BLUETOOTH DOC	Date / Year-Month-Day 2013-07-16	Approved Adopted	Revision V11	Document No MAP_SPEC
Prepared Car Working Group	E-mail address car-main@bluetooth.org			N.B.

MESSAGE ACCESS PROFILE

Abstract

The Message Access Profile (MAP) specification defines a set of features and procedures to exchange messages between devices. It is especially tailored for the automotive hands-free use case where an onboard terminal device (typically a Car-Kit installed in the car) takes advantage of the messaging capability of a communication device (typically a mobile phone). This profile can however also be used for other use cases that require the exchange of messages between two devices.

Revision History

Revision	Date	Comments
D01	2004-05-20	First preliminary draft
D03	2004-08-02	First release to the CWG
D05r01	2004-09-30	Release 1 of the 0.5 candidate
D05r03	2004-11-08	Format and Editorial Changes. S. Raxter
D05r04	2004-11-09	Correct references to Phone Book
D05r05	2005-08-15	Result of the Singapore F2F, final review and approval
D05r06	2005-08-29	Further edits
D05r07	2005-09-23	Outcome of the Illinois face to face included
D07r01	2006-06-18	Message format description (draft) + SAP use case
D07r02	2006-08-30	Results of the Bellevue Face To Face included
D07r03	2006-12-10	A few minor improvements
D07r04	2006-12-20	Minor comments included
D07r05	2007-01-19	Minor comments included
D07r06	2008-02-18	New structure SIM-Access, Renaming OBEX services, Renaming roles, Delete feature, PAN-usage
D09r01	2008-06-17	Release 1 of the 0.9 candidate, SAP configuration removed, several modifications and comments
D09r02	2008-07-04	Handling of particular message types
D09r03	2008-07-23	Comments BARB and CWG
D09r04	2008-07-29	Further comments BARB, minor modifications
D09r05	2008-07-29	Further comments BARB
D09r06	2008-08-05	Further comments BARB, minor modifications
D09r07	2008-08-26	Clarification folder name convention and secure simple pairing, editorials
D09r08	2008-09-25	Reduced the GAP references to those aspects that differ from the GAP requirements, updates
D10r01	2008-11-04	Modifications after IOP
D10r02	2009-02-16	Editorials and errata
D10r03	2009-05-19	Clarification usage of MAS Instances, BARB approval, OBEX UUIDs
V10r00	2009-06-04	Adopted by the Bluetooth SIG Board of Directors

Revision	Date	Comments
D11r00	2012-01-26	ESR05: Erratum 3385- Section 6.4.3 E3180 – Section 3.1.7 E3546 –Section 3.1.6 E3440 –Section 5.5.4 E3219 –Section 5.3 E3124 –Section 4.5 E3441, Section 3.1.3 E3442, Section 3.1.3 E3443, Section 3.1.8 E3197, Section 3.1.7 E3765, Section 3.1.3 E3603, Section 3.1.3 E3309, Section 2.2 E3530, Section 3.1.7 E3464, Section 6.3.3 E3123, Section 3.1.3
D11r01	2012-03-26	Address reviewers' comments (KK, TB, KP-W)
D11r02	2012-09-16	Reworked the new 'shalls' following the Dublin TCWG F2F. Accepted changes and prepared for BARB voting.
D11r02-dh	2013-01-24	Additional edits. This line may be removed before publication.
D11r03	2013-02-07	ESR06 errata E3445, Section 3.1.3 E3458, Section 3.1.6 E3952, Section 3.1.3 E4103, Section 3.1.3 E4144, Section 3.1.3 E4226, Section 4.1 E4302, Section 5.8.4 E4335, Section 3.1.3 (x4), Section 10 E4461, Section 3.1.4 E4462, Section 3.1.8 E4532, Section 3.1.3 E4578, Section 3.1.3
D11r04	2013-02-11	SIG editor addressed reviewer's comments and added Revision number to previous row (missing infile named D11r03)
D11r05	2013-04-02	Change text per updated CR for E4226
D11r06	2013-04-03	Updated text of CR for E4462
D11r07	2013-04-23	Removed ESR05 and ESR06 errata E3443 Section 3.1.8 E3197 Section 3.1.7 Rejected original text of the following errata and submited new text in a combined E5151 Errata : E3180 Section 3.1.7 E3442 Section 3.1.3 E3765 Section 3.1.3 E4226 Section 4.1 E4462 Section 3.1.8

Revision	Date	Comments
D11r08	2013-05-15	Remove references to errata, change font to be consistent within text boxes.
V11	2013-07-16	Adopted by the Bluetooth SIG Board of Directors

Contributors

Name	Company
Michael BUNTSCHECK	Berner & Mattner
Holger LENZ	Berner & Mattner
Joachim MERTZ (owner)	Berner & Mattner
Rüdiger MOSIG	Berner & Mattner
Rob HULVEY	Broadcom
Burch SEYMOUR	Continental
Thomas CARMODY	CSR
Meshach RAJSINGH	CSR
Stefan HOHL	Daimler
Souichi SAITO	Denso
Don LIECHTY	Extended Systems
Dietmar HEINZELMANN	Harman Becker
Brent KITCHEN	iAnywhere
Chen PENGGANG	IVT
Thomas KARLSSON	Mecel AB
John BARR	Motorola
Michael CARTER	Motorola
Leonard HINDS	Motorola
Tony MANSOUR	Motorola
Lois SHE	Motorola
Andrew TZAKIS	Motorola
Venki VAJJA	Motorola
Stephen RAXTER	National Analysis Center
Stephane BOUET	Nissan
Jamie MCHARDY	Nokia
Jurgen SCHNITZLER	Nokia
Brian TRACY	Nokia
Patrick CLAUBERG	Novero
Jay PERUMAL	Qualcomm
Josselin DE LA BROISE	Parrot
Kyle PENRI-WILLIAMS	Parrot
Terry BOURK	RF Micro Devices

Name	Company
Erwin WEINANS	Sony Ericsson
Tim REILLY	Stonestreet One
Tim HOWES	Symbian
Amir YASSUR	Texas Instruments
Kentaro NAGAHAMA	Toshiba
Robert MALING	Toyota
Akira MIYAJIMA	Toyota
Ryan BRUNER	Visteon

Disclaimer and Copyright Notice

The copyright in this specification is owned by the Promoter Members of Bluetooth® Special Interest Group (SIG), Inc. (“Bluetooth SIG”). Use of these specifications and any related intellectual property (collectively, the “Specification”), is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the “Promoters Agreement”), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members (the “Membership Agreements”) and the Bluetooth Specification Early Adopters Agreements (1.2 Early Adopters Agreements) among Early Adopter members of the unincorporated Bluetooth SIG and the Promoter Members (the “Early Adopters Agreement”). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a “Member”) is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.

THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the *Bluetooth* technology ("*Bluetooth* products") may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of *Bluetooth* products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. **NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.**

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserve the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Copyright © 2013. Bluetooth® SIG, Inc. All copyrights in the Bluetooth Specifications themselves are owned by Ericsson AB, Lenovo (Singapore) Pte. Ltd., Intel Corporation, Microsoft Corporation, Motorola Mobility LLC, Nokia Corporation, and Toshiba Corporation.

***Other third-party brands and names are the property of their respective owners.**

Contents

1 Introduction

1.1 Scope

Mobile Messaging becomes increasingly important. Across the globe, record traffic growth in established messaging markets combined with a number of newer emerging markets caused a volume growth of mobile messaging far beyond anticipated levels. The rapid development of the Smartphone market underlines the increasing importance of mobile messaging and therefore the provision of a profile to address messaging use cases.

The Message Access Profile (MAP) defines the features and procedures that shall be used by devices that exchange message objects. It is based on a Client-Server interaction model where the Client initiates the transactions.

In general MAP will be used to combine the messaging capabilities of a messaging server device and the user interface capabilities of a client device for notifying, browsing, reading, deleting, generation and sending of messages.

For instance the Message Access Profile can be used

- to access in a car to a mobile phone, using the car's display or audio system to avoid cumbersome handling of the mobile phone while driving, providing acoustical announcements of email or SMS reception Text2Speech output of messages etc.
- to provide message access to a mobile messaging device using any available PC or notebook to leverage the superior display and input capabilities of the computer
- to enable messaging to any stationary or mobile devices with IO capability, e.g., TVs, message panels, digital picture frames, eBooks, pagers, portable navigation systems or wearable *Bluetooth* devices.

In all use cases mentioned above the messaging client makes use of the messaging server's capabilities for access to a remote network and its message repository. Thus, the technical requirements for the client can be very restricted.

This MAP version supports the message types SMS,MMS, and email to and from cellular or other networks. Future versions may support additionally other message types, for instance cell-broadcast messages and may add further functionality.

1.2 Profile Dependencies

Interoperability between devices from different manufacturers is provided for a specific service and usage model if the devices conform to a Bluetooth-SIG defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives an unambiguous description of the air interface for specified service(s) and usage model(s).

A profile is dependent upon another profile if it re-uses parts of that profile, by explicitly referencing it. The Bluetooth profile structure and the dependencies of MAP are depicted in Figure 1.1. A profile has dependencies on the adjoined profile(s)..

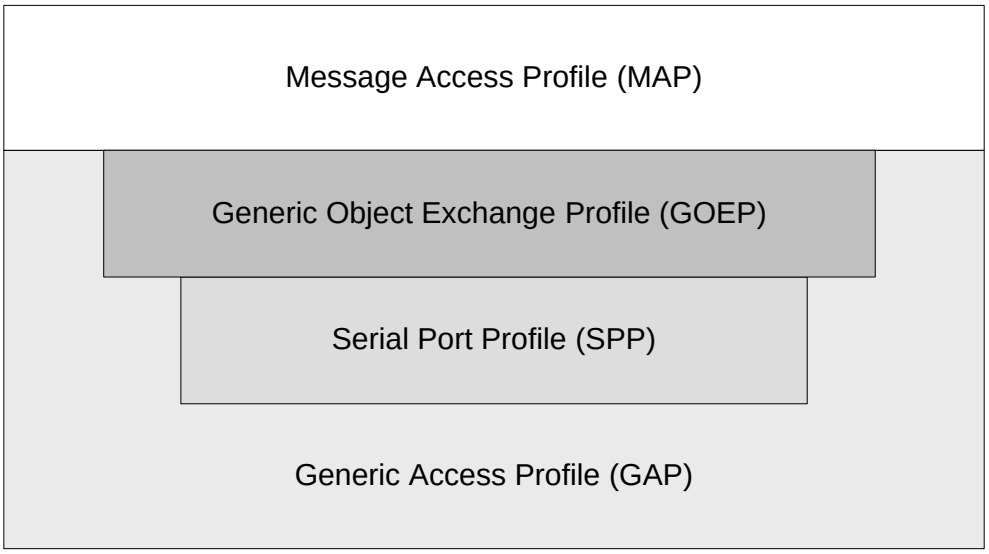

Figure 1.1: Bluetooth Profiles

As indicated in the figure, the Message Access Profile is dependent upon the Generic Object Exchange Profile [3], the Serial Port Profile [16], and the Generic Access Profile [2].

1.3 Symbols and Conventions

1.3.1 Requirement status symbols

This document adopts the following IEEE definitions for describing the criticality of the requirements according to [15]. In particular, the following keywords are used in this specification:

- The word ***shall*** is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals is *required to*).
- The word ***should*** is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals is *recommended that*).
- The word ***may*** is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted but optional*).

Any other descriptions not using the aforementioned terms are informative.

For the feature definition tables in chapter 4 the following symbols are used:

- "M" for mandatory to support
- "O" for optional to support

- "X" for excluded (used for capabilities that may be supported by the unit but shall never be used in this use case)
- "C" for conditional to support
- "N/A" for not applicable (in the given context it is impossible to use this capability)

Some excluded capabilities are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices in this use case. Therefore, these features shall never be activated while a unit is operating as a unit within this use case.

