value : multipart/related;boundary=cpm; type="Application/X-CPM-File-Transfer" NOTE: the new Application/X-CPM File-Transfer content-type is defined in section 5.2.2.1.
Message-body	Mandatory	Contains a list of MIME entities separated by the "cpm" boundary. The first MIME entity is the "root", as defined in [RFC2387] and its content-type is "Application/X-CPM-File-Transfer". The other MIME entities are the file/s that were exchanged during the CPM File Transfer request.

Table 2: CPM File Transfer History object MIME header fields' differences with the message object

5.2.2.1 Application/X-CPM-File-Transfer Content Type Definition

This new MIME type is used to store metadata about the CPM File Transfer operation. The elements defined below are mandatory unless explicitly stated otherwise.

The body of a message with the content-type header set to Application/X-CPM-File-Transfer SHALL be formatted as follows:

One <file-transfer> root element including:

> **<file-transfer-type>** this element indicates the type of CPM File Transfer. It SHALL be set to "Ad-Hoc", "Pre-Defined" or "1-1" **</file-transfer-type>**.
>
> **<invited-participants>** this element is optional. It SHALL be present only for Group CPM File Transfer and it SHALL NOT be present for 1-1 CPM File Transfer. It contains the list of addresses of the invited recipients of the file, separated by a semi-column, or the address of the Pre-defined Group**</invited-participants>**
>
> **<imdn>** this element is optional. It SHALL be present if it is found in the SIP request and it SHALL NOT be present otherwise. It contains the CPIM headers including the information on the IMDN(s) requested for the CPM File Transfer operation. **</imdn>**
>
> **<sdp>** SDP parameters associated with the corresponding stored files (e.g., file name) **</sdp>**
>
> If the SDP information includes thumbnail information (i.e. a=file-icon parameter), then it includes a cid value of the thumbnail content as described in [RFC5547]. In such case, the thumbnail binary content will be also included, MIME wrapped with the Content-ID MIME header field set to the value of the cid found in the a=file-icon SDP attribute.
>
> Next, for each file, metadata information MAY be stored within a <file-object> element as follows:
>
> **<file-object>**
>
> > **<cid>cid:** set to the Content-ID, as defined in [RFC2392], of the corresponding stored file **</cid>**
>
> **</file-object>**

5.2.3 Session Info Object

A session info object contains only the session information of the recorded CPM Session. The session info object is only relevant to the session history folder, or to the 1-1 conversation folder it belongs to and SHALL NOT be moved to another folder or be removed from its original folder until all the associated CPM Chat Messages, their disposition notifications and Group State Object are moved or removed.

The definition and identity specification of the message object SHALL be applicable to the session info object, which consists of a UID, UIDVALIDITY and their values. The session info object is a message object as defined in section 5.2.1 "*Message Object*", therefore all the MIME headers defined for the message object apply also for the session info object, with the additional clarifications and differences given in Table below.

The content inserted in the headers of this message is retrieved from the CPM Session Invitation.

The body of this object SHALL contain metadata about the CPM Session History and is formatted according to the table below.

An example of a session info object is shown in Appendix D.

Internet Message Format [RFC5322] header	**Internet Message Format Parameter status**	**Content**
P-Asserted-Service	Mandatory	Mandatory MIME header field for a session info object.

IMDN-Message-ID	Mandatory	For session info objects and Group State Objects a unique value needs to be generated and assigned by the storing entity (e.g. CPM Participating Function, CPM Message Store Client), with similar definition and uniqueness requirements as described in section 6.3 of [RFC5438].
Message-Correlator	Not applicable	Not present for CPM session info objects.
Message-Context	Not applicable	Not present for CPM session info objects.
Cc	Not applicable	
Bcc	Not applicable	
Content type	Mandatory	Set to the following value : Application/X-CPM-Session NOTE: the new "Application/X-CPM-Session" content-type is defined in section 5.2.3.1.
Message body	Mandatory	Contains the session content as defined in section 5.2.3.1.

Table 3: CPM session info object MIME header fields' differences with the message object

5.2.3.1 Application/X-CPM-Session Content Type Definition

This new MIME type is used to store metadata about the CPM Session. The elements defined below are mandatory unless explicitly stated otherwise.

The body of a message with the content-type header set to Application/X-CPM-Session SHALL be formatted as follows:

One <session> as the XML root element containing:

<session-type> this element indicates the type of CPM Session. It may be set to "Group" or "1-1" **</session-type>**

<session-replaces> this element is optional. If present, it SHALL contain the Contribution-ID of the CPM Session being replaced**</session-replaces>**

<invited-participants> this element is optional. It SHALL BE present for a CPM Group Session and SHALL NOT be present for a CPM 1-1 Session. It contains the list of addresses, separated by a semi-column, of the invited participants to an Ad-hoc CPM Group Session (i.e. not including the originator), or the address of the Pre-defined Group**</invited-participants>**

5.2.4 Group State Object

The Group State Object is a message object as defined in section 5.2.1, with an XML content defined in this section. All the MIME headers defined for the message object apply also for the Group State Object, with the additional clarifications and differences given in Table below.

The content inserted in the CPM MIME headers of this message is retrieved from the CPM Session Invitation.

The body of the Group State Object is an XML object with XML Content Type "application/group-state-object+xml" containing the valid CPM Group Session Identity for the Group Chat, as well as the set of active participants at the time indicated by the timestamp attribute.

Internet Message Format [RFC5322] header	Internet Message Format Parameter status	Content
P-Asserted-Service	Not applicable	Not present for CPM GSO since GSOs always belong to the CPM Group Sessions.
Accept-Contact	Not applicable	Not present for CPM GSO as it is always present in the session info object and all objects (including associated IMDNs) of a specific CPM Group Session.
IMDN-Message-ID	Mandatory	For session info objects and Group State Objects a unique value needs to be generated and assigned by the storing entity (e.g. CPM Participating Function, CPM Message Store Client), with similar definition and uniqueness requirements as described in section 6.3 of [RFC5438].
Message-Correlator	Not applicable	Not present for CPM Group State Objects.
Message-Context	Not applicable	Not present for CPM Group State Objects.
Cc	Not applicable	
Bcc	Not applicable	
Content type	Mandatory	Set to the following value : application/group-state-object+xml If an icon file body is contained in the CPM Group State Object it is set to the following value: multipart/related
Message body	Mandatory	Contains the Group State Object content as defined below in this section. In addition an icon file body may be stored.

Table 4: GSO MIME header fields' differences with the message object

The body of the group state object is an XML object with XML Content Type "application/group-state-object+xml" containing the valid CPM Group Session Identity for the Group Chat, as well as the set of active participants at the time indicated by the timestamp attribute.

The format of the body of the Group State object includes the following elements and attributes:

1) SHALL include one <groupstate> root element containing:
   a) SHALL include one "lastfocussessionid" attribute, containing the CPM Group Session Identity, of type "anyURI";
   b) SHALL include one "timestamp" attribute, of type "dateTime";
   c) SHALL include one "group-type" attribute, containing the type of CPM Group Session, of type "String". The possible enumeration values are "Closed" and "Open". The value "Closed" is used for CPM Closed Group Sessions as per definition in [OMA-CPM-RD];

d) MAY include one “iw-number” attribute, populated when delivery was done via interworking, and contains the unique interworking number used for CPM Chat Messages delivery via SMS or MMS, to the CPM User’s Client residing on the Primary Device. The attribute is of type “anyURI” and SHALL contain the number as a TEL URI;

e) MAY include one <status> element indicating the CPM User’s status in the CPM Group Session. The status can take the following attributes:

a) MAY have one <removed> element, indicating that the CPM User has been removed from the CPM Group Session. The <removed> element MAY include a <participant> element indicating the Participant that initiated the event that caused the creation of the Group State Object;

f) SHALL include one or more <participant> child elements containing:

i) SHALL include “name” attribute of the current participant in the CPM Session, of type “String”. If none is available for a specific participant, the empty value SHALL be used instead;

ii) SHALL include “comm-addr” attribute containing the communication address of the participant, of type “anyURI”.

iii) MAY include "yourown" attribute, when set to "true", indicating which participant element belongs to the CPM User to which the Group State Object is sent.

iv) MAY include a "role" attribute indicating the role of the CPM User as defined in Annex Q.1.2 " CPM User Roles" of [OMA-CPM-Conv].

g) MAY include one <subject> element, with the following children elements:

a) One <text> element including the subject data; and

b) One <participant> element containing the Participant that has made the change included in the <text> element;

c) One <timestamp> element of type “dateTime”, including the date and time when the change was made, as received by the CPM Participating Function in the SIP NOTIFY in the <timestamp> element of <conference-info>;

h) MAY include one <icon> element containing the following children elements:

a) One element <source> that contains either:

(1) one <icon-uri> element containing a URI pointing to the icon image; or

(2) one <file-info> element containing the content-ID of the body part of the Group State Object containing the icon image;

b) One <participant> element containing the Participant that has made the change included in the <text> element;

c) One <timestamp> element of type “dateTime”, including the date and time when the change was made, as received by the CPM Participating Function in the SIP NOTIFY in the <timestamp> element of <conference-info>.

i) MAY include one <max-user-count> element including the maximum of users that can be invited.

j) MAY include one <policy> element including the CPM Controlling Function Policy as defined in Annex Q.1 "*CPM Controlling Function Policy*" of [OMA-CPM-Conv].

The XML schema of the Group State Object is defined in Table 5 below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
elementFormDefault="qualified">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2009/01/xml.xsd"/>

  <xs:element name="groupstate">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="participant" type="participant-type" maxOccurs="unbounded"/>
     <xs:element name="status" minOccurs="0">
      <xs:complexType>
       <xs:choice>
        <xs:element name="removed">
         <xs:complexType>
          <xs:sequence>
             <xs:element name="participant" type="participant-type" minOccurs="0"/>
             <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
          </xs:sequence>
         </xs:complexType>
```

```
            </xs:element>
            <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
          </xs:choice>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="subject" type="subject-type" minOccurs="0"/>
      <xs:element name="icon" type="icon-type" minOccurs="0"/>
      <xs:attribute name="lastfocussessionid" type="xs:string" use="required"/>
      <xs:attribute name="iw-number" type="xs:anyURI"/>
      <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
      <xs:attribute name="group-type" type="groupType" use="required"/>
      <xs:anyAttribute processContents="lax"/>
    </xs:sequence>
    <xs:element name="max-user-count" type="xs:unsignedInt" minOccurs="0"/>
    <xs:element name="policy" type="xs:string" minOccurs="0"/>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xs:complexType>
</xs:element>

<xs:simpleType name="groupType">
  <xs:restriction base="xs:normalizedString">
    <xs:enumeration value="Closed"/>
    <xs:enumeration value="Open"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="icon-type">
  <xs:complexType>
  <xs:sequence>
    <xs:choice>
      <xs:element name="icon-uri" type="xs:anyURI"/>
      <xs:element name="file-info" type="xs:string"/>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xs:choice>
    <xs:element name="participant" type="participant-type" minOccurs="0"/>
    <xs:element name="timestamp" type="dateTime" minOccurs="0"/>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xs:sequence>
  </xs:complexType>
  <xs:any namespace="##other" processContents="lax"/>
</xs:element>

<xs:element name="subject-type">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="text" type="xs:string"/>
            <xs:element name="participant" type="participant-type" minOccurs="0"/>
            <xs:element name="timestamp" type="dateTime" minOccurs="0"/>
            <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
</xs:element>

<xs:element name="participant-type">
    <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="comm-addr" type="xs:anyURI" use="required"/>
            <xs:attribute name="yourown" type="xs:boolean" default="false"/>
            <xs:attribute name="role" type="xs:string" use="required"/>
            <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    </xs:element>