1.3.2 Signaling diagram conventions

The signaling diagrams in this specification are informative only. Within the diagrams, the following conventions are used to describe procedures:

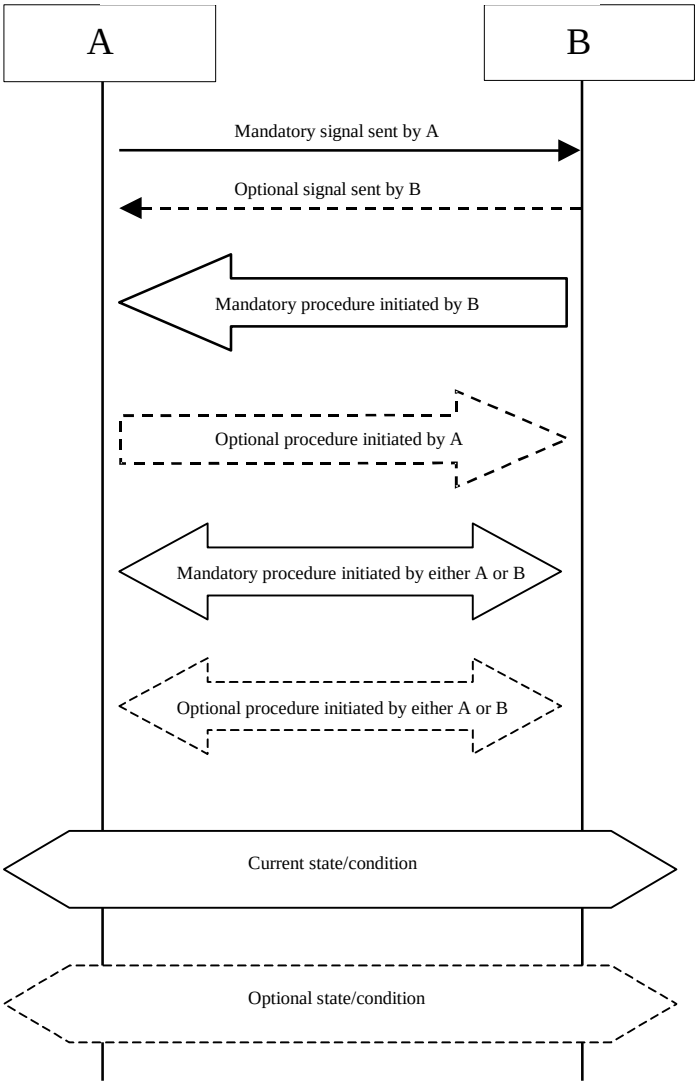

Figure 1.2: Conventions used in signaling diagrams

2 Profile Overview

2.1 Profile Stack

The figure below shows the protocols and entities used in this profile:

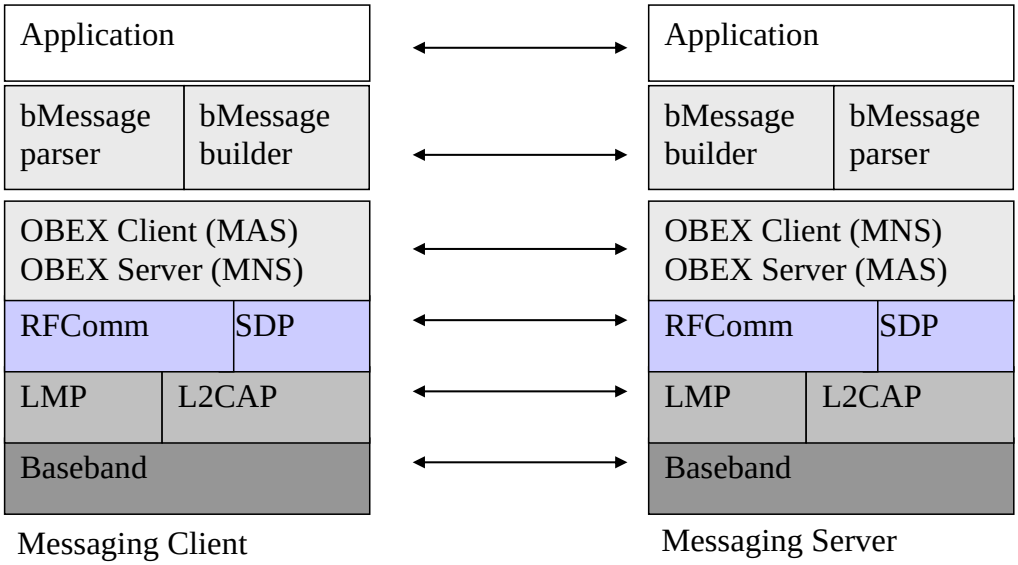

Figure 2.1: Protocol model for transport of bMessages

MAS = Message Access service, MNS = Message Notification service, see chapter 6; bMessages are application objects used by MAP for message transport, see chapter 3.

2.2 Configuration and Roles

The following roles are defined for this profile:

- **Message Server Equipment (MSE)** – is the device that provides the message repository engine i.e. has the ability to provide a client unit with messages that are stored in this device and notifications of changes in its message repository.
- **Message Client Equipment (MCE)** – is the device that uses the message repository engine of the MSE for browsing and displaying existing messages and to upload messages created on the MCE to the MSE.

These terms are used in the rest of this document to designate these roles.

The figures below show typical configurations of devices for which the Message Access Profile is applicable:

- Handsfree Configuration:

  In Figure 2.2 the handsfree unit in the car receives/sends messages from/to a mobile phone which provides capabilities both for network access and message repository (Handsfree Use Case).

- PC Use Case:

  Figure 2.3 shows a configuration with a PC acting as MAP client such that the user is able to use it's PC as IO-device for the messages stored in the cellular phone.

In any case the mobile phone acts as MSE whereas the other devices are having the MCE role.

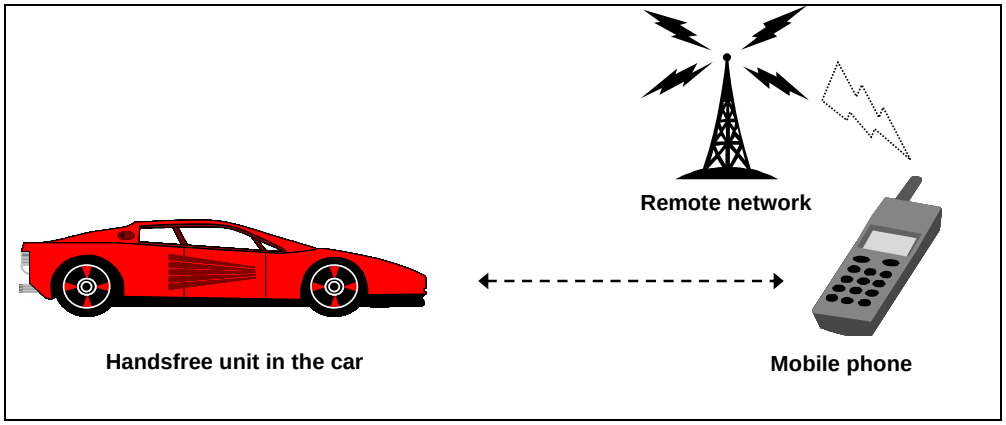

Figure 2.2: Message Access Profile applied to the Handsfree use case

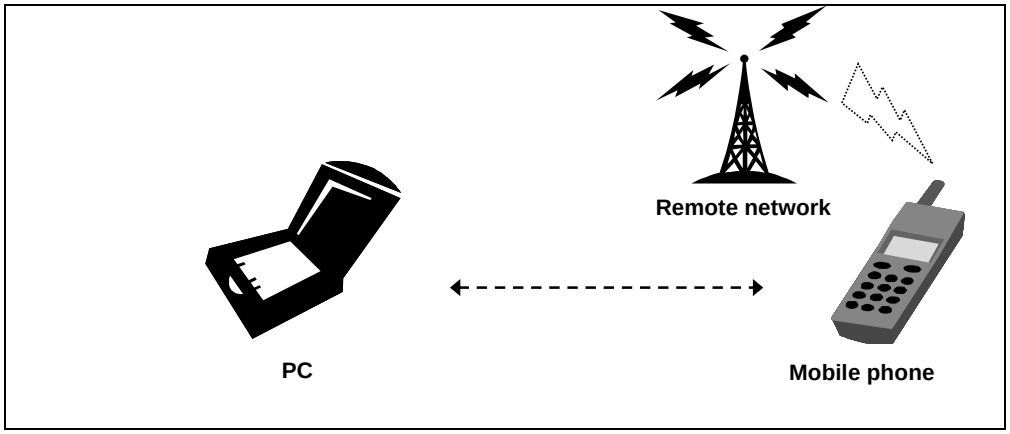

Figure 2.3: Message Access Profile used by a PC to send/receive messages via a mobile phone

This MAP version doesn't consider the configuration where the MSE is in a SIM Access Profile session with the MCE and the network access functionality is on the MCE side. This SAP use case will be covered by a dedicated whitepaper.

2.3 User Requirements and Scenarios

The following are the main scenarios that are covered by this profile:

1. Notifying a remote device of the arrival of new messages on a messaging device:

   In this scenario, the Message Access Profile is used to transfer message arrival notifications, typically from a mobile phone (MSE) to an on-board unit (MCE).

2. Browsing messages in a messaging device:

   In this scenario, the messages that are locally stored in an MSE device can be browsed and, if needed, retrieved individually by a MCE. A typical configuration

would be that of a Bluetooth car-kit or a PC browsing the content of a mobile phone's message repository.

3. Uploading messages onto a messaging device:

   In this scenario, messages are created on the MCE device and uploaded to the MSE device for storage.

4. Deleting messages onto a messaging device:

   In this scenario, the MCE device deletes selected messages on the MSE; e.g. received Spam mails.

5. Sending messages through a remote device:

   In this scenario, the MCE uses the messaging capabilities of the MSE to send messages to a network.

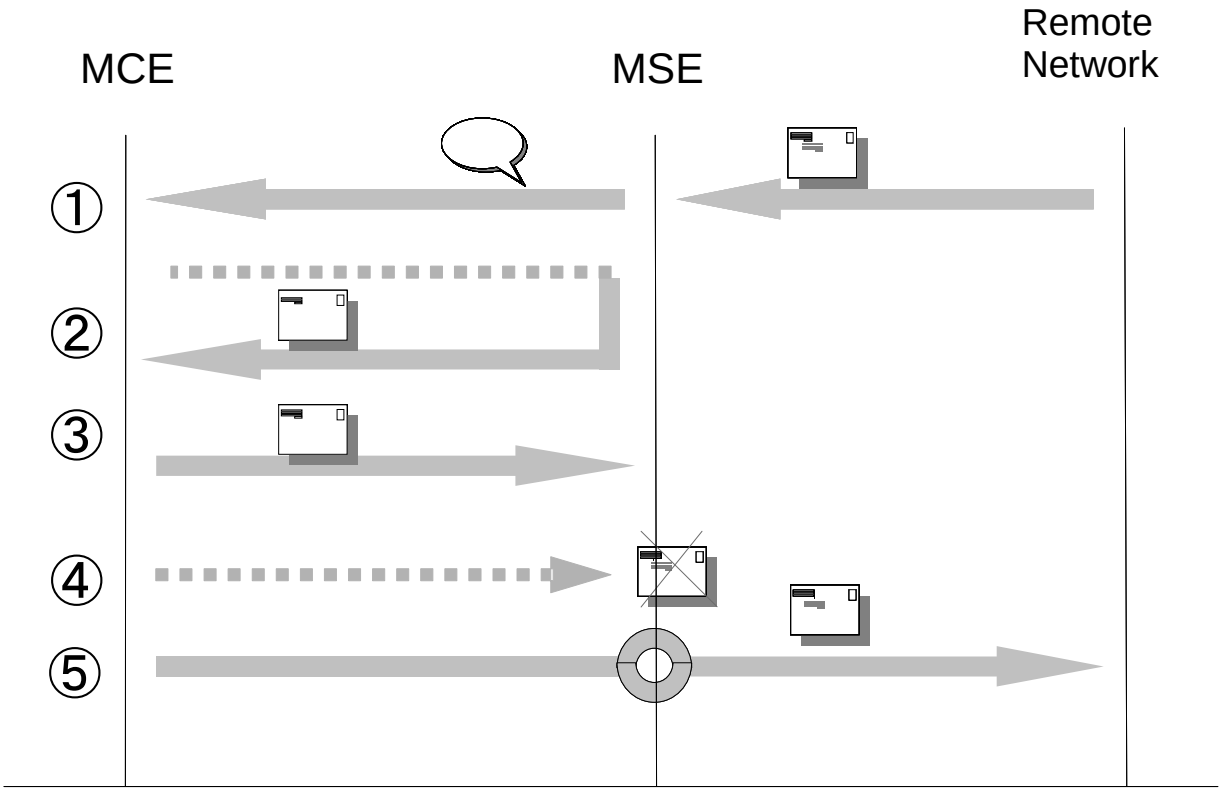

Figure 2.4: Summary of the main scenarios defined by MAP

2.4 Message Types

The following message types are supported by this profile:

- EMAIL: emails on RFC2822 or MIME type basis
- SMS: short messages for cellular or other networks [12], [14]
- MMS: 3GPP MMS messages [13]

These terms are used in the rest of this document to designate these roles.

The figures below show typical configurations of devices for which the Message Access Profile is applicable:

2.5 Profile Fundamentals

The MCE device shall use the services of the MSE device only after successfully creating a secure connection. This includes exchanging of security initialization messages, creation of link keys, and enabling encryption. Security mode 2 or 3 shall be supported for devices implementing the Bluetooth 2.0+EDR, or earlier, specification. Security mode 4 shall be supported for devices implementing the Bluetooth 2.1+EDR, or later, specification.

Either the MSE or MCE may initiate bonding. At a minimum, the MSE shall support Inquiry and Paging and the MCE shall support Inquiry Scan and Page Scan in order to initiate bonding. Either device can initiate the link establishment.

Only the MCE can start a MAP session on OBEX level (OBEX services MAS and MNS, see also chapter 6).

2.6 Bluetooth Security

The two devices shall create a secure connection using the GAP authentication procedure as described in the Generic Access Profile [2]. This procedure shall include alternatively entering a Bluetooth Passkey code for legacy devices or Secure Simple Pairing and will include creation of link keys. For the first case, a fixed passkey may also be used during the GAP bonding procedure.

The Message Access Profile mandates the use of several Bluetooth security features:

Bonding: The MCE and MSE shall be bonded before setting up a Message Access Profile connection. Security Mode 4 shall be used if supported both by MSE and MCE. Each of the four association models of Secure Simple Pairing may be used (see [21], chapter 5 and [22], chapter 7) where it is recommended to use authenticated link keys, i.e. using the association models 'Numeric Comparison', 'Out Of Band' or 'Passkey Entry'.

For legacy devices either security mode 2 or 3 shall be used for the Message Access Profile connection.

Encryption: The link between MCE and MSE shall be encrypted using Bluetooth encryption.

Bluetooth Passkey: When using security mode 2 or 3, the MCE and MSE shall prohibit the use of a zero-length Bluetooth passkey.

Furthermore, the following issues are mandated for devices complying with the Message Access Profile:

Link keys: Combination keys shall be used for Message Access Profile connections.

Encryption key length: The length of the encryption key should be at least 64 bits. For increased security, use of the maximum length allowed given regional regulation is encouraged.

2.7 Conformance

If conformance to this profile is claimed, all capabilities indicated as mandatory for this profile shall be supported in the specified manner (process mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory, optional and conditional capabilities, for which support is indicated, are subject to verification as part of the Bluetooth certification program.

3 Application Layer

3.1 Message Access Profile Objects and Formats

3.1.1 Handle

When exchanging message listings or message objects, the MCE and MSE shall use locally unique identifiers, called handles, to identify individual messages. The MSE device shall assign a handle to each message. The handle shall be a 64 bit unsigned integer whose value is defined by the MSE.

Each handle shall be locally unique across all messages on the MSE. This means that the handle, when presented to the MCE, shall unambiguously identify one and only one message. If a MCE is connected to two MSEs, it is responsible for keeping track of the handles per MSE, as in such case two identical handles may exist each in a different context.

Each handle shall be valid during the entire duration of a MAP session. After the end of a MAP session e.g., initiated by the user or caused by a link loss, the handle is not guaranteed to be valid any more. Thus, a MCE shall not re-use a handle obtained in a previous MAP session.

A MAP session starts with the establishment of a Message Access service connection. The duration of the MAP session shall be defined as the time during which at least one Message Access Service connection or Message Notification Service connection is ongoing (see also MAP OBEX definitions chapter 6.1).

3.1.2 Character-Set

The character set used for the attributes of the Message Access Profile objects bMessage, Folder-Listing, Messages-Listing and Event-Report (see following chapters) shall be UTF-8.

Special charset and encoding conventions are defined for the bMessage-parameter <bmessage-body-content> containing the message object itself (see 3.1.3).

3.1.3 Message format (x-bt/message)

Exchanged messages shall use the bMessage format. The bMessage object encapsulates the delivered message objects and provides additionally a suitable set of properties with helpful information. The general encoding characteristics as defined for vCards in chapter 2 of [5] shall be applied. The formal BNF definition of the bMessage format is as follows:

```
<bmessage-object>::= {
       "BEGIN:BMSG" <CRLF>
       <bmessage-property>
       [<bmessage-originator>]*
       <bmessage-envelope>
       "END:BMSG" <CRLF>
}
<bmessage-property>::=<bmessage-version-property>
      <bmessage-readstatus-property> <bmessage-type-property>
      <bmessage-folder-property>
<bmessage-version-property>::="VERSION:"
      <common-digit>*"."<common-digit>* <CRLF>
<bmessage-readstatus-property>::="STATUS:" 'readstatus' <CRLF>
<bmessage-type-property>::="TYPE:" 'type' <CRLF>
<bmessage-folder-property>::="FOLDER:" 'foldername' <CRLF>
<bmessage-originator>::= <vcard> <CRLF>
<bmessage-envelope> ::= {
       "BEGIN:BENV" <CRLF>
       [<bmessage-recipient>]*
       <bmessage-envelope> | <bmessage-content>
       "END:BENV" <CRLF>
}
<bmessage-recipient> ::= <vcard> <CRLF>
<bmessage-content>::= {
       "BEGIN:BBODY"<CRLF>
       [<bmessage-body-part-ID> <CRLF>]
       <bmessage-body-property>
       <bmessage-body-content>* <CRLF>
       "END:BBODY"<CRLF>
}
<bmessage-body-part-ID>::="PARTID:" 'Part-ID'
<bmessage-body-property>::=[<bmessage-body-encoding-property>]
      [<bmessage-body-charset-property>]
      [<bmessage-body-language-property>]
      <bmessage-body-content-length-property>
<bmessage-body-encoding-property>::="ENCODING:"'encoding' <CRLF>
<bmessage-body-charset-property>::="CHARSET:"'charset' <CRLF>
<bmessage-body-language-property>::="LANGUAGE:"'language' <CRLF>
<bmessage-body-content-length-property>::=
      "LENGTH:" <common-digit>* <CRLF>
<bmessage-body-content>::={
       "BEGIN:MSG"<CRLF>
       'message'<CRLF>
       "END:MSG"<CRLF>
}
```

The following conventions shall be applied for the bMessage properties:

bmessage-version-property:

The value for this property shall be "VERSION:1.0 <CRLF>", which is the present bMessage version 1.0.

bmessage-originator:

This property includes a vCard identifying the originator i.e. the original sender of the message (see also definition 'vCard' property below).

bmessage-recipient:

This property includes a vCard identifying the recipient of the messages (see also definition 'vCard' property below).

In the case of an email, all the recipients in the MIME 'to:' field shall have their own bmessage-recipient property. When sending a message the MSE shall also parse the MIME 'cc:' & 'bcc:' fields and apply the correct behavior for those recipients as defined in RFC2822.

bmessage-readstatus-property:

The property value shall be either "READ" or "UNREAD", Indicating whether a message has been read or not by the MCE. During a MAP session, the MCE shall be responsible for setting this status (see chapters 4.2 and 5.7). A MSE shall not change this status during a MAP session, without initiation by the user or a connected MCE device). The initial status when starting a MAP session is 'read' if the message has already been read on the MSE.

bmessage-type-property:

The property value shall be either

- "EMAIL" for emails on RFC2822 or MIME type basis or
- "SMS_GSM" for GSM short messages [12] or
- "SMS_CDMA" for CDMA short messages [14] or
- "MMS" for 3GPP MMS messages [13].

bmessage-folder-property:

The folder name including the path where the bMessage is located in. Any folder- within the MSE folder structure may be used for this property as described 3.1.4 (e.g., "telecom/msg/inbox"). This property shall be restricted to 512 bytes. If the path and folder description exceeds this length the name of the addressed folder and as far as possible the folder layers above shall be delivered. If the bMessage is used for an upload (PushMessage function, see 5.8) the MCE should send an empty bmessage-folder-property and the MSE should discard this property to avoid redundancy with the ‘name’ header’ of the PushMessage function.”

bmessage-body-part-ID:

This property shall be used if and only if the content of the related message can't be delivered completely within one <bmessage-content> object, i.e. in case of a fragmented email. The part-ID of the first <bmessage-content> object shall be 0, the following <bmessage-content> objects shall have a part-ID incremented by 1 each. The range of this value shall be 0 to 65535, accordingly restricting the number of fractions to this maximum.

bmessage-body-charset-property:

The character set used by the message object contained by <bmessage-body-content>. This property shall be used only if the message contains textual content. For detailed requirements, see description of the <bmessage-body-content property> below.

bmessage-body-encoding-property:

The encoding used by the message object contained by <bmessage-body-content>. The property shall have one of the values

- For Email/MMS [9], [13]: "8BIT" (for 8-Bit-Clean encoding).
- For GSM-SMS [18]: "G-7BIT" (GSM 7 bit Default Alphabet), "G-7BITEXT" (GSM 7 bit Alphabet with national language extension), "G-UCS2" and "G-8BIT"
- For CDMA-SMS [17]: "C-8BIT" (Octet, unspecified), "C-EPM" (Extended Protocol Message), "C-7ASCII (7-bit ASCII), "C-IA5" (IA5), "C-UNICODE" (UNICODE), "C-SJIS" (Shift-JIS), "C-KOREAN" (Korean), "C-LATINHEB (Latin/Hebrew) and C-LATIN (Latin)

For detailed requirements, see description of the <bmessage-body-content> property below.

bmessage-body-language-property:

The language of the message. This property may be used if the message includes textual content. The property shall have one of the values:

- For GSM-SMS [18]: " TURKISH", " SPANISH", "PORTUGUESE", "UNKNOWN" Referenzen
- For CDMA-SMS [17]:, "ENGLISH", "FRENCH", "SPANISH", "JAPANESE", "KOREAN", "CHINESE", "HEBREW", "UNKNOWN"

vCard:

The allowed properties for a vCard [5], [7] in a bMessage shall be the following:

For version 2.1:

VERSION, N, TEL, EMAIL (VERSION and N shall be included, TEL and EMAIL may be used)

For version 3.0:

VERSION, N, FN, TEL, EMAIL (VERSION, N, and FN shall be included, TEL and EMAIL may be used)

Furthermore, the 'recipient' attribute of bMessages delivered by the MCE to the MSE for sending shall include:

- The EMAIL property in case of an email
- The TEL property in case of an SMS
- Exactly one of the these properties in case of an MMS

All the other vCard properties shall not be used. The properties may be empty if not known, e.g., N/FN in case of a SMS.

bmessage-envelope:

The maximum level of <bmessage-envelope> encapsulation shall be 3. If the message originally received by the MSE has more than 3 encapsulation levels, the MSE shall deliver the upper 3 levels, i.e., the most recent ones.