</xs:schema>
```

Table 5 Group State Object – XML schema

An example of the Group State Object body is shown in Appendix G.

5.2.5 Stand-alone Media Object

The standalone Media Object is realized by the message concept of IMAP4, whereby the Media Object is wrapped into a MIME formatted object in order to fit into an IMAP4 message. Other than the formatting of the contents, the standalone Media Object is the same as the message object defined in section 5.2.1 "*Message Object*", including the definition and naming specification of the message object. The CPM MIME headers defined for the message object are not applicable to the standalone Media Objects. Standalone Media Objects are further stored in the conversation folder.

5.2.6 Conference Information Object

The Conference Information Object is a message object as defined in section 5.2.1, as specified in Appendix X of [OMA-CPM-Conv]. All the MIME headers defined for the message object apply also for the Conference Information Object, with the additional clarifications and differences given in Table below.

The content inserted in the CPM MIME headers of this message is retrieved from the CPM Session Invitation.

The body of the Conference Information Object is an XML object with XML Content Types defined in [RFC5438] containing the conference information.

Internet Message Format [RFC5322] header	Internet Message Format Parameter status	Content
P-Asserted-Service	Not applicable	Not present for the conference information, since it belongs always to the CPM Group Sessions.
IMDN-Message-ID	Mandatory	For a message with a Conference Information Object a unique value needs to be generated and assigned by the storing entity (e.g. CPM Participating Function, CPM Message Store Client), with similar definition and uniqueness requirements as described in section 6.3 of [RFC5438].
Message-Correlator	Not applicable	Not present for Conference Information Objects.
Message-Context	Not applicable	Not present for Conference Information Objects.
Content type	Mandatory	Content types value for the Conference Information Object is application/conference-info+xml If an icon file body is contained in the CPM Group State Object it is set to the following value: multipart/related

Message body	Mandatory	Contains the Conference Information Object content as defined in [RFC 4575] in accordance with the rules, procedures and extensions defined in [OMA-CPM-Conv]. In addition, an icon file body may be stored.

Table 6: Conference Information Object MIME header fields' differences with the message object

5.2.7 Conversation History Folder

The conversation history folder is realized via the mailbox concept of IMAP4. A conversation history folder stores all items related to a single CPM Conversation as objects (e.g. message objects and file transfer history objects) and sub-folders (i.e. session history folders for CPM group Session). The name of the conversation history folder is set to the CPM Conversation Identity used in that CPM Conversation.

5.2.8 Session History Folder

The session history folder is viewed as a special kind of sub-folder and is realized via the mailbox concept of IMAP4. The name of a session history folder SHALL be the Contribution-ID of the corresponding CPM Group Session. For more information, please refer to the storage representation examples in Appendix I.2.

A session history folder SHALL contain:

- One session info object, and
- Zero or more message object(s), and/or
- Zero or more file transfer object(s), and/or
- Zero or more stand-alone Media Object(s), and/or
- Zero or more group state objects

The name of the session history folder MUST be unique within the scope of the parent Conversation History Object.

5.2.9 User Folder

The user folder is a Message Storage folder realized by the mailbox concept of IMAP4, described in [RFC3501]. The user folder is identified by the name given to it. The CPM user folder aligns with the rules and procedures for names of the mailbox concept of IMAP4, as described in [RFC3501]. For additional information, see section 5.5.1.1 of the CPM System Description [OMA-CPM-SD].

5.2.10 Non-CPM Folders

The CPM Message Store SHOULD allow creation of folders by other enablers, services or applications in the non-CPM folders.

Such specific non-CPM folders SHALL NOT be allowed under the /Default folder.

Each enabler, service or application SHOULD define their own folder structure under their own top-level dedicated folder that is assigned a unique folder name, e.g. a /Voicemail folder. The registration of the non-CPM top-level folder names defined in industry specifications SHALL be done in the OMA Registry at http://technical.openmobilealliance.org/Technical/technical-information/omna/oma-cpm-folder-directory to secure uniqueness amongst various enablers, services or applications.

Such non-CPM folders SHOULD be accessible also by external entities (e.g. voicemail server for the example above), to allow direct read/write application specific data using IMAP-based CPM-MSG.

5.2.10.1 Non-CPM Objects

The CPM Message Store allows storage of objects by other enablers, services or applications.

An Enabler, service or application SHOULD specify their own object format (mandate specific MIME headers and define the content such as e.g., for a voicemail object) and SHOULD register the new objects and a reference to their specification in the OMNA Registry at http://technical.openmobilealliance.org/Technical/technical-information/omna/oma-cpm-folder-directory.

5.3 Identification of Storage Objects

The combination of a folder name, its stored object’s UID, and the folder’s UIDVALIDITY MUST permanently and persistently refer to one and only one stored object in a Message Storage Server. In particular, the internal date, size, envelope, body structure, and message texts (HEADER, TEXT and all BODY fetch data items) MUST never change according to [RFC3501]. This requirement does not include message numbers, nor does it include object attributes that can be set by a STORE command (e.g., FLAGS).

Associated with any folder object, there is a next UID value, which is the predicted value that will be assigned to a new storage object in that folder. Provided by the CPM Message Storage Server, this next UID is intended to provide a means for the CPM Message Storage Client to determine whether or not any messages have been delivered to the folder since the previous time it checked this value.

NOTE: The “next UID” value of a folder changes whenever a new object is stored in that folder.

5.4 Notifications

If the Message Storage Server supports IMAP NOTIFY or IMAP IDLE, the Message Storage Server SHALL notify the Message Storage Client of the changes in the stored resources either solicited as a result of a client request or unsolicited and unilaterally according to the IMAP extension for the general notification model in [RFC5551], [RFC5465] and [RFC2177]. NOTE: Per [RFC2177], the IDLE command provides a way for the client to go into a mode where the Message Storage Server pushes its notifications about the server events in selected folders.

When IMAP notifications are supported the data content of an IMAP notification may include a wide range of information for example:

1. Folder size and stored objects status updates, e.g., addition and/or removal of messages to prevent synchronization errors,
2. “Next UID” for the changes to the storage requested by a third entity, e.g., CPM PF,
3. Stored objects’ attribute flags, e.g., “\Recent” flag for recently arrived objects in a folder.

The changes in the Message Storage Server’s resources MAY include one or more of the following specific operations per section 5 “Event Types” of [RFC5465]:

1. message addition and deletion
2. Message flags, e.g., read, clear
3. Folder management, e.g., create, delete, rename.

5.5 Metadata Structure

CPM’s message storage functionality supports a metadata model that consists of three distinct parts:

1. A set of metadata flags (i.e. IMAP system flags and keywords) that are associated with message objects, file transfer history objects and standalone Media Objects. These flags indicate additional state information about the stored object, and
2. A set of metadata annotations that can be associated with folder objects (i.e. mailboxes) and message objects (i.e. messages) stored in the Message Storage Server. These annotations provide system or user-defined information that the system or the user associates with these stored objects.
3. A set of metadata annotations that can be associated with the server (i.e. server annotations). These annotations provide system- or user-defined information that the system or the user associates with the server rather than individual objects.

The Message Storage Client and the Message Storage Server SHALL at least support metadata flags, folder metadata annotations and server annotations, and, in addition, MAY support message metadata annotations.

With respect to the metadata flags, the Message Storage Client and Message Storage Server SHALL support at least the following flags defined in [RFC3501], [RFC5788]:

- \Seen (message has been read),
- \Answered (message has been answered),
- \Flagged (message is "flagged" for urgent and/or special attention),
- \Deleted (message is "deleted" for removal by later EXPUNGE),

- \Draft (message has not completed composition (marked as a draft)),
- \Recent (message is "recently" arrived in this mailbox),
- $Forwarded (message has been forwarded).

and SHOULD support the following flag defined in Appendix L:

- Archived (message is archived).

NOTE: All flag assignments and operations SHALL be handled according to the procedures specified in [RFC3501].

With respect to the server metadata and folder metadata annotations, the Message Storage Client and the Message Storage Server SHALL support the structure defined in [RFC5464] for metadata annotations as well as the METADATA entries defined in 0 for folder objects.

With respect to the message metadata annotations, if supported, the Message Storage Client and the Message Storage Server SHALL support the structure defined in [RFC5257] for message objects, for file transfer history objects and for standalone Media Objects.

6. Procedures at Message Storage Client

The Message Storage Client is a functional component of the CPM enabler, which allows the CPM User to view and manage (store, fetch, delete etc.) the resources stored in the Message Storage Server. In addition to that the Message Storage Client notifies the CPM User of any changes to the stored resources in the Message Storage Server (e.g. new message arrived, message got read on another client).

The Message Storage Client SHALL act as an IMAP4 client as defined in [RFC3501]. In addition to that, the Message Storage Client SHALL support:

- the “UIDPLUS” IMAP4 extension as defined in [RFC4315],
- the “LIST-STATUS” IMAP4 extension as defined in [RFC5819], and

SHOULD support:

- the VANISHED UID FETCH Modifier as defined in [RFC7162], and
- the “XLIST-CHANGEDSINCE” IMAP4 extension as defined in Appendix J,

MAY support:

- the “ACL” IMAP4 extension as defined in [RFC4314],
- the “URLAUTH” IMAP4 extension as defined in [RFC4467],
- the “CONDSTORE” and the “QRESYNC” IMAP4 extensions as defined in [RFC7162],
- the “ENABLE” IMAP4 extension as defined in [RFC5161],
- the “METADATA” IMAP4 extension as defined in [RFC5464],
- the “NOTIFY” IMAP extension as defined in [RFC5465],
- the “ANNOTATE” IMAP4 extension as defined in [RFC5257] and/or the “CONVERT” IMAP4 extension as defined in [RFC5259], and
- the “IDLE” IMAP4 extension as defined in [RFC2177].

6.1 General Operations

6.1.1 Authenticate Operation

When the CPM Client needs to authenticate with the Message Storage Server, it SHALL either use the SASL method or the plain-text username/password method.

When the Message Storage Client wants to use the SASL method, the Message Storage Client SHALL send to the Message Storage Server an AUTHENTICATE request as defined in [RFC3501] with the authentication mechanism that it wants to use and then SHALL complete the authentication process as defined in [RFC3501].

When the Message Storage Client wants to use the plain-test username/password method, the Message Storage Client SHALL send to the Message Storage Server a LOGIN request as defined in [RFC3501] with the username and password associated with the Message Storage Client’s CPM User.

The CPM Client SHALL set up a TLS session, prior to authenticating with the Message Storage Server. In order to do so, the Message Storage Client SHALL directly attempt the TLS session setup without sending to the Message Storage Server a STARTTLS request as defined in [RFC3501]. The Message Storage Server SHALL support either TLS or PSK-TLS as defined in [RFC2246] and [OMA-SEC-CF] for the TLS protocol negotiations.

A subset, or all of, the security and/or authentication functions of the Message Storage Server MAY be delegated to another network entity (e.g. a proxy). In such case, that network entity SHALL follow the security and/or authentication procedures defined for the Message Storage Server in this document.

6.1.2 Set Active Folder Operation

When a Message Storage Client needs to set a particular folder as the active folder, the Message Storage Client SHALL send to the Message Storage Server a SELECT request as defined in [RFC3501] with the folder name of the folder that is to be set as the active folder.

6.2 Access Control List Operations

6.2.1 Set Access Control List

When a Message Storage Client needs to set the access rights for another Principal on one of the folders of its associated CPM User, the Message Storage Client SHALL send to the Message Storage Server a SETACL request as defined in [RFC4314] with the folder name, the granted access rights and the identifier of the Principal to which access is given.

6.2.2 Get Access Control List

When a Message Storage Client needs to get the access control list on one of the folders of its associated CPM User, the Message Storage Client SHALL send to the Message Storage Server a GETACL request as defined in [RFC4314] with the folder name.

6.2.3 Delete Access Control List

When a Message Storage Client needs to delete the access rights for another Principal on one of the folders of its associated CPM User, the Message Storage Client SHALL send to the Message Storage Server a DELETEACL request as defined in [RFC4314] with the folder name and the identifier of the Principal to which access is given.

6.2.4 Access Rights Retrieval Operations

When a Message Storage Client needs to retrieve the access rights for another Principal on one of the folders of its associated CPM User, the Message Storage Client SHALL send to the Message Storage Server a LISTRIGHTS request as defined in [RFC4314] with a folder name and the identifier of the Principal whose access rights are to be retrieved.

6.3 Message and History Operations

6.3.1 Object Store Operation

When a Message Storage Client needs to store an object (e.g. a message object, a file transfer history object, a standalone Media Object) into a folder on the Message Storage Server, the Message Storage Client SHALL send to the Message Storage Server an APPEND request as defined in [RFC3501] and [RFC4315] including the name of the folder and the data of the object. The Message Storage Client MAY include an initial set of metadata flags in the APPEND request towards the Message Storage Server, as defined in [RFC3501] and [RFC4315]. The Message Storage Client also MAY include a set of metadata annotations in the APPEND request towards the Message Storage Server, as defined in [RFC5257]. Upon successfully storing the object into message store the Message Storage Client SHALL receive the UID assigned to that object from the Message Store Server.

NOTE: The set of metadata flags and metadata annotations associated with the stored object can be changed later using the metadata update operation defined in section 6.6.1 "*Metadata Update Operation*".

6.3.2 Object Fetch Operation

When a Message Storage Client needs to fetch a message object, file transfer history object or stand-alone Media Object from the active folder on the Message Storage Server, the Message Storage Client SHALL send to the Message Storage Server a FETCH or a UID FETCH request as defined in [RFC3501] with the UID(s) pointing to a stored object(s) in Message Storage Server.

NOTE: The Message Storage Client may specify that only specific parts of a message object, file transfer history object or standalone Media Object are to be fetched, as defined in [RFC3501].

6.3.3 Object Preview Fetch Operation

When a Message Storage Client needs to fetch a preview of a message object, file transfer history object or standalone Media Object from the active folder on the Message Storage Server, the Message Storage Client SHALL send to the Message

Storage Server a CONVERT or a UID CONVERT request as defined in [RFC5259] with the UID pointing to the stored object on the Message Storage Server to be previewed.

NOTE 1: The Message Storage Client may specify that only a preview of specific parts of a message object, file transfer history object or standalone Media Object is to be fetched, as defined in [RFC3501] and [RFC5259].

NOTE 2: The CONVERT and UID CONVERT commands can be used to transcode the media type of a MIME part into another media type and/or into the same media type with different encoding parameters.

NOTE 3: A “Preview Fetch” operation MAY involve a server-side content adaptation in response to the Message Storage Client’s request for the stored object in a compacted or digested form rather than in its original full size and shape.

NOTE 4: Conversions only affect what is sent to the Message Storage Client; the original data in the Message Storage Server SHALL NOT be converted.

6.3.4 Object Copy Operation

When a Message Storage Client needs to copy an existing message object, an existing file transfer history object or an existing standalone Media Object in the Message Storage Server, the Message Storage Client SHALL send to the Message Storage Server a COPY request as defined in [RFC3501] with the UID pointing to an object to be copied from the Message Storage Server and specifying the destination folder. All associated IMDNs SHALL also be copied along with the message object or the file transfer history object.

6.3.5 Object Remove Operation

When a Message Storage Client needs to remove a stored message object, a file transfer history object, a Group State Object or a stored standalone Media Object, the Message Storage Client SHALL send to the Message Storage Server a STORE request as defined in [RFC3501] with the UID pointing to the stored object to update the flag list associated with the object’s data and setting the “\Deleted” flag. The Message Storage Client SHALL NOT remove a stored session info object unless it is done in the context of the entire session history folder itself being (re)moved.

After setting the “\Deleted” flag, the Message Storage Client SHALL send to the Message Storage Server an EXPUNGE request as defined in [RFC3501] in order to permanently remove the message(s) that have been identified for removal from the list of objects with the “\Deleted” flag set. All associated IMDNs SHALL also be deleted along with the message object or the file transfer history object.

NOTE: The Message Storage Client can use the EXPUNGE request to permanently remove multiple messages, i.e. there may be multiple STORE commands to set the “Deleted” flag before an EXPUNGE command is executed.

6.4 Folder Operations

6.4.1 Folder Create Operation

When a Message Storage Client needs to create a new folder, the Message Storage Client SHALL send to the Message Storage Server a CREATE request as defined in [RFC3501] including the name of the folder.

6.4.2 List Folder Operation

When a Message Storage Client needs to find out what folders exist in the CPM Message Store for that CPM User, the Message Storage Client SHALL send to the Message Storage Server a LIST request as defined in [RFC3501].

After initial synchronization, the Message Storage Client SHOULD make use of the LIST-STATUS operation as defined in [RFC5819] for any subsequent folder synchronization. A Message Storage Client MAY choose to use separate STATUS operations on every folder instead of LIST-STATUS, however this is not as efficient, hence LIST-STATUS approach is highly recommended. A CPM Client supporting [RFC7162] SHOULD make use of the XLIST-CHANGEDSINCE operation as defined in Appendix J instead of, or in conjunction with, LIST-STATUS in order to minimize the set of mailboxes returned by the server during each synchronization.

When a Message Storage Client needs to list the contents of the currently selected folder, the Message Storage Client SHALL send to the Message Storage Server a FETCH request as defined in [RFC3501].

6.4.3 Folder Move Operation

NOTE: The folder move operation is also used for renaming folders.

When a Message Storage Client needs to rename a folder or to move a folder, the Message Storage Client SHALL send to the Message Storage Server a RENAME request as defined in [RFC3501] including the old name and the new name of the folder to be renamed or moved.

6.4.4 Folder Remove Operation

When a Message Storage Client needs to delete a folder, the Message Storage Client SHALL send to the Message Storage Server a DELETE request as defined in [RFC3501] including the name of the folder that is to be deleted.

6.4.5 Folder Search Operation

When a CPM Client needs to search in the active folder on the Message Storage Server, the Message Storage Client SHALL send to the Message Storage Server SEARCH request as defined in [RFC3501] including one or more search key data.

6.5 Reference Operations

6.5.1 Generate Reference Operation

When a Message Storage Client needs to generate a reference for (part of) a message object, a file transfer history object or a stand-alone Media Object, the Message Storage Client SHALL send to the Message Storage Server a GENURLAUTH request as defined in [RFC4467] including an IMAP URL (as per [RFC5092] and [RFC4467]) pointing to the (part of) the object for which a reference needs to be created.

6.5.2 Fetch by Reference Operation

When a Message Storage Client needs to fetch the (part of) a message object, a file transfer history object or a stand-alone Media Object on the basis of the reference, the Message Storage Client SHALL send to the Message Storage Server a URLFETCH request as defined in [RFC4467].

6.6 Metadata Management Operations

The Message Storage Client SHALL support:

- updating the metadata flags defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*,
- fetching the metadata flags defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*,
- updating server metadata annotations defined for IMAP4 mailboxes in section 5.5 *"Metadata Structure"*.
- fetching server metadata annotations defined for IMAP4 mailboxes in section 5.5 *"Metadata Structure"*.

The Message Storage Client MAY support:

- updating message and folder metadata annotations defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*.
- fetching message and folder metadata annotations defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*.