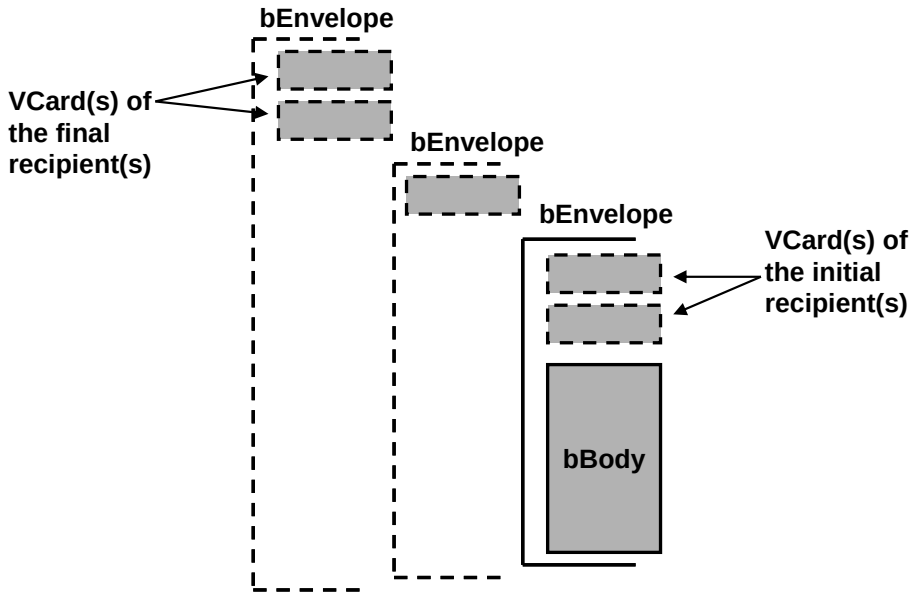

Figure 3.1: bEnvelopes encapsulation in a bMessage

bmessage-body-content-length-property:

Overall length of the related <bmessage-body-content> objects(s) in bytes, where the length counting starts with the "B" of the first occurrence of "BEGIN:MSG" and ends with the <CRLF> of the last occurrence of "END:MSG"<CRLF>.

bmessage-body-content:

The <bmessage-body-content> shall be constructed according to the type of message contained therein as indicated by the mandatory <bmessage-type-property>. The conversion shall be done by the MSE according to the following requirements:

- When generating a bMessage, all occurrences of <CRLF>'END:MSG' in a <bmessage-bodycontent>, after any encoding or encapsulation, shall be replaced by <CRLF>'/END:MSG'. Occurrences of <CRLF>'/END:MSG' shall be replaced by <CRLF>'//END:MSG', and so on. Adding an escape character shall increase the LENGTH field by 1. Similarly, when parsing a <bmessage-body-content>, all occurrences of <CRLF>'/END:MSG' shall have a '/' removed before any further decoding or processing.
- If the message is a MMS (bMessage-type "MMS"):
  - If the bMessage is uploaded from the MCE to the MSE the <bmessage-body-content> shall be formatted as an email specified in RFC2822 and MIME RFC2045-RFC2047.
  - If the message is downloaded by the MCE from the MSE the original MMS message shall be converted into email format following the RFC4356 specification, before being used as <bmessage-body-content>.

  - Furthermore, for the <bmessage-body-content> the requirements described hereunder for the bMessage-type "EMAIL" shall be applied.

- If the message is an email (bMessage-type "EMAIL"):or a MMS converted to email format as described above (bMessage-type "MMS"):
  - The <bmessage-body-content> contains the message formatted as specified in RFC2822 and MIME RFC2045-RFC2047
  - The message object contained in <bmessage-body-content> shall be coded using the following rules:
    - If the message or – in case of a multipart-message – a part of the message includes textual content (MIME type/subtype = text/xxx), the contained message text shall be coded in UTF-8 charset:
      - The related <bmessage-body-charset-property> shall be "UTF-8".
      - If the bMessage is downloaded from the MSE to the MCE, the MSE shall trans-code the included text to UTF-8 charset before delivery.
      - If the bMessage is uploaded from the MCE to the MSE, the MSE may trans-code the text of the message from UTF-8 to any charset required, e.g., before sending to the network.
      - If the bMessage is uploaded from the MCE to the MSE, the MSE must add the "From" attribute of the included email before sending if it's not present in the email. This situation may occur if the MCE doesn't know this email address.”
    - If the message or – in case of a multipart-message – all parts of the message include only non-textual content (MIME type/subtype other than text/…)
      - The contained message or the message part shall not be trans-coded to UTF-8.
      - The related <bmessage-body-charset-property> shall not be used.
    - In any case the encoding method used for the <bmessage-body-content> shall be 8-Bit:
      - No other encoding methods ((e.g., QuotedPrintable or Base64) shall be used.
      - Accordingly, the only authorized value of the related <bmessage-body-encoding-property> shall be "8BIT".
      - If the bMessage is downloaded from the MSE to the MCE, the MSE shall trans-code it to 8-Bit encoding before it is delivered.
      - If the bMessage is uploaded from the MCE to the MSE, the MSE may trans-code the message encoding from 8-Bit encoding to any encoding required, e.g., before sending to the network.
  - If the original message is a GSM-SMS or a CDMA-SMS (bMessage-types"SMS_GSM" or "SMS_CDMA"):

- Only SMS related to the following two SMS-PDUs [12], [14], [23] shall be used by MAP:
  - SMS-Deliver PDUs received from the network
  - SMS-Submit PDUs for submission to the network
- Two kinds of formats for the <bmessage-body-content> are supported for the transmission of bMessages between MCE and MSE. The MSE shall support both formats; the MCE may choose either of these formats (see also definitions of functions getMessage and pushMessage, chapter 5):
  1) <bmessage-body-content> includes the text of the SMS coded in UTF-8:
     - This kind of bMessage coding may be used when the transmitted message includes textual content (e.g., 7-bit or UCS-2 encoding for GSM). It shall not be used if the message includes only binary content.
     - The bMessage shall contain exactly one <bmessage-body-content> object which includes a text string with the message text coded in UTF-8 charset.
     - Accordingly, the related <bmessage-body-charset-property> shall be "UTF-8".
       The <bmessage-body-encoding-property> shall not be used.
     - If the bMessage is downloaded from the MSE to the MCE, the MSE shall trans-code the textual parts of the related SMS-Deliver PDU to UTF-8 charset before delivery.
     - If the bMessage is uploaded from the MCE to the MSE for sending, the MSE shall use the UTF-8 coded text in <bmessage-body-content> and the other parameters delivered by the related <bmessage-object> to produce a SMS-Submit PDU or – if the delivered text is too long – a concatenated SMS PDUs.
  2) SMS PDU object in <bmessage-body-content>:
     - This kind of bMessage coding shall be used for SMS with binary content and may be used for SMS with textual content. The <bmessage-body-charset-property> shall not be used.
     - In case of a non-concatenated SMS (single PDU) the bMessage shall contain exactly one <bmessage-body-content> object that includes the SMS PDU.
     - In the case of GSM-SMS: a GSM 04.11 [23] SC address followed by GSM 03.40 [12] TPDU in hexadecimal format: MSE/MCE converts each octet of Transfer Layer Protocol data unit into a two-character, IRA [24] encoded, hexadecimal

number (e.g., octet with decimal value 42 is presented to TE as a two-character hexadecimal value, 2A. The IRA encodings being decimal 50 and 65, or hexadecimal 32 and 41).

- In the case of CDMA-SMS: a 3GPP2 C.S0015-0 [14] Transport Layer Message in hexadecimal format: MSE/MCE converts each octet of Transfer Layer Protocol data unit into a two-character, IRA [24] encoded, hexadecimal number (e.g., octet with decimal value 42 is presented to TE as a two-character hexadecimal value, 2A. The IRA encodings being decimal 50 and 65, or hexadecimal 32 and 41).
- In case of a concatenated SMS (multiple PDU) the bMessage shall contain several <bmessagebody- content> objects, each with a PDU of the concatenated SMS. If the bMessage is uploaded from the MCE to the MSE for sending, the MCE shall order the contained <bmessage-body-content> objects as required for sending. If the bMessage is downloaded by the MCE from the MSE, the MSE shall order the contained <bmessage-body-content> objects in the same way like the SMS PDUs of the original concatenated SMS.

Hereunder is an example of a bMessage of the type EMAIL with status UNREAD:

```
BEGIN:BMSG
      VERSION:1.0
      STATUS:UNREAD
      TYPE:EMAIL
      FOLDER:TELECOM/MSG/INBOX
      BEGIN:VCARD
            VERSION:2.1
            N:Mat
            EMAIL:ma@abc.edu
      END:VCARD
      BEGIN:BENV
            BEGIN:VCARD
                  VERSION:2.1
                  N:Tanaka
                  EMAIL:tanaka@def.edu
            END:VCARD
            BEGIN:BENV
                  BEGIN:VCARD
                        VERSION:2.1
                        N:Laurent
                        EMAIL:laurent@ghi.edu
                  END:VCARD
                  BEGIN:BBODY
                        ENCODING:8BIT
                        LENGTH:125
                        BEGIN:MSG
                              Date: 20 Jun 96
                              Subject: Fish
                              From: tanaka@def.edu
                              To: laurent@ghi.edu

                              Let's go fishing!
                              BR, Mat
                        END:MSG
                  END:BBODY
            END:BENV
      END:BENV
END:BMSG
```

Below is an example of a bMessage embedded with a native SMS-Deliver PDU containing the message, “Let’s go fishing!”:

```
BEGIN:BMSG
      VERSION:1.0
      STATUS:UNREAD
      TYPE:SMS_GSM
      FOLDER:TELECOM/MSG/INBOX
      BEGIN:VCARD
            VERSION:2.1
            N:Joachim
            TEL:00498912345678
      END:VCARD
      BEGIN:BENV
            BEGIN:BBODY
                  ENCODING:G-7BIT
                  LENGTH: 96
                  BEGIN:MSG
                        0191000E91009498214365870000011303231
                        12928211CC32FD34079DDF20737A8E4EBBCF21
      END:MSG
            END:BBODY
      END:BENV
END:BMSG
```

3.1.4 Folders Structure

The following is an example of the folder architecture that the MSE presents to the MCE. This folder structure is only the representation used by MAP. The folder structure doesn't necessarily have to exist in the same way on the MSE device.

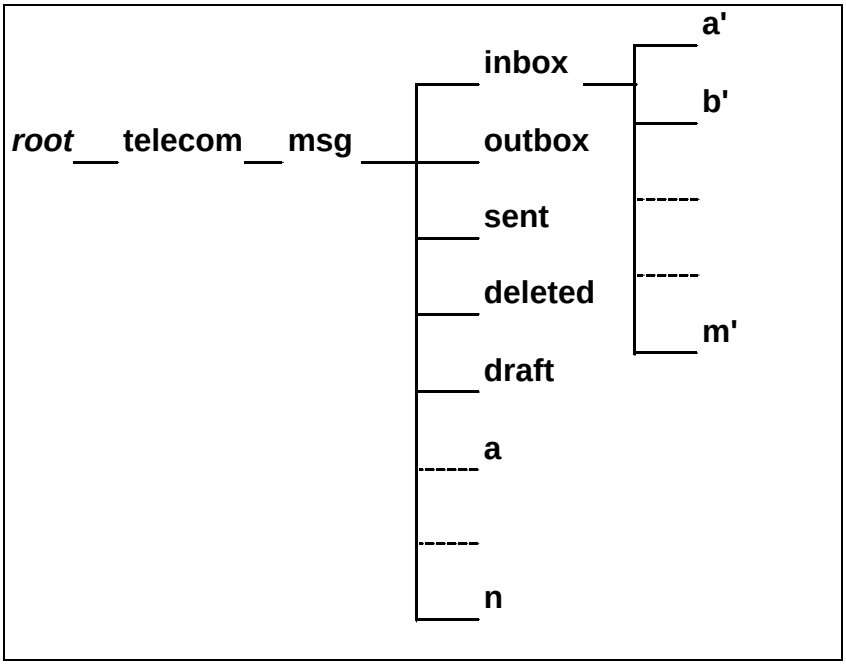

Figure 3.2: MAP virtual folder structure

The following folders shall be present at the MSE's message repository representation (bold characters in Figure 3.2):

- telecom/msg/inbox: This folder shall contain all the messages that have been received by the MSE, at least during the period of the current MAP session, and which not have been shifted to other folders on the MSE device This folder will be called 'Inbox' folder hereinafter.
- telecom/msg/sent: This folder shall contain messages that were successfully sent to the network by the MSE, at least during the period of the current MAP session. In particular, these messages have been queued in the 'Outbox' folder before sending and are shifted by the MSE to this folder after transmission. This folder will be called 'Sent' folder hereinafter.
- telecom/msg/deleted: This folder shall contain the messages that have been deleted on the MSE, at least during the period of the current MAP session. This folder will be called 'Deleted' folder hereinafter.

The following folder shall be present if and only if the MSE device has the capability to transmit messages to a remote network. In particular it shall be present also during temporal disturbances of the transmission process e.g., by network problems.

- telecom/msg/outbox: This folder shall contain all the messages that are queued for transmission by the MSE, at least during the period of the current MAP session. This folder will be called 'Outbox' folder hereinafter. In particular, the MCE shall only upload messages for sending to this folder. If a message has been transmitted to the network, it shall be shifted by the MSE to the 'Sent' folder (sending success). If a message – for any reason – can't be sent, these messages shall remain in the 'Outbox' folder.

The telecom/msg/draft folder is the repository for messages that have not been sent and may require further editing. If a comparable folder is present on the MSE, it shall be presented to the MCE and its name shall be 'Draft'.

In addition to the above fixed folders, the profile also supports user-defined folders.

If folders have a real representation on the MSE device, the folders presented by the MAP-MSE should contain – if possible – the same messages as in the real folder structure; (e.g., the Sent folder), as far this recommendation doesn't conflict with mandatory requirements.

If the MSE device owns more than one message repository, e.g., several email accounts it may present these multiple message repositories by multiple MAS Instances (see also chapters 3.1.8 and 7).

3.1.5 Folder-Listing Object (x-obex/folder-listing)

The Folder-Listing object is defined in 9.1 of [10] and shall be encoded in UTF-8. In the context of the MAP profile, the Folder-Listing object shall not contain message entries. It shall only contain folder entries located in the current folder level.

3.1.6 Messages-Listing Object (x-bt/MAP-msg-listing)

The Messages-Listing object is an XML object and shall be encoded in UTF-8.

The entries of the Messages-Listing object shall be sorted by the value of the "datetime" parameter (see below). Sort order shall be in descending order, the newest message being transferred first.

The Messages-Listing object is defined according to the following DTD (Document Type Definition, [20]):

```
<!DTD for the MAP Messages-Listing Object-->

<!DOCTYPE MAP-msg-listing [

<!ELEMENT MAP-msg-listing ( msg )* >
<!ATTLIST MAP-msg-listing version CDATA #FIXED "1.0">

<!ELEMENT msg EMPTY>
<!ATTLIST msg
     handle CDATA #REQUIRED
     subject CDATA #REQUIRED
     datetime CDATA #REQUIRED
     sender_name CDATA #IMPLIED
     sender_addressing CDATA #IMPLIED
     replyto_addressing CDATA #IMPLIED
     recipient_name CDATA #IMPLIED
     recipient_addressing CDATA #REQUIRED
     type CDATA #REQUIRED
     size CDATA #REQUIRED
     text (yes|no) "no"
     reception_status CDATA #REQUIRED
     attachment_size CDATA #REQUIRED
     priority (yes|no) "no"
     read (yes|no) "no"
     sent (yes|no) "no"
     protected (yes|no) "no"
>
]>
```

where the parameters shall be used as described below:

- "handle" is the message handle in hexadecimal representation with up to 16 digits; leading zero digits may be used so the MCE shall accept both handles with and without leading zeros (e.g.,"00000012345678AB" or "12345678AB").
- "subject" is the summary of the message. This parameter shall not exceed 256 bytes. The content of this parameter is left to the implementer's decision. Typically, in case the original message has a title, the subject should contain the title of the original message. If the original message does not have any title, the implementer may use the first words of the message as "subject" value. The subject may be a blank string ("").
- "datetime" is the timestamp of the message in format "YYYYMMDDTHHMMSS" as defined for the OBEX "time" header in 2.2 of [10]. Local Time time basis and 24 hours representation shall be used. If the message in the MSE internal repository doesn't have a 4-digit year representation, the MSE shall complete it to 4 digits in a

suitable way. If the sending time is included in the message it shall be used for this parameter; otherwise the reception time of the MSE shall be used.

- "sender_name" is the name of the sender of the message, when it is known by the MSE device. This parameter shall not exceed 256 bytes and shall be truncated to the first 255 bytes if required. If a sender name is included in the original message the MSE shall use this information for the content of this attribute. Otherwise, the content of this parameter is left to the implementer's decision, e.g., the MSE may use a phonebook entry for "sender_name".
- "sender_addressing" is the addressing information of the sender. In case of an email this is the sender's email address ("From:" field). In case of a SMS this is the sender's phone number in canonical form (chapter 2.4.1 of [5]). In case of a MMS this is the senders email address or phone number. This parameter shall not exceed 256 bytes. If the sender addressing of the original message is longer than 256 bytes it shall be truncated to the first 256 bytes. If the sender is not known (e.g., in case of a suppressed sender address) this parameter may be left empty.
- "replyto_addressing" is the addressing information for replies to the sender. This parameter shall be used only for emails to deliver the sender's reply-to email address ("Reply-To:" field). This parameter shall not exceed 256 bytes. If the reply-to addressing of the original message is longer than 256 bytes it shall be truncated to the first 256 bytes. If the reply-to address is not present in the email this parameter shall be left empty.
- "recipient_name" is the name of the recipient of the message, when it is known by the MSE device. This parameter shall not exceed 256 bytes and shall be truncated to the first 256 bytes if required.
- "recipient_addressing" is the addressing information of the recipient. In case of an email this is the recipient's email address or a list of email addresses, each delimited by ";" (ASCII x3B). In case of a SMS this is the recipient's phone number in canonical form (chapter 2.4.1 of [5]). In case of a MMS this is the recipient's email address or phone number. This parameter shall not exceed 256 bytes. If the recipient is not known (e.g., in case of a draft message) this parameter may be left empty. If the recipient addressing of the original message is longer than 256 bytes it shall be truncated to the first 256 bytes.
- "type" gives the type of the message. The following types are supported by the present profile:
  - "EMAIL": e-mail RFC2822 or MIME RFC2045-47 type basis
  - "SMS_GSM": GSM short message
  - "SMS_CDMA": CDMA short message
  - "MMS": 3GPP MMS

At least one of these types shall be supported by the MSE.

- "reception_status" gives the status of reception of the message. The parameter shall have one of the values:

  - "complete": complete message has been received by the MSE
  - "fractioned": only a part of the message has been received by the MSE (e.g., fractioned email of push-service)
  - "notification": only a notification of the message has been received by the MSE
- "size" is the overall size in bytes of the original message as received from network. The MCE implementation has to take into account that the size of the message delivered in a bMessage may be different from this size due to trans-coding effects (see 3.1.3).
- "attachment_size": The overall size of the attachments in bytes. Value = 0 indicates that the message has no attachments.
- "text": Value "yes" indicates the original message or – in case of multipart-messages – a part of the message includes textual content, "no" indicates that the message has no textual but only binary content.
- "read": Value "yes" indicates that the message has already been read on the MSE, "no" indicates that the message has not yet been read (see also bmessage-readstatus-property in 3.1.3).
- "sent": Value "yes" indicates that the message has already been sent to the recipient, "no" indicates that the message has not yet been sent.
- "protected": Value "yes" indicates that the message or a part of the message (e.g. attachment) is protected by a DRM scheme, "no" indicates that the message is not protected by DRM.
- "priority": Value "yes" indicates that the message is of high priority, "no" indicates that the message is not of high priority.

The parameters above shall not occur in the Messages-Listing object if its delivery is suppressed actively by the parameter-mask attribute in the GetMessagesListing function (see 5.5). In particular also Messages-Listing parameters marked as REQUIRED in the DTD above may be masked out.

Example of Messages-Listing object:

```
<MAP-msg-listing version = "1.0">
  <msg handle = "20000100001" subject = "Hello" datetime =
"20071213T130510" sender_name = "Jamie" sender_addressing = "+1-987-
6543210" recipient_addressing = "+1-0123-456789" type = "SMS_GSM"
size = "256" attachment_size = "0" priority = "no" read = "yes" sent =
"no" protected = "no"/>
  <msg handle = "20000100002" subject= "Guten Tag" datetime =
"20071214T092200" sender_name = "Dmitri" sender_addressing = "8765432109"
recipient_addressing = "+49-9012-345678" type = "SMS_GSM" size = "512"
attachment_size = "3000" priority = "no" read = "no" sent = "yes"
protected = "no"/>
  <msg handle = "20000100003" subject = "Ohayougozaimasu" datetime =
"20071215T134326" sender_name = "Andy" sender_addressing = "+49-7654-
321098" recipient_addressing = "+49-89-01234567" type = "SMS_GSM" size =
"256" attachment_size = "0" priority = "no" read = "yes" sent = "no"
protected = "no"/>
  <msg handle = "20000100000" subject = "Bonjour" datetime =
"20071215T171204" sender_name = "Marc" sender_addressing =
"marc@carworkinggroup.bluetooth" recipient_addressing =
"burch@carworkinggroup.bluetooth" type = "EMAIL" size = "1032"
attachment_size = "0" priority = "yes" read = "yes" sent = "no" protected
= "yes"/>
</MAP-msg-listing>
```

3.1.7 MAP-Event-Report object

The MAP-Event-Report object is an XML object including the description of exactly one event. The MAP-Event-Report object shall be encoded in UTF-8.

The MAP-Event-Report object is defined according to the following DTD:

```
<!DTD for the OBEX MAP-Event-Report object-->

<!DOCTYPE MAP-event-report [

<!ELEMENT MAP-event-report ( event ) >
<!ATTLIST MAP-event-report version CDATA #FIXED "1.0">

<!ELEMENT event EMPTY>
<!ATTLIST event
     type CDATA #REQUIRED
     handle CDATA #IMPLIED
     folder CDATA #IMPLIED
     old_folder CDATA #IMPLIED
     msg_type CDATA #IMPLIED
>
]>
```

Where the parameters shall be used as described below:

- "type" shall have one of the following values:
  1. "NewMessage": indicates that a new message has been received by the MSE device. This event type shall also be used if a new message replaces an old one on the MSE (e.g., a Replace-SMS [12]). In this case the MSE shall send the

NewMessage including the same handle as used for the replaced message where the readstatus-property of the related bMessage object shall be set to "UNREAD".

2. "DeliverySuccess": indicates that a message has been successfully delivered to its intended recipient.
3. "SendingSuccess": indicates that a message has been successfully sent to a remote network.
4. "DeliveryFailure": indicates that the delivery of a message to its intended recipient failed. This event type shall not be applied for emails. It should be reported not more than once for a message.
5. "SendingFailure": indicates that sending to the remote network failed. The MSE may send this several times for one message if the 'Retry' application parameter of the PushMessage function (section 5.8) has been set to 'ON' by the MCE, i.e. each time the MSE's delivery to the network has failed.
6. "MemoryFull": indicates that the memory of the MSE device is full and will not allow any new arriving messages to be received and stored. This event should be sent if the memory becomes full during the MAP session and at the setup of the MNS connection if the memory is already full.
7. "MemoryAvailable": indicates that the memory of the MSE device is ready again to receive new messages. This event shall be sent only if there was a "MemoryFull" event before.
8. "MessageDeleted": indicates that a message has been deleted from the reported folder on the MSE and has therefore been shifted to the 'Deleted' folder.
9. "MessageShift": indicates that a message has been shifted on the MSE from one folder (indicated by the "old_folder" parameter) to another folder (indicated by the "folder" parameter).