6.6.1 Metadata Update Operation

When a Message Storage Client needs to update the metadata flags of a message object, the Message Storage Client SHALL send to the Message Storage Server a STORE request as defined in [RFC3501] including the UID of the message object and the changes to the flags (e.g., \Seen, \Deleted, etc.).

When a Message Storage Client needs to update message metadata annotations, if supported, the Message Storage Client SHALL send to the Message Storage Server a STORE request as defined in [RFC3501] including the UID of the message object and the 'ANNOTATION' data item as defined in [RFC5257].

When a Message Storage Client needs to update server or folder metadata, the Message Storage Client SHALL send to the Message Storage Server a SETMETADATA request as defined in [RFC5464] including:

- the name of the folder whose metadata is to be updated,
- an entry specifier and value pair (i.e. the entry specifier and the value corresponding to the entry specifier).

NOTE: The name of the folder SHALL be an empty string (i.e. "") to update server annotations.

The SETMETADATA request allows a requestor to include multiple update in a request. To reduce network overhead and delays, it is RECOMMENDED that a requestor updates all annotations at once by including all entry specifier and value pairs in a single request.

6.6.2 Metadata Fetch Operation

When a Message Storage Client needs to retrieve the metadata flags of a message object, the Message Storage Client SHALL send to the Message Storage Server a FETCH request as defined in [RFC3501] including the UID of the message object and the ‘FLAGS’ data item name.

When a Message Storage Client needs to retrieve the metadata flags of a message object, if supported, the Message Storage Client SHALL send to the Message Storage Server a FETCH request as defined in [RFC3501] including the UID of the message object and the ‘ANNOTATION’ data item name, as defined in [RFC5257].

When a Message Storage Client needs to retrieve server or folder metadata annotations, the Message Storage Client SHALL send to the Message Storage Server a GETMETADATA request as defined in [RFC5464] including:

- the name or the folder whose metadata is requested,
- the entry specifier.

NOTE: The name of the folder SHALL be an empty string (i.e. "") to retrieve server annotations.

The GETMETADATA request allows a requestor to extend the list of entry specifiers in a fetch operation using the DEPTH command option. To reduce network overhead and delays, it is RECOMMENDED that a requestor retrieves annotations in batch(es), using the DEPTH command option. Therefore, a requestor’s request SHOULD include:

- the DEPTH command option with the value “infinity”,
- the entry specifier as a higher level specifier, e.g. “/shared/OMNA”, “/shared/OMNA/OMACPM20”, etc.

6.7 Synchronization

[OMA-CPM-SD] gives a description of the synchronization process between the Message Storage Client and the Message Storage Server. This process only consists of operations described above.

While executing these operations, the Message Storage Client SHALL support and use the IMAP4 extensions described in [RFC5819] and [RFC4315], and MAY use the IMAP4 extensions described in [RFC7162] and [RFC5161] to get an optimized and quick synchronization between the – potentially offline – Message Storage Client and the Message Storage Server.

Extra server state information and persistent storage of mod-sequences may not always be supported, in which case the Message Storage Client receives such indication (as defined in section 3.1.2 and section 3.2.5 in [RFC7162]) from the Message Storage Server.

If supported, the Message Storage Client SHALL follow the general instructions as defined in section 5 in [RFC7162], with the following amendment:

1. Synchronize new messages: Fetch all messages whose UID is greater than the last UID known to the Message Storage Client.
2. If VANISHED UID FETCH Modifier is supported, synchronize purged messages: Use VANISHED UID FETCH Modifier, as specified in section 3.2 of [RFC7162].
3. Synchronize significant flag changes:
    a. Sync Read/Unread flag for all messages:
       In order to optimize the query, the client can search for the value that is expected to produce the shortest list of UIDs. For example, if it is assumed that most messages will have been read/seen, then search for the message without flag value “\Seen”. The search will return UIDs with the “\Seen” flag set and UIDs not in the search result do have the flag set.

b. Optionally, use similar approach to synchronize other significant flags. The rest of the flags will not be synchronized into the local store.

6.8 Notification Operations

The “NOTIFY” IMAP Extension [RFC5465] allows the Message Storage Client to request for solicited notifications about events in specified folders of a Message Storage Server. If the Message Storage Client supports the NOTIFY extension, it SHALL send a NOTIFY command to the Message Storage Server to limit its unsolicited notifications to certain selected folders and certain events such as objects being added to or deleted from those selected folders. Without this Message Storage Client NOTIFY command, an IMAP server will only send information about the changes in the Message Storage Server to the client in the following cases:

1. as the result of a Message Storage Client command such as FETCH responses to a FETCH or STORE command),
2. as unsolicited responses sent just before the end of a command (e.g., EXISTS or EXPUNGE) as the result of changes in other sessions, and
3. during the Message Storage Client’s IDLE command, if used.

Upon registration and receiving of either solicited or unsolicited notifications from the Message Storage Server, the Message Storage Client SHALL update its corresponding information in the client(s)’ locally stored resources.

7. Procedures at Message Storage Server

The Message Storage Server is a functional component of the CPM enabler, which allows authorized and/or authenticated principals (such as Message Storage Clients or CPM Participating Functions) to access a resource in the Message Storage Server.

The Message Storage Server SHALL act as an IMAP4 server as defined in [RFC3501], so the Message Store Server SHALL advertise the IMAP extensions supported to the Message Store Client.

In addition to that, the Message Storage Server SHALL support:

- the “UIDPLUS” IMAP4 extension as defined in [RFC4315],
- the “LIST-STATUS” IMAP4 extension as defined in [RFC5819]

and MAY support:

1) the “ACL” IMAP4 extension as defined in [RFC4314],
2) the “URLAUTH” IMAP4 extension as defined in [RFC4467],
3) the “CONDSTORE” IMAP4 extension as defined in [RFC7162],
4) the “ENABLE” IMAP4 extension as defined in [RFC5161],
5) the “QRESYNC” IMAP4 extension as defined in [RFC7162],
6) the “METADATA” IMAP4 extension as defined in [RFC5464],
7) the “XLIST-CHANGEDSINCE” IMAP4 extension as defined in Appendix J,
8) the “ANNOTATE” IMAP4 extension as defined in [RFC5257], and
9) the “CONVERT” IMAP4 extension as defined in [RFC5259], and
10) the “NOTIFY” IMAP extension as defined in [RFC5465], and
11) the “IDLE” IMAP4 extension as defined in [RFC2177].

A Message Storage Server that does not support extra server state or the persistent storage of mod-sequences for the mailbox SHOULD support the VANISHED UID FETCH Modifier as defined in [RFC7162].

7.1 General Operations

7.1.1 Authenticate Operation

Upon receiving a STARTTLS request, the Message Storage Server SHALL process the request and return a response according to the STARTTLS command as defined in [RFC3501]. The Message Storage Server SHALL at least support TLS and PSK-TLS as defined in [RFC2246] and [OMA-SEC-CF] for the TLS protocol negotiations.

Upon receiving an AUTHENTICATE request, the Message Storage Server SHALL process the request and return a response according to the AUTHENTICATE command as defined in [RFC3501] and then complete the authentication process as defined in [RFC3501].

Upon receiving a LOGIN request, the Message Storage Server SHALL process the request and return a response according to the LOGIN command as defined in [RFC3501].

7.1.2 Set Active Folder Operation

Upon receiving a SELECT request, the Message Storage Server SHALL process the request and return a response according to the SELECT command as defined in [RFC3501].

7.2 Access Control List Operations

7.2.1 Set Access Control List

Upon receiving a SETACL request, the Message Storage Server SHALL process the request and return a response according to the SETACL command as defined in [RFC4314].

7.2.2 Get Access Control List

Upon receiving a GETACL request, the Message Storage Server SHALL process the request and return a response according to the GETACL command as defined in [RFC4314].

7.2.3 Delete Access Control List

Upon receiving a DELETEACL request, the Message Storage Server SHALL process the request and return a response according to the DELETEACL command as defined in [RFC4314].

7.2.4 Access Rights Retrieval Operations

Upon receiving a LISTRIGHTS request the Message Storage Server SHALL process the request and return response according to the descriptions as defined in [RFC4314].

7.3 Objects Operations

7.3.1 Object Store Operation

Upon receiving an APPEND request including the object to store, the Message Storage Server SHALL handle the request according to the APPEND command as defined in [RFC3501] and [RFC4315] and store the specified object to the end of the specified destination folder. Upon successful storing the object into message store the Message Storage Server SHALL return the UID assigned to that object back to the requesting Message Store Client.

7.3.1.1 Handling Deferred CPM Message Objects

Upon receiving a request for storing a Deferred CPM Message object that is associated with an expiry time, the Message Storage Server SHALL:

1. handle the request according to the APPEND command as defined in [RFC3501] and store the object to the end of the specified destination folder designated for temporarily holding deferred messages and
2. notify the Message Storage Client of the arrival of the new message.

The stored message object will remain in the designated folder until either it is moved by the Message Storage Client's corresponding commands or it reaches its deferred expiry time and is deleted from the folder.

7.3.2 Object Fetch Operation

Upon receiving a FETCH request with the UID pointing to a stored object, the Message Storage Server SHALL handle the request according to the FETCH or UID FETCH command as defined in [RFC3501].

7.3.3 Object Preview Fetch Operation

Upon receiving a CONVERT or UID CONVERT request with the UID pointing to a stored object of the Message Storage Server, the Message Storage Server SHALL handle the preview request for the supplied format and dimensions according to the CONVERT or UID CONVERT command as defined in [RFC5259]

7.3.4 Object Copy Operation

Upon receiving a COPY request with the UID pointing to a stored object of the Message Storage Server, the Message Storage Server SHALL copy the specified Message(s) or Message History to the end of the specified destination folder according to the COPY command as defined in [RFC3501]

7.3.5 Object Remove Operation

Upon receiving an IMAP STORE command with a UID pointing to the stored object to update the flag list of the object's data to include the "\Deleted" flag, the Message Storage Server SHALL handle the flagging request by setting the stored object's "\Deleted" flag according to the STORE command as defined in [RFC3501]. A session info object can only be removed via session folder removal operation as described in sect. 6.4.4 "*Folder Remove Operation*".

Upon receiving an EXPUNGE request to remove all stored objects from the CPM User's Message Storage Server that have the "\Deleted" flag set for removal, the Message Storage Server SHALL handle the request to permanently remove these stored objects according to the EXPUNGE command as defined in [RFC3501].

7.4 Metadata Management Operations

The Message Storage Server SHALL support:

- updating the metadata flags defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*,
- fetching the metadata flags defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*,
- updating server metadata annotations defined for IMAP4 mailboxes in section 5.5 *"Metadata Structure"*.
- fetching server metadata annotations defined for IMAP4 mailboxes in section 5.5 *"Metadata Structure"*.

The Message Storage Server MAY support:

- updating message and folder metadata annotations defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*.
- fetching message and folder metadata annotations defined for IMAP4 message objects in section 5.5 *"Metadata Structure"*.

7.4.1 Metadata Update Operation

Upon receiving a STORE request with the UID addressing a stored object of the Message Storage Server, the Message Storage Server SHALL update the metadata flags as indicated in the request according to the STORE command as defined in [RFC3501]. If the STORE request includes 'ANNOTATION' data item and the Message Storage Server supports updating message metadata annotations (i.e. indicates support for the ANNOTATE IMAP4 extension defined in [RFC5257]), the Message Storage Server SHALL handle any metadata annotations specified in the STORE request according to the ANNOTATE IMAP4 extension defined in [RFC5257].

Upon receiving a SETMETADATA request with the name of a folder stored object on the Message Storage Server - or the empty string (i.e. "") in case of server annotations -, the Message Storage Server SHALL apply appropriate metadata updates (i.e. add, remove or modify) according to the SETMETADATA command as defined in [RFC5464].

7.4.2 Metadata Fetch Operation

Upon receiving a FETCH request with the UID addressing one or more stored objects of the Message Storage Server, the Message Storage Server SHALL return the metadata flags according to the FETCH command as defined in [RFC3501]. If the FETCH request includes the 'ANNOTATION' data item name and the Message Storage Server supports fetching message metadata annotations (i.e. indicates support for the ANNOTATE IMAP4 extension defined in [RFC5257]), the Message Storage Server SHALL return any message metadata annotations according to the ANNOTATE IMAP4 extension defined in [RFC5257].

Upon receiving a GETMETADATA request with the name of a folder stored on the Message Storage Server - or the empty string (i.e. "") in case of server annotations -, the Message Storage Server SHALL return the appropriate metadata annotations according to the GETMETADATA command as defined in [RFC5464].

7.5 Folder Operations

7.5.1 Folder Create Operation

Upon receiving a CREATE request, the Message Storage Server SHALL create the folder (e.g., mailbox) with the requested name according to the CREATE command as defined in [RFC3501].

The CPM folders are created to store CPM Conversation Histories, or CPM Group Sessions as described in section 8.5 "*Record CPM Conversation History*" of [OMA-CPM-Conv].

7.5.2 List Folders Operation

Upon receiving a LIST request, the Message Storage Server SHALL determine and return the names of all folders of the CPM User on the Message Storage Server according to the LIST command as defined in [RFC3501] and [RFC5819]. If the XLIST-CHANGEDSINCE capability is supported by the server, as defined in Appendix I and the LIST command includes the CHANGEDSINCE argument, the CPM Message Store Server SHALL return a LIST response containing only the folders which have changed since the client was last synchronized with the CPM Message Store.

7.5.3 Folder Move Operation

NOTE: The folder move operation is also used for renaming folders.

Upon receiving a RENAME request with the request-folder name and its new name, the Message Storage Server SHALL rename the indicated folder according to the RENAME command as defined in [RFC3501].

The RENAME operation SHALL NOT be used on a session history folder or a conversation folder while a CPM Session that is being stored (live recorded) is not yet completed. This avoids error situations where during a CPM Session the respective folder does not exist anymore while on-going CPM Chat Messages need to be stored in their respective folder.

7.5.4 Folder Remove Operation

Upon receiving a DELETE request with the request-folder name, the Message Storage Server SHALL remove the indicated folder according to the DELETE command as defined in [RFC3501].

7.5.5 Folder Search Operation

Upon receiving a SEARCH request, with one or more search key data, the Message Storage Server SHALL handle the request according to the IMAP SEARCH command as defined in [RFC3501].

7.6 Reference Operations

7.6.1 Generate Reference Operation

Upon receiving a GENURLAUTH request, the Message Storage Server SHALL process the request and return a response according to the GENURLAUTH command as defined in [RFC4467].

7.6.2 Fetch by Reference Operation

Upon receiving a URLFETCH request, the Message Storage Server SHALL process the request and return a response according to the URLFETCH command as defined in [RFC4467].

7.7 Message and History Synchronization Operations

[OMA-CPM-SD] gives a description of the synchronization process between the Message Storage Client and the Message Storage Server. This process only consists of operations described above and therefore needs no further explanation in this section.

In addition to that, the Message Storage Server SHALL support the IMAP4 extensions described in [RFC5819] and [RFC4315], and MAY use the IMAP4 extensions described in [RFC5161], [RFC7162] and Appendix J to allow the Message Storage Client to have an optimized and quick synchronization process.

A Message Storage Server that does not support extra server state or the persistent storage of mod-sequences for the mailbox SHALL indicate that to the client as defined in section 3.2.5.1 and section 3.2.5 in [RFC7162]. In addition such Message Storage Server SHOULD support the VANISHED UID FETCH Modifier as defined in sections 3.2.6 and 5.1 in [RFC7162].

7.8 Notifications Operations

Upon receiving the Message Storage Client's NOTIFY command and/or occurrence of changes in the Message Storage Server's resources, the Message Storage Server SHALL reflect the presence of changes via sending Notification messages.

The changes in the Message Storage Server's resources MAY include one or more of the following specific operations per section 5 "Event Types" of [RFC5465].

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 2.2 History

Document Identifier	Date	Sections	Description
Draft Version OMA-TS-CPM_Message_Store-V2_2	09 May 2017	all	Taking the working draft using v2.1-C with all enhancements to become the v2.2 candidate version
Candidate Version OMA-TS-CPM_Message_Store-V2_2	26 Sep 2017	n/a	Status changed to Candidate by TP TP Ref # OMA-TP-2017-0040-INP_CPM-V2_2_ERP_for_1st_Candidate_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [OMA-SCRRULES].

B.1 SCR for Message Storage Client

Item	Function	Reference	Requirement
CPM-TS-MS-C-001_M	Authentication – setting up TLS connection	6.1.1	
CPM-TS-MS-C-002_M	Authentication – SASL method	6.1.1	
CPM-TS-MS-C-003_M	Authentication – username/password method	6.1.1	
CPM-TS-MS-C-004_M	Requesting to set an active folder	6.1.2	
CPM-TS-MS-C-005_M	Setting an ACL	6.2.1	
CPM-TS-MS-C-006_M	Getting an ACL	6.2.2	
CPM-TS-MS-C-007-M	Deleting an ACL	6.2.3	
CPM-TS-MS-C-008_M	Retrieving access rights	6.2.4	
CPM-TS-MS-C-009_M	Requesting to store an object(i.e. Message and Message histories)	6.3.1	
CPM-TS-MS-C-010_M	Requesting to fetch message(s)	6.3.2	
CPM-TS-MS-C-011_O	Requesting to fetch a preview object	6.3.3	
CPM-TS-MS-C-012_M	Requesting to copy objects to another folder	6.3.4	
CPM-TS-MS-C-013_M	Requesting to remove objects	6.3.5	
CPM-TS-MS-C-014_M	Requesting to create a folder	6.4.1	
CPM-TS-MS-C-015_M	Requesting to get a list of folders	6.4.2	
CPM-TS-MS-C-016_M	Requesting to rename the folder	6.4.3	
CPM-TS-MS-C-017_M	Requesting to remove the folder	6.4.4	
CPM-TS-MS-C-018_M	Requesting to search contents by a few key words	6.4.5	
CPM-TS-MS-C-019_M	Updating metadata flags of a stored object	6.6.1	
CPM-TS-MS-C-019a_M	Fetching metadata flags of a stored object	6.6.2	
CPM-TS-MS-C-020_O	Updating metadata annotations of a stored object (ANNOTATE extension)	6.6.1	

Item	Function	Reference	Requirement
CPM-TS-MS-C-020a_O	Fetching metadata annotations of a stored object (ANNOTATE extension)	6.6.2	
CPM-TS-MS-C-021_M	Updating metadata annotations of a folder (METADATA extension)	6.6.1	
CPM-TS-MS-C-021a_O	Fetching metadata annotations of a folder (METADATA extension)	6.6	
CPM-TS-MS-C-021b_M	Support CPM METADATA annotations	0	
CPM-TS-MS-C-021c_M	Support CPM METADATA server annotations	6.6	
CPM-TS-MS-C-022_M	Synchronization	6.6	
CPM-TS-MS-C-023_O	Notifications	6.7	

B.2 SCR for Message Storage Server

Item	Function	Reference	Requirement
CPM-TS-MS-S-001_M	Authentication – setting up TLS connection	7.1.1	
CPM-TS-MS-S-002_M	Authentication – SASL method	7.1.1	
CPM-TS-MS-S-003_M	Authentication – username/password method	7.1.1	
CPM-TS-MS-S-004_M	Requesting to set an active folder	7.1.2	
CPM-TS-MS-S-005_M	Setting an ACL	7.2.1	
CPM-TS-MS-S-006_M	Getting an ACL	7.2.2	
CPM-TS-MS-S-007-M	Deleting an ACL	7.2.3	
CPM-TS-MS-S-008_M	Retrieving access rights	7.2.4	
CPM-TS-MS-S-009_M	Storing objects to a folder	7.3.1	
CPM-TS-MS-S-010_M	Sending objects	7.3.2	
CPM-TS-MS-S-011_O	Converting the object as requested	7.3.3	
CPM-TS-MS-S-012_M	Copying the objects to the another folder	7.3.4	
CPM-TS-MS-S-013_M	Removing the object	7.3.5	
CPM-TS-MS-S-014_M	Creating a folder with a requested folder name	7.5.1	
CPM-TS-MS-S-015_M	Sending the name of all folders	7.5.2	
CPM-TS-MS-S-016_M	Renaming the folder	7.5.3	
CPM-TS-MS-S-017_M	Deleting the folder	7.5.4	

Item	Function	Reference	Requirement
CPM-TS-MS-S-018_M	Searching objects by a few keywords	7.5.5	
CPM-TS-MS-S-019_M	Updating metadata flags for a stored object	7.4.1	
CPM-TS-MS-S-019a_M	Fetching metadata flags for a stored object	7.4.2	
CPM-TS-MS-S-020_O	Updating metadata annotations of a stored object (ANNOTATE extension)	7.4.1	
CPM-TS-MS-S-020a_O	Fetching metadata annotations of a stored object (ANNOTATE extension)	7.4.2	
CPM-TS-MS-S-021_M	Updating metadata annotations of a folder (METADATA extension)	7.4.1	
CPM-TS-MS-S-021a_O	Fetching metadata annotations of a folder (METADATA extension)	7.4	
CPM-TS-MS-S-021b_M	Support CPM METADATA annotations	0	
CPM-TS-MS-S-021c_M	Support CPM METADATA server annotations	7.4	
CPM-TS-MS-S-022_M	Processing the synchronization	7.7	
CPM-TS-MS-S-023_O	Notifications	7.8	

Appendix C. CPM-defined MIME headers for IMAP objects

The following MIME header fields are defined in this specification as extensions to the [RFC2822] field definitions:

- Conversation-ID,
- Contribution-ID,
- InReplyTo-Contribution-ID,
- IMDN-Message-ID,
- P-Asserted-Service, and
- Message-Correlator.

C.1 Conversation-ID MIME header field

The limits for the occurrence of the field are defined in the following table:

Field	Min Number	Max Number
conversation-id	0	1

Table 7: Conversation-ID header field

The field itself is defined in ABNF as follows:

```
Conversation-ID = "Conversation-ID:" Token
```

C.2 Contribution-ID MIME header field

The limits for the occurrence of the field are defined in the following table:

Field	Min Number	Max Number
contribution-id	0	1

Table 8: Contribution-ID header field

The field itself is defined in ABNF as follows:

```
Contribution-ID = "Contribution-ID:" Token
```

C.3 InReplyTo-Contribution-ID MIME header field

The limits for the occurrence of the field are defined in the following table:

Field	Min Number	Max Number
InReplyTo-Contribution-id	0	1

Table 9: InReplyTo-Contribution-ID header field

The field itself is defined in ABNF as follows:

```
InReplyTo-Contribution-ID = "InReplyTo-Contribution-ID:" Token
```

C.4 IMDN-Message-ID MIME header field

The limits for the occurrence of the field are defined in the following table:

Field	Min Number	Max Number
IMDN-Message-ID	0	1

Table 10: IMDN-Message-ID header field

The field itself is defined in ABNF as follows:

```
IMDN-Message-ID = "IMDN-Message-ID:" Token
```

C.5 P-Asserted-Service MIME header field

The limits for the occurrence of the field are defined in the following table:

Field	Min Number	Max Number
P-Asserted-Service	0	1

Table 11: P-Asserted-Service header field

The field itself is defined in ABNF as follows:

```
P-Asserted-Service = "P-Asserted-Service:" Token
```

C.6 Message-Correlator MIME header field

The limits for the occurrence of the field are defined in the following table:

Field	Min Number	Max Number
Message-Correlator	0	1

Table 12: Message-Correlator header field

The field itself is defined in ABNF as follows:

```
Message-Correlator = "Message-Correlator:" message-correlator-value
CRLF
message-correlator-value= ascii-value / non-ascii-value
ascii-value                    = * (%x20-7E)
non-ascii-value    = "=?" charset "?" encoding "?" encoded-text "?="
                ; encoding as defined in [RFC2047] for
                ; encoded-word
charset                        = "utf-8"
encoding                       = "b"
```

C.7 Accept-Contact MIME header

The occurrence of the MIME header is defined in the following table:

Field	Min Number	Max Number
Accept-Contact	1	*

Table 13: Accept-Contact header

The ABNF definition for this header is as defined in [RFC3841].

Appendix D. Example of Session History Folder

Note: all message bodies are encoded base 64, not shown in the Appendix examples for demonstration purposes. The base 64 encoded parts are marked in gray background for representation purposes.

Below is an example of a 1-1 session history folder with the following content:

- One session info object.
- One CPM Chat Message with text content in message object 1.

Another example is provided in Appendix F.

The session info object

```
From: John Doe <jdoe@machine.example>
To: Alice <sip:alice@example.com>
Date: Fri, 29 Apr 2014 09:46:50 -0800
Subject: the weather will be fine today
Conversation-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
Contribution-ID: abcdef-1234-5678-90ab-cdef01234567
InReplyTo-Contribution-ID: 01234567-89ab-cdef-0123-456789abcdef
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.oma.cpm.session
IMDN-Message-ID: 654131a654131a654131a654131a8994656
Content-Type: Application/X-CPM-Session

<?xml version="1.0" encoding="UTF-8"?>
<session>
<session-type>1-1</session-type>
</session>
```

The message object 1 carrying a CPM Chat Message

```
From: John Doe <jdoe@machine.example>
To: Alice <sip:alice@example.com>
DateTime: 2014-04-29T09:47:00-08:00
Subject: Hello World
Conversation-Id: a2133b-654131a-f564321-c5d654
Contribution-Id: 683135-a654bb2-35c641d-84679e
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.oma.cpm.session
IMDN-Message-ID: 654131a654131a654131a654131a8994656
Content-Type: Message/CPIM

From: <sip:anonymous@anonymous.invalid>
To: <sip:anonymous@anonymous.invalid>
imdn.Message-ID: 654131a654131a654131a654131a8994656
DateTime: 2014-04-29T09:48:23.24-08:00
imdn.Disposition-Notification: positive-delivery, display
Content-type: text/plain; charset=utf-8

Here is the text of my message.
```

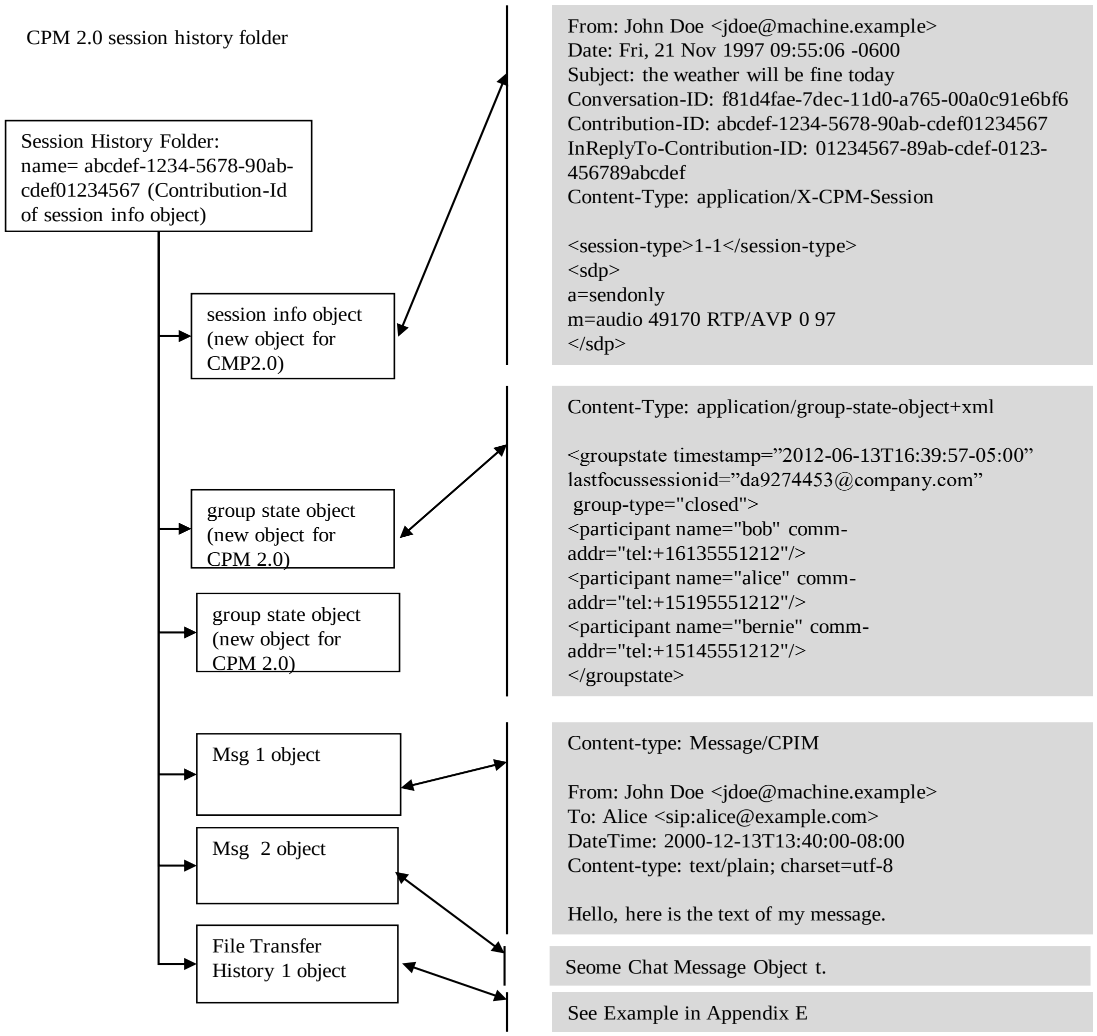
Appendix E. Example of File Transfer History Object

Below is an example of a 1-1 File Transfer History Object in which one file was received.

```
From: John Doe <jdoe@machine.example.com>
To: Alice <sip:alice@example.com>
Date: Fri, 21 Nov 1997 09:55:06 -0600
Conversation-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
Contribution-ID: abcdef-1234-5678-90ab-cdef01234567
InReplyTo-Contribution-ID: 01234567-89ab-cdef-0123-456789abcdef
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.oma.cpm.filetransfer
IMDN-Message-ID: 654131a654131a131bfrufh37846r44tcbrfb94656

Content-type: multipart/related;boundary=cpm; type=”Application/X-CPM-File-Transfer”

--cpm
Content-Type: Application/X-CPM-File-Transfer

<?xml version="1.0" encoding="UTF-8"?>
<file-transfer>
<file-transfer-type>1-1</file-transfer-type>
<imdn></imdn>
<sdp>
        i=This is my latest picture
        a=sendonly
        a=file-selector:name:"My picture.jpg" type:image/jpeg size:4092
        a=file-disposition:render
        a=file-date:creation:"Mon, 15 May 2006 15:01:31 +0300"
        a=file-icon:cid:mythumbnail@example.com
</sdp>
<file-object>
      <cid>cid:1234@example.com</cid>
</file-object>
</file-transfer>

--cpm
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <mythumbnail@example.com>

   ... mythumbnail.jpg...

--cpm
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <1234@example.com>

   ... My picture.jpg...
--cpm--
```

Appendix F. Storage of CPM Session

The illustration below includes the main header fields in the SIP, MSRP and CPIM messages. While other header fields may exist, they are not shown below for simplification. Note that any standalone Media Object exchanged during the CPM Session is stored in the conversation folder and not in the session history folder.


CPM 2.0 session history folder
Session History Folder: name= abcdef-1234-5678-90ab-cdef01234567 (Contribution-Id of session info object)
session info object (new object for CMP2.0)
group state object (new object for CPM 2.0)
group state object (new object for CPM 2.0)
Msg 1 object
Msg 2 object
File Transfer History 1 object

```
From: John Doe <jdoe@machine.example>
Date: Fri, 21 Nov 1997 09:55:06 -0600
Subject: the weather will be fine today
Conversation-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
Contribution-ID: abcdef-1234-5678-90ab-cdef01234567
InReplyTo-Contribution-ID: 01234567-89ab-cdef-0123-
456789abcdef
Content-Type: application/X-CPM-Session

<session-type>1-1</session-type>
<sdp>
a=sendonly
m=audio 49170 RTP/AVP 0 97
</sdp>
```

```
Content-Type: application/group-state-object+xml

<groupstate timestamp=”2012-06-13T16:39:57-05:00”
lastfocussessionid=”da9274453@company.com”
 group-type="closed">
<participant name="bob" comm-
addr="tel:+16135551212"/>
<participant name="alice" comm-
addr="tel:+15195551212"/>
<participant name="bernie" comm-
addr="tel:+15145551212"/>
</groupstate>
```

```
Content-type: Message/CPIM

From: John Doe <jdoe@machine.example>
To: Alice <sip:alice@example.com>
DateTime: 2000-12-13T13:40:00-08:00
Content-type: text/plain; charset=utf-8

Hello, here is the text of my message.
```

Seome Chat Message Object t.

See Example in Appendix E

Figure 1 Illustration of content of the CPM Session History folder

Appendix G. Group State Object example

An example of the Group State Object body is in table below:

```
<?xml version="1.0" encoding="UTF-8"?>
<groupstate
            timestamp="2015-12-24T14:39:57-05:00"
            lastfocussessionid="sip:da9274453@company.com"
            iw-number="tel:+15145550001"
            group-type="Open"

      <max-user-count>25</max-user-count>
      <policy>rem-admin</policy>

      <status>
        <removed>
          <participant name="" comm-addr="tel:+15145550110"/>
        </removed>
      </status>

      <participant name="bob" comm-addr="tel:+16135551212" yourown="true"/>
      <participant name="alice" comm-addr="tel:+15195551212"/>
      <participant name="" comm-addr="tel:+15145550110" role="Administrator"/>

      <subject>
        <text>Happy x-mas!</text>
        <participant name="alice" comm-addr="tel:+15195551212"/>
        <timestamp>2015-12-24T09:22:12-05:00</timestamp>
      </subject>

      <icon>
        <source>
          <icon-uri>https://content.company.com/christmas-tree.jpg</icon-uri>
        </source>
        <participant name="alice" comm-addr="tel:+15195551212"/>
        <timestamp>2015-12-24T09:22:28-05:00</timestamp>
      </icon>



</groupstate>
```

Table 14: Group State Object – XML content Example

Appendix H. CPM METADATA annotations (Normative)

This section contains the metadata entries defined by the OMA CPM Enabler.

Each CPM metadata entry is listed in Table 15 along with the following information:

- the valid namespace(s),
- whether the metadata entry can be used as a server and/or a mailbox annotation,
- its purpose, and;
- a reference to the metadata entry format description.

Entry name	Valid namespace(s)	Server annotation	Mailbox annotation	Purpose	Format
DefaultFolderLocation	/shared/OMNA/OMACPM20	Y	N	Indicates the location of the “default” system folder; see section “5.5.1.1 Folders” in [OMA-CPM-SD].	H.1.1

Table 15: CPM METADATA annotation reference sheet

H.1 Metadata entry formats

While all metadata entries contain strings in general, it is important to define the format of contents of these strings. This section contains a sub-section describing the format of each metadata entry defined in Table 15.

H.1.1 DefaultFolderLocation

The metadata entry MUST contain a fully qualified mailbox name suffixed with the server's hierarchy separator character (i.e. a parameter for the SELECT command).

For example:

```
/INBOX/CPM2/-=default=-
```

Appendix I. Representation of CPM Conversations in the CPM Message Store (Informative)

This section contains examples of conversations that may take place between CPM Users, focusing on illustrating how the communication representation corresponds to the storage representation. The examples illustrate basic scenarios where a conversation takes place using the same channel, more advanced scenarios where a conversation takes place using different channels but only one channel at a time. The last example included is the most advanced scenario where a single conversation takes place using various channel in parallel.

The basis for these examples is provided by the original high-level diagrams in section 4 of the CPM Requirements document [OMA-CPM-RD].

Please note that these examples do not indicate directions. It does not matter if the message or file was sent or received – it is reflected in the storage just the same.

I.1 Standalone exchanges

The following figure illustrates a new conversation, taking place solely using standalone exchanges such as standalone messages (pager mode, large message mode) and file transfers. Each CPM Standalone Message corresponds to a single message object and each CPM File Transfer corresponds to a single file transfer history object.

A watchful observer will notice that the InReplyTo-Contribution-ID is not always included. However, when it is included, it establishes a link between two distinct contributions (e.g. a standalone message and a file transfer in the example below).

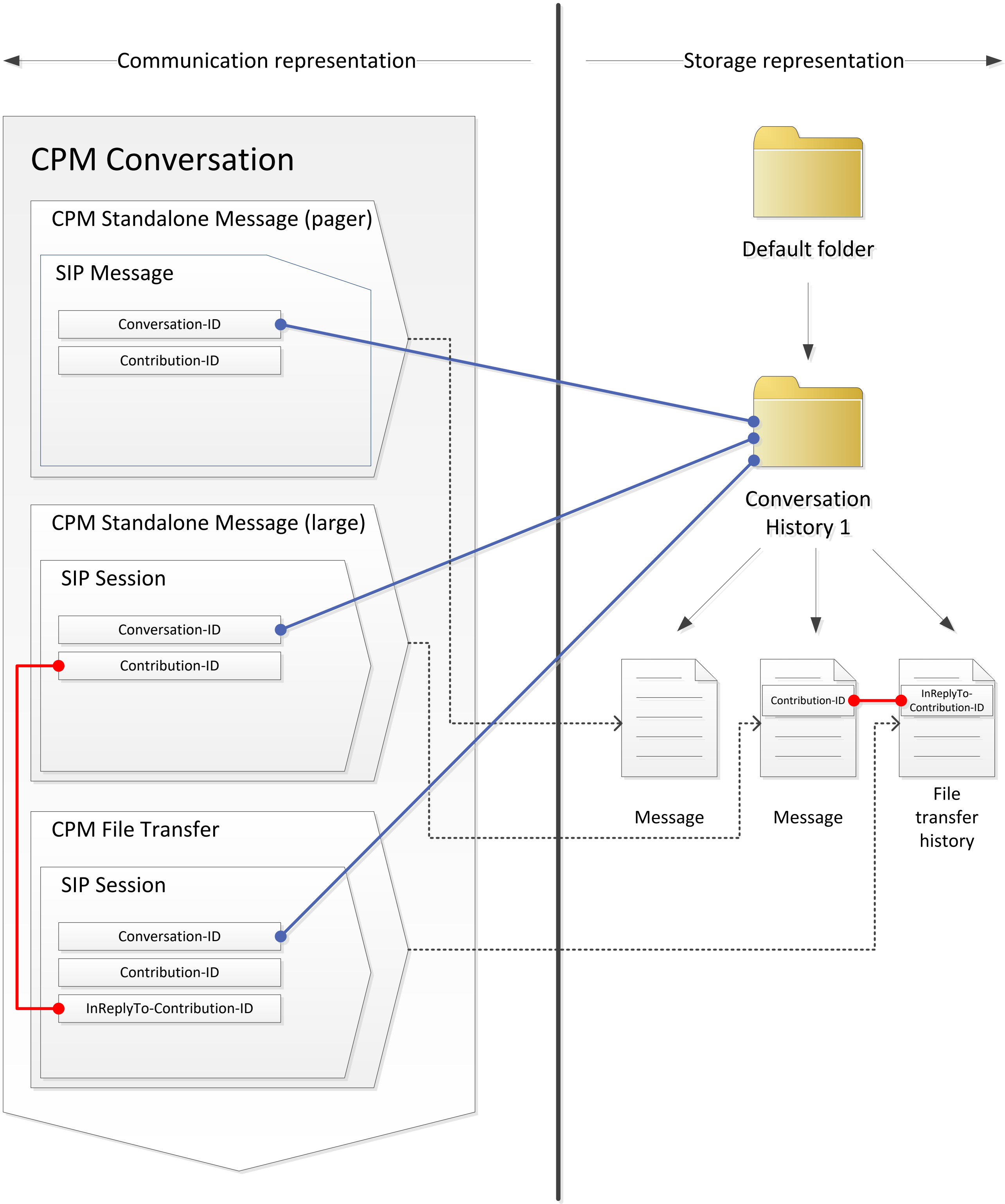


Figure 2: Example of storage representation; standalone exchanges

I.2 Chat exchanges example

The following figure illustrates a new conversation, taking place solely using group chat exchanges (applies to group chat) such as chat message and file transfers. At the beginning of the group chat session, the session info object is stored into the session history folder. In this example all chat messages and file transfers are stored at the end of the chat session. Additionally, one or more Group State Objects are stored (this example shows one such object). Each chat message corresponds to a single message object and each CPM File Transfer corresponds to a single CPM File Transfer History Object.

A watchful observer will notice that the Contribution-ID links a group chat session and the group file transfers in the same CPM Group Session together.

Note: while this example is valid for long-lived chat sessions as well, the point of the long-lived chat sessions is better illustrated on Figure 4.

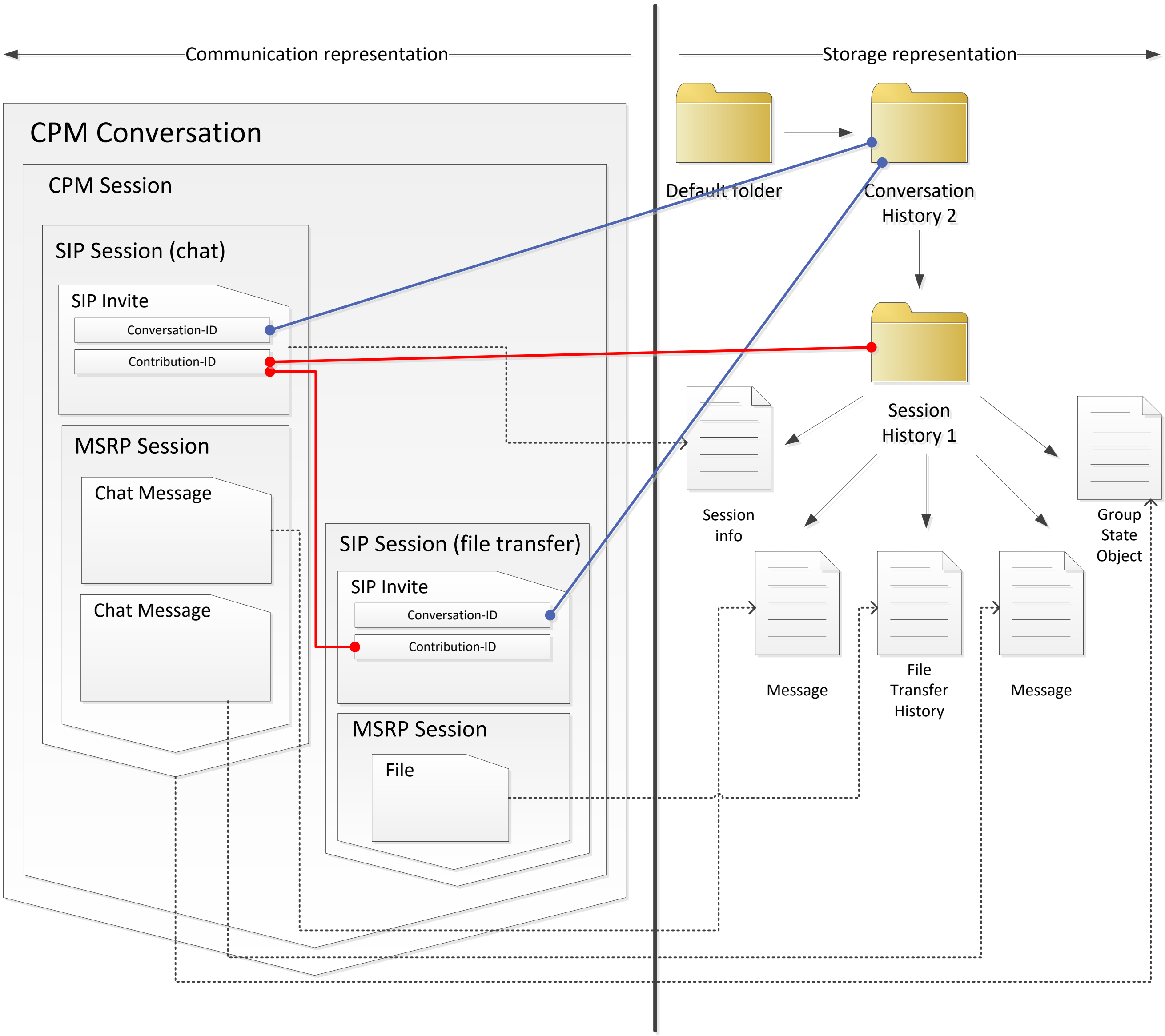


Figure 3: Example of storage representation; chat exchanges

I.3 Long-lived chat exchanges

The following figure illustrates a new long-lived group chat conversation, taking place solely using long-lived group chat exchanges and file transfers. At the beginning of the long-lived group chat session, the session info object is stored into the session history folder. In this example, all chat messages and file transfers are stored at the end of each SIP session. Additionally, one or more Group State Objects are stored and correspond to the CPM Session. Each chat message corresponds to a single message object and each CPM File Transfer corresponds to a single CPM File Transfer History Object.

After some time, the long-lived group chat session is restarted, using the Conversation-ID and Contribution-ID to establish the link. The session info object is already stored for this CPM Session; hence a new session info object is not stored. In this example, all chat messages and file transfers are stored at the end of each SIP session. Additionally, additional Group State Objects are stored and correspond to this CPM Session. Each chat message corresponds to a single message object and each CPM File Transfer corresponds to a single CPM File Transfer History Object.

Editor’s note: update figure 3 to demonstrate two GSOs corresponding to single CPM Session

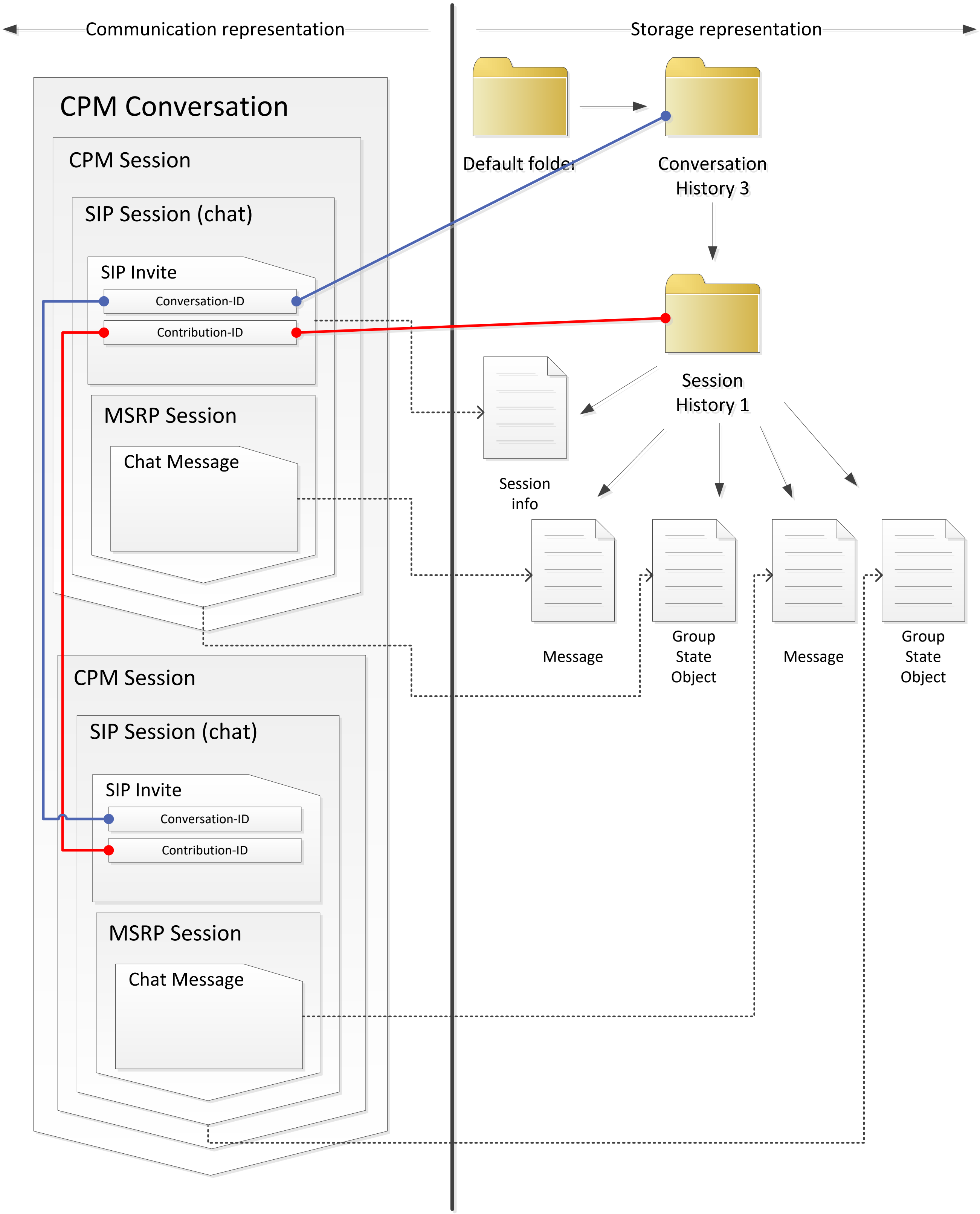


Figure 4: Example of storage representation; long lived chat exchanges

I.4 Standalone to 1-1 chat

The following figure illustrates a new conversation, that takes place initially using standalone exchanges and later on it takes place using a 1-1 chat, which is an example of a conversation that begun on a mobile device and continued over a desktop computer.

The 1-1 chat invitation could contain an InReplyTo-Contribution-ID referencing any of the Standalone Messages in the conversation, but it would not make any difference in practice as the link is established solely using the Conversation-ID, which is the same in all Standalone Messages and the 1-1 chat. If a different Conversation-ID was used in the 1-1 chat, a new conversation would be started instead, resulting in a new Conversation History, as shown in Figure 3.

There is nothing preventing the users from having the same conversation using both standalone exchanges and 1-1 chat in parallel: the messages would be reflected into storage exactly the same way. However, the timestamp of the stored messages would reveal that these two did, in fact, happen in parallel.

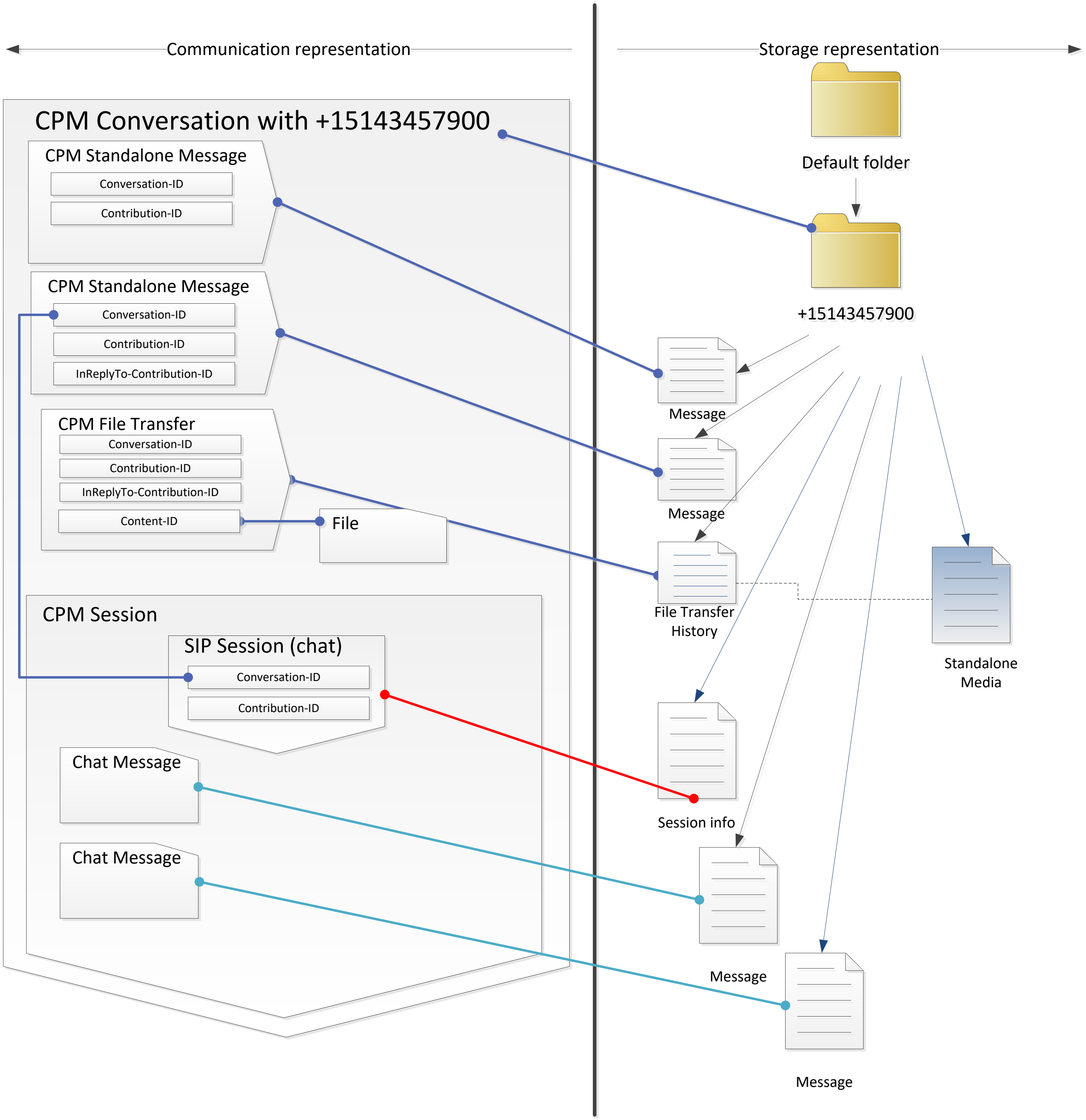


Figure 5: Example of storage representation; standalone to 1-1 chat

I.5 Extending 1-1 chat to group chat

The following figure illustrates a new conversation, that takes place initially using 1-1 chat and later on it is extended to a group chat by replacing the 1-1 chat with a group chat using the "*Extending a CPM 1-1 Session to a CPM Group Session*" procedure.