- "handle" is the handle of the message that the "type" indication refers to. For a NewMessage event, "handle" is the handle that was assigned by the MSE device to the newly received message. For a DeliverySuccess, a SendingSuccess, a DeliveryFailure or a SendingFailure, "handle" is the handle of the message that was successfully or unsuccessfully delivered or sent. "handle" is not used when the event "type" is "MemoryFull" or "MemoryAvailable".
- "folder" shall be the name of the folder including the path in which the corresponding message has been filed by the MSE (e.g., "TELECOM/MSG/INBOX", see also 3.1.4) This parameter shall be restricted to 512 Bytes. If the path and folder description exceeds this length the name of the addressed folder and as far as possible the folder layers above shall be delivered. The "folder" parameter shall not be used when the event "type" is "MemoryFull" or "MemoryAvailable".
- "old_folder" shall be used only in case of a message shift to indicate the folder on the MSE from which the message has been shifted out. The same formatting conventions as for the "folder" parameter shall be fulfilled.

- "msg_type" gives the type of the message. "msg_type" is not used when the event "type" is "MemoryFull" or "MemoryAvailable". The following types are supported by the present profile:
  - "EMAIL": e-mail RFC2822 or MIME type basis
  - "SMS_GSM": GSM short message
  - "SMS_CDMA": CDMA short message
  - "MMS": 3GPP MMS

Example of Message Event object

```
<MAP-event-report version = "1.0">
<event type = "NewMessage" handle = "12345678" folder = "TELECOM/MSG/INBOX"
msg_type = "SMS_CDMA" />
</MAP-event-report>
```

3.1.8 MSE Instances

MAP MSE devices may present one or several MAS Instances to the MCE, each providing the overall MAP MSE Server functionality. For each MAS Instance there shall be exactly one MAS-Server SDP record (see also chapter 7) and a MCE can access to each MAS Instance by a dedicated OBEX connection. Figure 3.3 gives an example about a potential configuration:

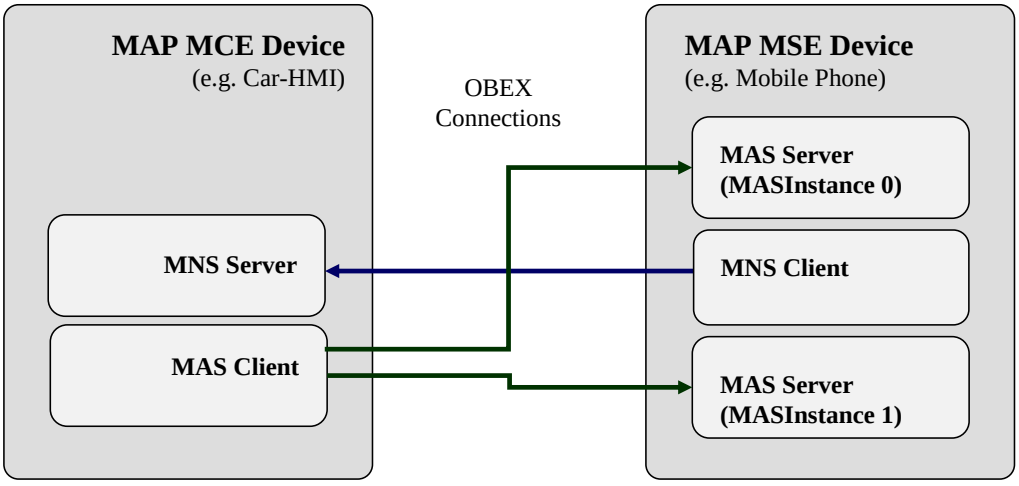

Figure 3.3: Example for two MAS Instances and its connections

The following requirements have to be considered:

- MSE device
  - MAP is capable to differentiate in one MAS Instance between Email, SMS and MMS, so an MSE device – taking into consideration the recommendations below – should present its messages by one MAS Instance to minimize the number of OBEX connections required. This MAS Instance shall at least include the SMS and MMS repository of the MSE device, if these message types are supported, and may include emails. For this MAS Instance the value of the related SDP record attribute 'MASInstanceID' shall be 0. The MSE device may present its email repository by an additional MAS Instance.
  - If an MSE device provides more than one email-account it should present additional MAS Instance(s) for each of these email accounts to enable the MCE differentiation of the email messages of particular accounts.
  - If an MSE device provides more than one subscriber number (e.g., two active SIM cards) it should present additional MAS Instance(s) for each of these subscriber numbers to enable the MCE a differentiation of the SMS/MMS messages of particular subscriber numbers.
  - The SDP record attribute 'ServiceName' should enable an easy recognition of the related message repository on the MSE device by the user. Nevertheless, as the SDP records may be accessible without pairing, the 'ServiceName' should not expose any personal information, e.g., a complete phone number or email address with respect to privacy protection.
  - The MSE device shall enable a simultaneous and independent OBEX connection to all provided MAS instances.
  - To enable the MCE to re-connect to a once selected MAS-Instance without user interaction an MSE-Device should keep the MASInstanceIDs in the SDP record constant (see section **Error! Reference source not found.**) and should not change it for different MAP sessions. If a MASInstance is changed or removed (e.g., related message account deleted) the MSE device should not re-use this MASInstanceID until the maximum ID-number (255) has been reached. In the latter case the MSE should restart numbering the MASInstanceID from 0, using the smallest available free number.
  - For the same reason, the MSE Device should not change the ServiceName for different MAP sessions as long as the user does not change the account name on the MSE.
- MCE device
  - An MCE may connect via OBEX to one, several or all MAS Instances of an MSE device.
  - Depending on the implementation this connection may be automated or may require user interaction. In particular the MCE may connect without user interaction if

  - there is only one MAS Instance,
  - the MCE connects generally to all MAS Instances, e.g., in case of a watch as a 'Notification Client' signaling all incoming messages,
  - the MCE connection is done on basis of the message type presented by the SDP record attribute ' SupportedMessageTypes', e.g., connect to (all) SMS repository(s).

- If the MCE cannot select a MAS Instance automatically the MCE may present the user in an appropriate way a list of the MAS ServiceNames for selection.

4 Message Access Profile Features

The current profile is composed of five features. For a device to comply with this specification, it shall observe the following implementation requirements table:

Feature	Support by the MCE	Support by the MSE
Message Notification	C1	M
Message Browsing	C1	M
Message Uploading	O	M
Message Delete	O	M
Notification Registration	C2	M

Table 4.1: MAP features

C1: The MCE shall support at least one of the C1-labelled features

C2: The MCE shall support Message Notification Registration if it supports Message Notification. Not applicable otherwise.

Table 4.2 maps the scenarios described in section 2.4 to the MAP features.

	MAP Feature⇒ **Scenario**	**Message Notification**	**Message Browsing**	**Message Uploading**	**Message Delete**	**Notification Registration**
1	Notifying the MCE of the arrival of a new message	✓	N A	N A	N A	✓
2	Browsing messages from the MCE	N A	✓	N A	N A	N A
3	Uploading messages to the MSE	N A	N A	✓	N A	N A
4	Deleting messages from the MSE	N A	N A	N A	✓	N A
5	Sending messages through the MSE	✓	N A	✓	N A	N A

Table 4.2: MAP scenarios and features

Both scenario 3 and 5 use the Message Uploading feature. In the case of scenario 5, the messages shall be uploaded into the 'Outbox' folder of the MSE. In the case of scenario 3, the messages shall not be uploaded into the 'Outbox' folder of the MSE, but any other folder is legal.

Scenario 1 and 5 are using the Message Notification feature. In the case of scenario 5, the MCE shall get a notification when the messages has been sent to the network and thus has been shifted by the MSE from the 'Outbox' to the 'Sent' folder.

4.1 Message Notification Feature

If support for the Notification feature is claimed by the MCE this feature allows the MSE to send event reports to the MCE, e.g message arrival notifications:

Function	Support by the MCE	Support by the MSE
SendEvent	M	M

The MSE shall notify the MCE about any change of a Messages-Listing of any folder on the MSE side if it not has been initiated by the MCE itself. For example, if the user deletes a message on the MSE-device directly this requires a message deletion notification for the related folder, whereas if the user deletes a message via the MCE no notification shall be sent.

In case of a message reception from the network the MSE shall send the related notification of a message arrival only if the MSE has received a complete message. (e.g., last part of a concatenated SMS). Since some push email services will only push a fraction of the email to the email client (e.g., email header) the MSE may, for emails, send the notification after receiving only a fraction.

The mode of the Notification may be configured by the MCE using the Notification Registration feature (see 4.5). The default mode for the Notification is "off", i.e. the notification is inactive at the MSE if the MAP session has been started. If the MNS session is terminated by either device, then the message notification mode shall be set to default mode on MSE, i.e., the notification is inactive.

A function sequence for the Message Notification feature is illustrated hereunder (see also chapter 5 for definitions of related functions):

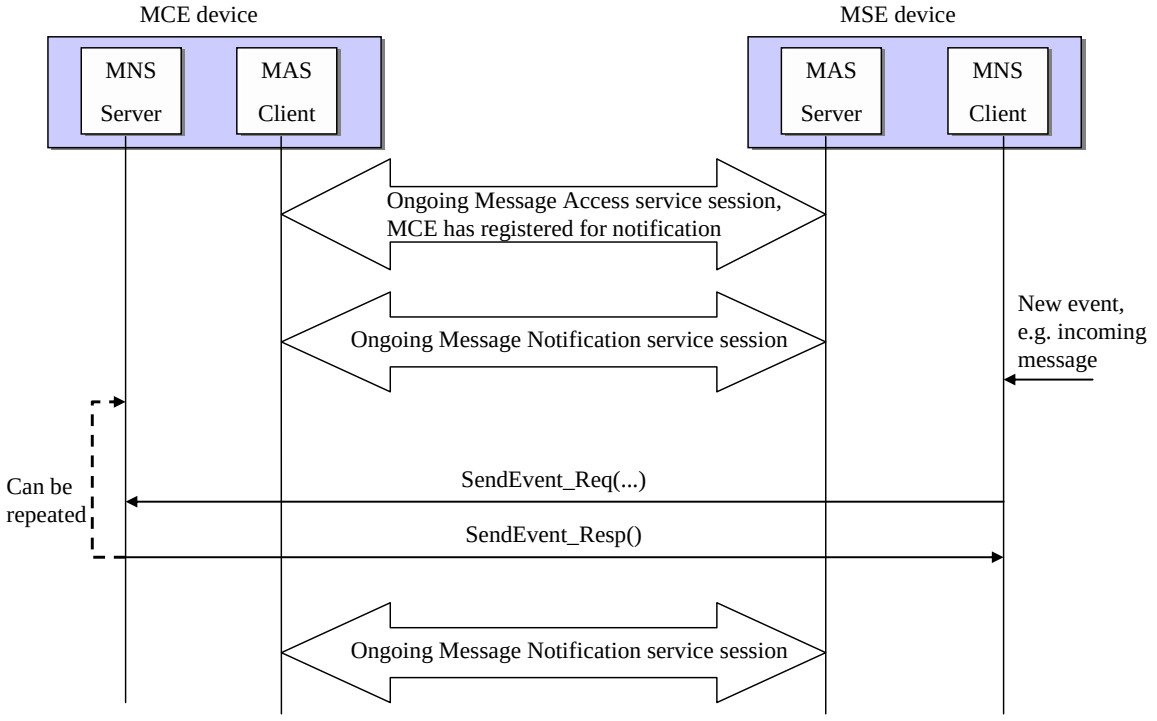

Figure 4.1: Sequence for the Message Notification feature

4.2 Message Browsing Feature

This feature allows the MCE to browse messages on the MSE and to retrieve individual messages from the MSE:

Function	Support by the MCE	Support by the MSE
UpdateInbox	O	M
SetFolder	M	M
GetFolderListing	M	M
GetMessagesListing	M	M
GetMessage	O	M
SetMessageStatus	O	M

A typical function sequence for the Message Browsing feature is illustrated hereunder (see also chapter 5 for definitions of related functions):

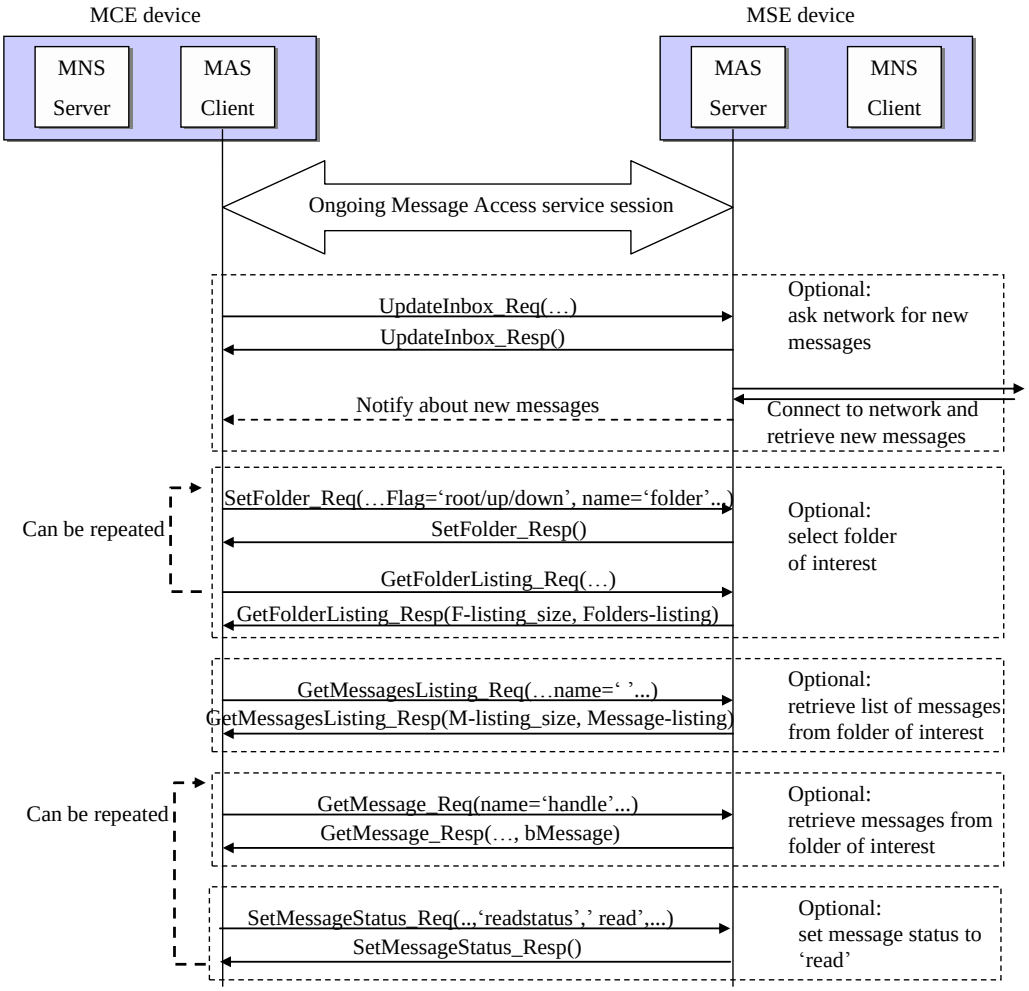

Figure 4.2: Typical sequence for the Message Browsing feature

In case of a 'GetMessage' applied on a notification-message (e.g., MMS-Notification or email Notification) the MSE shall not deliver the notification-message but shall retrieve

the corresponding message from the network and deliver it with the same message handle as used before for the message-notification. The required sequence is shown hereunder:

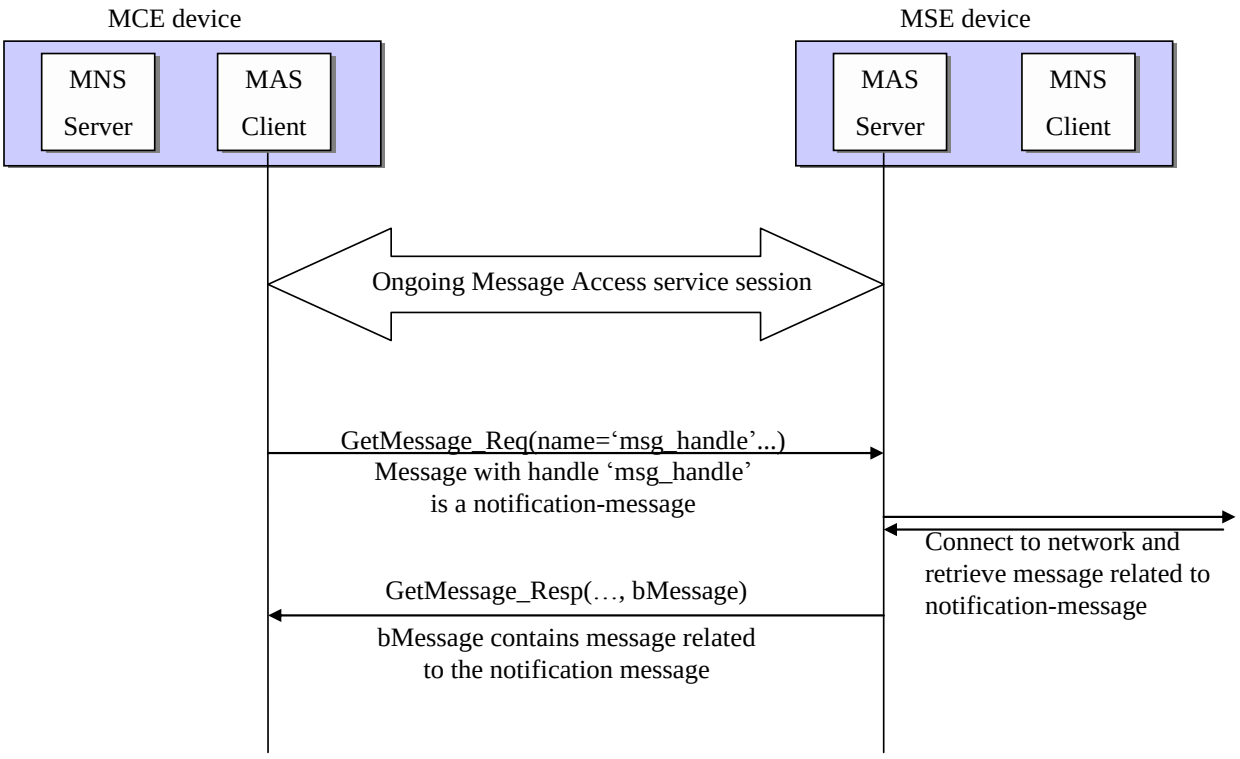

Figure 4.3: Typical sequence for 'GetMessage' applied on a notification-message

In case of a 'GetMessage' applied on a message which is not stored completely on the MSE (e.g., in case of a fragmented email as used by some Push-Mail services) the content of the related message can't be delivered completely within one 'GetMessage' response. In this case several requests/responses have to be performed (see also chapter 5.6). The required sequence is shown hereunder:

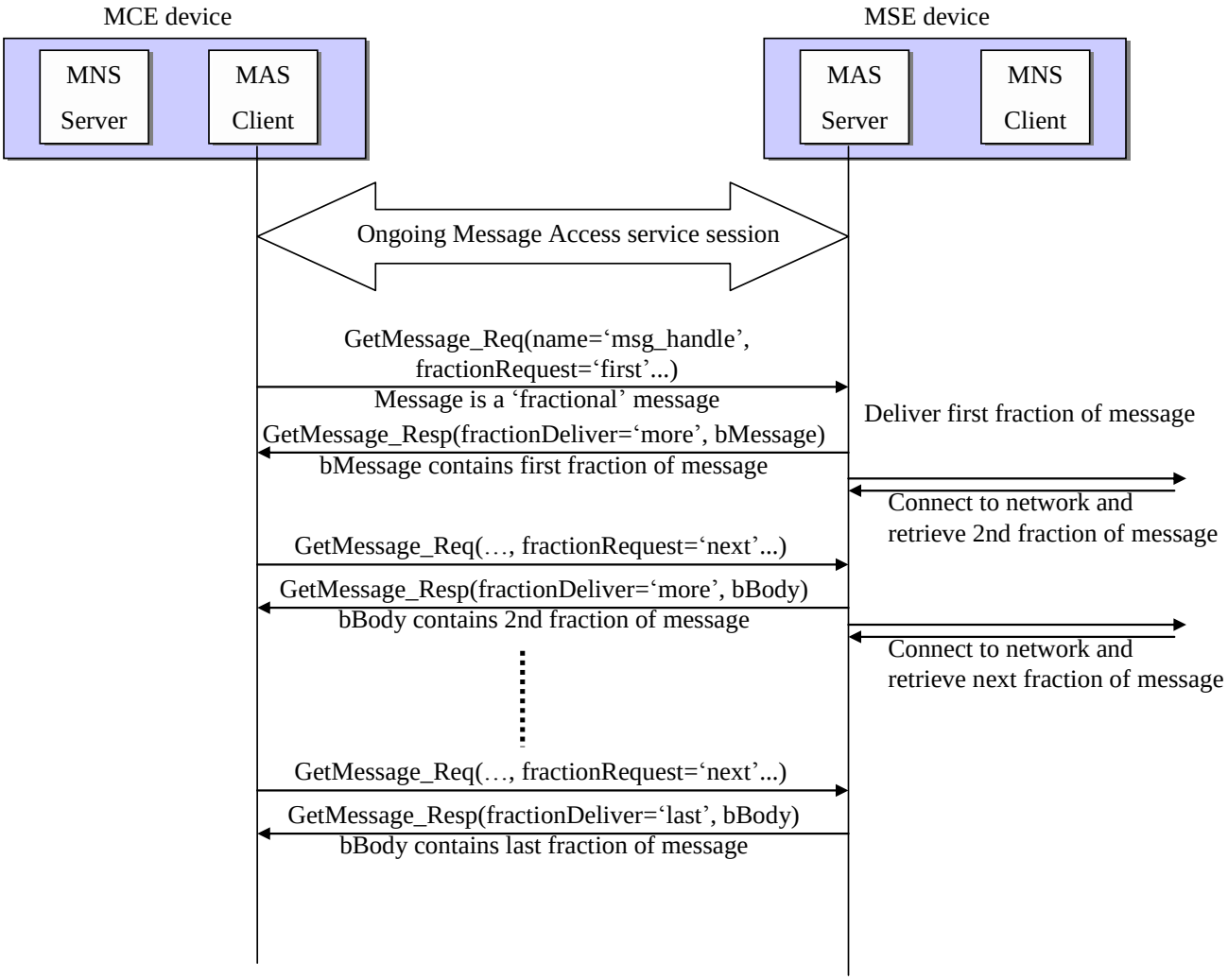

Figure 4.4: Typical sequence for 'GetMessage' applied on a fractional message

4.3 Message Uploading Feature

This feature allows uploading messages from the MCE to the MSE. This feature is used to realize both scenario 3 and 5 (see chapter 2.3). However, the MSE may implement policies that disable the sending of messages to the remote network, thus making scenario 5 impossible to realize.