The link between the 1-1 chat and the group chat is established solely using the Conversation-ID and the Session-Replaces header field. If the Session-Replaces header field were missing, it would mean that a new, parallel CPM Group Chat Session is starting (i.e. the 1-1 chat session is retained) in the same conversation – and consequently, in the same Conversation History, as shown in Figure 7.

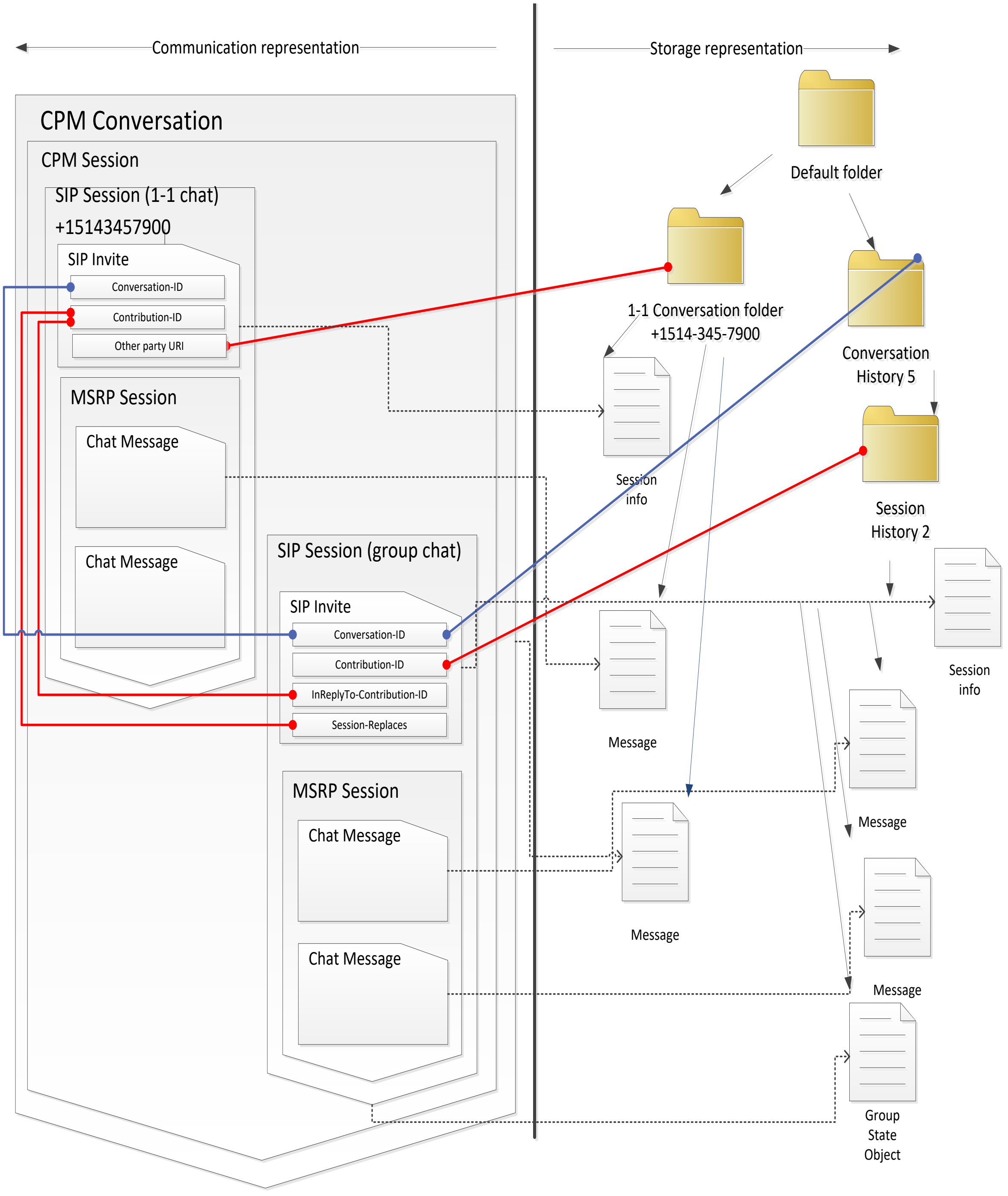


Figure 6: Example of storage representation; extending 1-1 chat to group chat

I.6 Conversation via parallel channels

The following figure illustrates a new conversation that takes place using standalone group exchanges, a long-lived group chat in parallel.

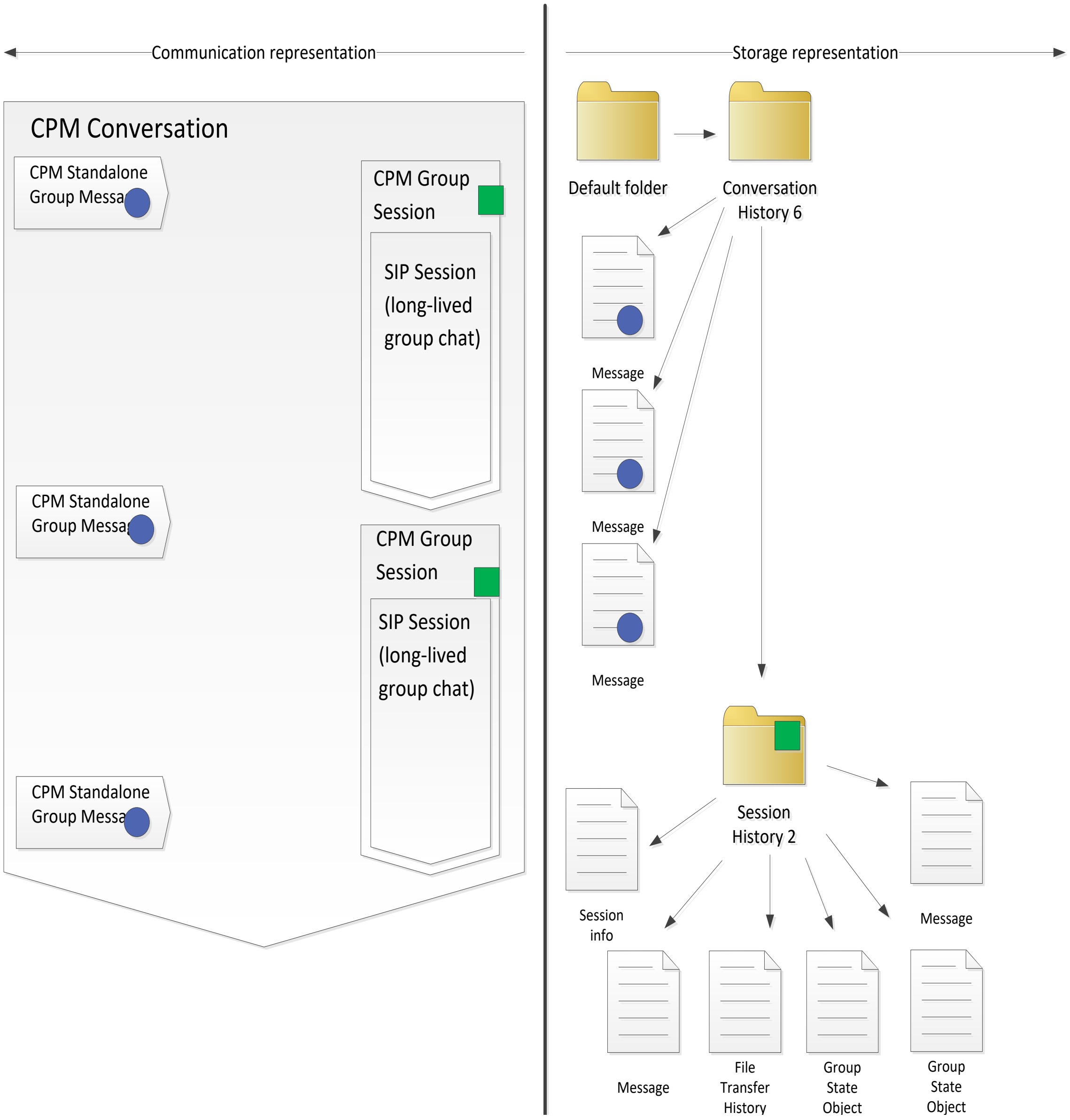


Figure 7: Example of storage representation; conversation via parallel channels

Appendix J. List Folders Operations (Normative)

For a more efficient identification of which folders/mailboxes need synchronization, the CPM Message Store Client and CPM Message Store Server use the following IMAP commands for the folder list operations, as described in section 6.4.2, and section 7.5.2:

1. LIST-STATUS extension defined in [RFC5819], providing a more efficient identification of which mailboxes may need synchronization;
2. CONDSTORE/QRESYNC extension defined in [RFC7162], enabling a more efficient synchronization of individual mailboxes;
3. XLIST-CHANGEDSINCE extension defined in this Appendix, combining the benefits of both of the above-mentioned extensions, enabling a CPM Client to request only the LIST of the folders that have changed since last synchronization done by the CPM Client. This allows the CPM Client to subsequently issue a SELECT command only on the folders requiring synchronization.

J.1 XLIST-CHANGEDSINCE Overview

The XLIST-CHANGEDSINCE extension puts together the functionality provided by the LIST-STATUS command as defined in [RFC5819], with the benefits of CONDSTORE/QRESYNC defined in [RFC7162], which results in the ability to only return the folders which have larger MODSEQ values than the base MODSEQ value in the LIST command. This allows the CPM Message Store Client to determine which folders have changed since the last MODSEQ value known to the CPM Client.

Currently CONDSTORE MODSEQ values can be accessed in the LIST-STATUS operation, and are displayed as part of the output of the LIST. The LIST-EXTENDED extension defined in [RFC5258] provides a method of constraining the resulting list based on “Selection options”.

The XLIST-CHANGEDSINCE extension adds a new “Selection option” for the LIST command to accept a MODSEQ value. The returned list of mailboxes would then only contain mailboxes modified later than the supplied MODSEQ value. When a CPM Client issues a LIST command with the CHANGEDSINCE argument, it SHOULD provide the highest MODSEQ value that it has retrieved at the last synchronization.

J.2 Advertising Support for XLIST-CHANGEDSINCE

A CPM Message Store Server that supports the CHANGEDSINCE extension SHALL return "XLIST-CHANGEDSINCE" as one of the supported capabilities in the CAPABILITY command response.

A CPM Message Store Server supporting XLIST-CHANGEDSINCE extension must necessarily also support CONDSTORE capabilities as defined in [RFC7162]. The CPM Message Store Server MAY advertise QRESYNC without advertising CONDSTORE capability, though advertising both is recommended by [RFC7162].

Although the CHANGEDSINCE argument makes use of certain aspects of the LIST-EXTENDED capability defined in [RFC5258], full support of the extension LIST-EXTENDED is not required in which case the LIST-EXTENDED capability does not need to be advertised.

J.3 XLIST-CHANGEDSINCE Handling

The LIST command with CHANGEDSINCE argument:

1) if the QRESYNC capability is supported by the CPM Message Store Server:
    a) SHALL return all the changed mailboxes, including those with expunged messages (as expunge causes a change in the MODSEQ value for the mailbox in this case); else
2) else, if the QRESYNC capability is not supported by the CPM Message Store Server:
    a) SHALL return the list of mailboxes with either new messages, or with changed metadata (e.g. flag changes) and MAY include folders with expunged messages.

In order to support LIST-CHANGEDSINCE, the CPM Message Store Server SHALL support MODSEQ values that are unique across all mailboxes accessible within a user session. The [RFC4551] does not require globally unique MODSEQ values. A CPM Message Store Server SHALL NOT advertise the XLIST-CHANGEDSINCE capability if MODSEQ values are unique only per mailbox.

J.4 LIST Command Changes

The IMAP4 List Command Extensions [RFC5258] includes the ability to affect the set of folders returned by the LIST command using “Selection options”, including SUBSCRIBED, REMOTE and RECURSIVEMATCH.

The set of defined Selection options is extended herein to include a new value: CHANGEDSINCE. The CHANGEDSINCE selection option SHALL be provided with a MODSEQ argument, as defined by [RFC7162]. The MODSEQ argument identifies the point in time after which any changes should be identified. The syntax of the command is unchanged from that defined in [RFC5258], other than addition of a new Selection option value.

When provided with a CHANGEDSINCE argument, the LIST command SHALL return only those folders matching the reference folder pattern, which have MODSEQ values higher than the value provided with the CHANGEDSINCE argument.

In the examples below, "C:" and "S:" indicate lines sent by the CPM Message Store Client and CPM Message Store Server respectively.

Example 1: The LIST command is used with CHANGEDSINCE argument:

```
       C: a23 LIST (CHANGEDSINCE (821102884)) “” *
S: * LIST (\Marked \NoInferiors) "/" "Default"
S: * LIST () "/" "tel:15143457900"
S: * LIST () "/" "f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
S: * LIST () "/" "f81d4fae-7dec-11d0-a765-00a0c91e6bf5/dfed4fae-3dec-11d0-
a765-00a0c91a3bf2"
S: a23 OK LIST Completed
```

Example 2:

The LIST command used with CHANGEDSINCE is compatible with the LIST -STATUS capability. A CPM Message Store Server supporting both XLIST-CHANGEDSINCE and LIST-STATUS, and with the CONDSTORE addition of MODSEQ to the STATUS command, allows the MODSEQ values to be displayed in the LIST command:

```
C: a24 LIST (CHANGEDSINCE (821102884)) “” * RETURN (STATUS (UIDNEXT
HIGHESTMODSEQ))
S: * LIST (\Marked \NoInferiors) "/" "Default"
S: * STATUS "Default" (UIDNEXT 1782 HIGHESTMODSEQ 821109221)
S: * LIST () "/" "tel:15143457900"
S: * STATUS "tel:15143457900" (UIDNEXT 17 HIGHESTMODSEQ 821102893)
S: * LIST () "/" "f81d4fae-7dec-11d0-a765-00a0c91e6bf5/dfed4fae-3dec-11d0-
a765-00a0c91a3bf2"
S: * STATUS "f81d4fae-7dec-11d0-a765-00a0c91e6bf5/dfed4fae-3dec-11d0-a765-
00a0c91a3bf2" (UIDNEXT 8 HIGHESTMODSEQ 821106452)
S: * LIST () "/" "a81d4fae-7dec-11d0-a765-00a0c91e6bf1/afed4fae-3dec-11d0-
a765-00a0c91a3bf5"
S: * STATUS "a81d4fae-7dec-11d0-a765-00a0c91e6bf1/afed4fae-3dec-11d0-a765-
00a0c91a3bf5" (UIDNEXT 88 HIGHESTMODSEQ 821108927)
S: a24 OK LIST Completed
```

As shown, by using XLIST-CHANGEDSINCE, the LIST response can still include MODSEQ but would only include the mailboxes which had MODSEQ values higher than that provided with the CHANGEDSINCE argument

J.5 Behaviour for Mailboxes with NOMODSEQ

The [RFC7162] specifies the behaviour for folders that do not support MODSEQ values. Those folders return NOMODSEQ response code when a SELECT/EXAMINE command is issued.

The LIST command with CHANGEDSINCE argument SHALL not return any folders which have NOMODSEQ. Those folders cannot be identified as candidates using this extension mechanism.

J.6 Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in [RFC5234]. Terms not defined here are taken from [RFC3501] and [RFC5258].
The syntax of the LIST command is not significantly changed from that shown in LIST-EXTENDED [RFC5258]. Specifically, the extension of "list-select-independent-opt" in this extension makes use of the "option-extension" field syntax as defined therein.

```
capability =/ “XLIST-CHANGEDSINCE”


list-select-independent-opt =/ “CHANGEDSINCE” SP “(“ mod-sequence-value “)”
                    ; mod-sequence-value is defined in [RFC7162]
```

Appendix K. CPM-defined IMAP Metadata Flag

K.1 Archived

Archived is a CPM-defined IMAP metadata flag associated with a Message Object stored in the Message Storage Server. The Archived flag follows the definition of keyword in [RFC3501].

The definition of the Archived metadata flag is:

```
        Archived
            The message is archived
```

The formal syntax of the Archived status flag is:

```
        flag-keyword = "Archived"
```

Example:

```
        C: A003 STORE 2:4 +FLAGS (Archived)
```

The Message Storage Client that supports the Archived metadata flag SHALL check that flag is listed under PERMANENTFLAGS received from the Message Storage Server the response to SELECT command.

In such case, the Message Storage Client SHALL set the Archived flag when the CPM User has indicated that a particular message object is to be archived in the Message Storage Server. Such an object SHALL NOT be managed under expiration rules defined for other message objects and SHALL remain archived until the CPM User takes action to either un-archive it, or delete it. A Message Storage Client SHOULD replace the Archived flag with the \Deleted flag when a message object is deleted. If both flags are present on a message object, the \Deleted flag SHALL take precedence over the Archived flag, and thus expunge (expiry) SHALL remove Archived messages with the \Deleted flag.