Function	Support by the MCE	Support by the MSE
SetFolder	M	M
GetFolderListing	M	M
PushMessage	M	M

A function sequence for the Message Uploading feature is illustrated hereunder (see also chapter 5 for definitions of related functions):

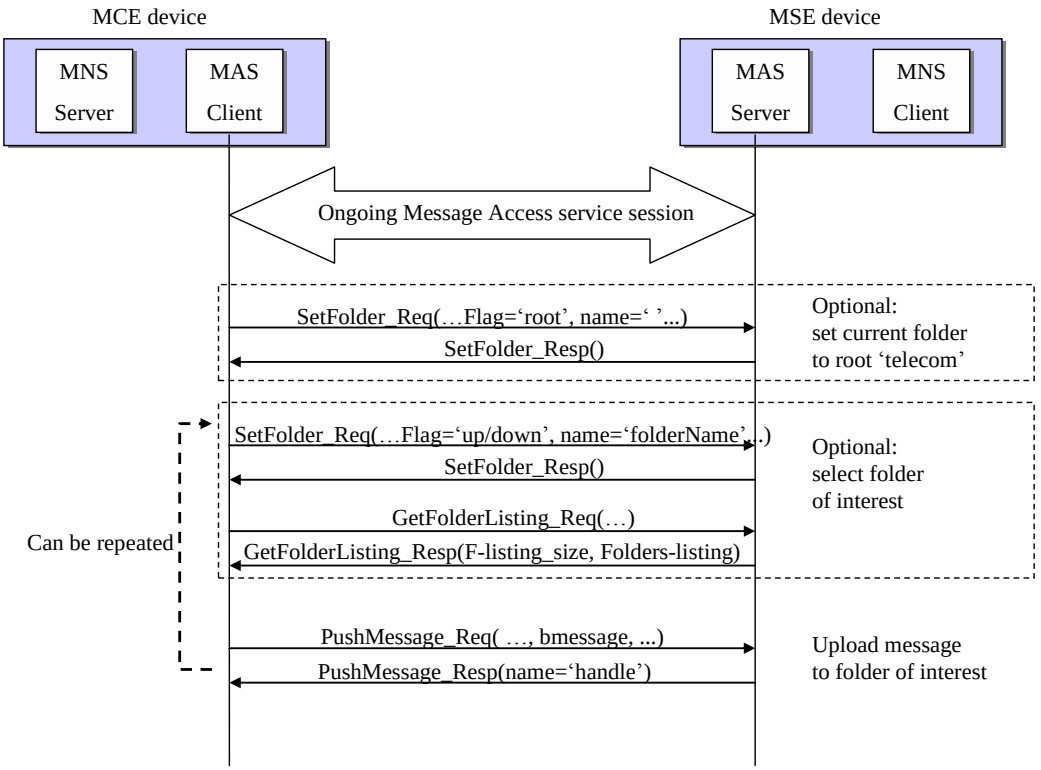

Figure 4.5: Sequence for the Message Uploading feature

4.4 Message Delete Feature

This feature allows the MCE to delete messages on the MSE's message repository.

Function	Support by the MCE	Support by the MSE
SetMessageStatus	M	M

A function sequence for the Message Delete feature is illustrated hereunder (see also chapter 5 for definitions of related functions):

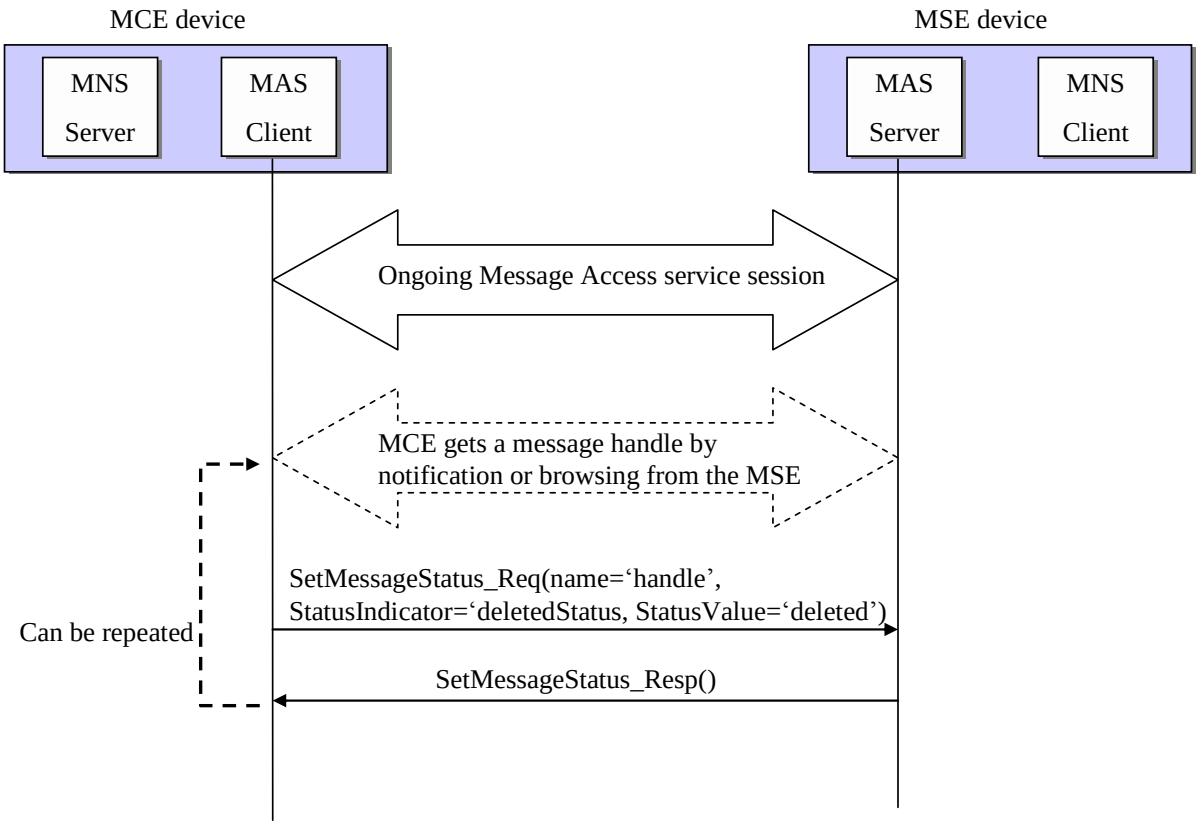

Figure 4.6: Sequence for the Message Delete feature

4.5 Notification Registration Feature

If support for the MAP Notification-Registration feature is claimed by the MSE this feature allows the MCE to set the Notification mode on the MSE side to

- "Off", if no notification required
- "On", if notification required (MNS OBEX connection)

Function	Support by the MCE	Support by the MSE
SetNotificationRegistration	M	M

A function sequence for the Notification-Registration feature is illustrated hereunder (see also chapter 5 for definitions of related functions):

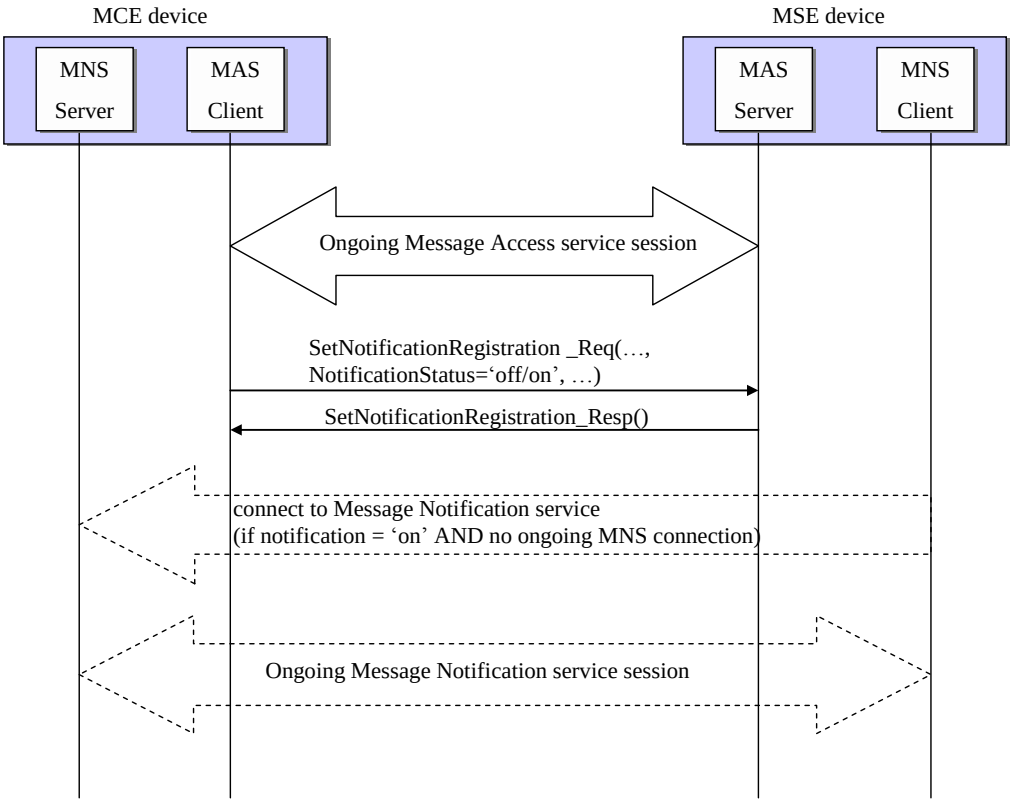

Figure 4.7: Sequence for the Notification Registration feature

The MCE has to perform the NotificationRegistration as described above using a MAS connection for each MAS Instance it requires notifications from. However, the MSE shall initiate Message Notification Service session only if there is no ongoing Message Notification Service session with the MCE as there shall be only one MNS connection for all MAS Instances (see also 3.1.8). The MSE shall terminate Message Notification Service session when MCE sets the SetNotificationRegistration to OFF for all connected MAS instances.

5 Message Access Profile Functions

5.1 SendEvent Function

If the MCE supports the Message Notification feature and the MCE has registered itself at the MSE for Notification, the MSE shall use the SendEvent function to notify the MCE on events affecting the messages listings within the MSE's folders structure, e.g., the arrival of new messages.

In case the MSE has accepted messages for sending to the network from the MCE (using the Message Uploading feature) the MSE shall notify the MCE of the sending status of these messages using the SendEvent function.

The request is formatted as follows:

Fields	Opcode	Put	M
Headers	Connection ID	Connection Identifier	M
	Type	x-bt/MAP-event-report	M
	Application parameters	MASInstanceID	M
	Body/EndOfBody	MAP-Event-Report object	M

The response is formatted as follows:

Fields	Response Code	Success or error code	M

5.1.1 Connection ID

The type header shall be used to indicate the connection ID, received during the connection establishment, in order to signal the recipient of the request which OBEX connection this request belongs to.

5.1.2 Type

The type header shall be used to indicate the type of object to be transmitted:

<x-bt/MAP-event-report> MAP-Event-Report object

5.1.3 Application parameters

For further details, see 6.3.1.

5.1.3.1 MASInstanceID

This header shall be used by the MSE to indicate the corresponding 'MASInstanceID' of the MAS Instance (see 3.1.8 and 7.1). As only one MNS service connection can be established from the MSE device to the MCE this parameter is required by the MCE to determine for which MAS Instance this event is delivered. For further details see 6.3.1.

5.1.4 Body/EndOfBody

These headers shall contain the MAP-event-report object that is sent by the MSE device.

5.2 SetNotificationRegistration Function

If the MCE supports the Message Notification feature, the MCE shall use this function to register itself for being notified of the arrival of new messages.

The request is formatted as follows:

Fields	Opcode	Put	M
Headers	Connection ID	Connection Identifier	M
	Type	x-bt/MAP-NotificationRegistration	M
	Application parameters	NotificationStatus	M
	Body/EndOfBody	Filler-byte 0x30	M

The response is formatted as follows:

Fields	Response Code	Success or error code	M

5.2.1 Connection ID

See chapter 5.1.1.

5.2.2 Type

The type header shall be used to indicate the type of the object to be transmitted:

<x-bt/MAP-NotificationRegistration>

5.2.3 Application Parameters

For further details, see 6.3.1.

5.2.3.1 NotificationStatus

The MCE shall indicate the request for being notified by incoming message events using this parameter. The header shall have either of the values

- "Off", meaning no notification required or
- "On", meaning the notification service (MNS) session shall be established

For further details, see 6.3.1.

5.2.4 Body/EndOfBody

To avoid a PUT with an empty body leading to a 'delete' these headers contain a filler byte. The value of this byte shall be set to 0x30 (="0").

5.3 SetFolder Function

If supported, the MCE shall use this function to navigate the folders of the MSE.

The request is formatted as follows:

Fields	Opcode	SetPath	M
Headers	Connection ID	Connection Identifier	M
	Flags	up/down/root	M
	Name	Name of the folder	O

The response is formatted as follows:

Fields	Response Code	Success or error code	M

5.3.1 Connection ID

See chapter 5.1.1.

5.3.2 Flags and Name

These headers shall be used to indicate the folder to be navigated to. Note that the OBEX SetPath Command is the basis for the SetFolder function where Flags is the 3rd byte of the OBEX request ([10], section 3.3.6). OBEX SetPath allows only for setting the current folder to the root, parent, or a child folder.

For example, in order to set the current folder to "msg", it is necessary to apply SetPath twice: the first is necessary to change into "telecom" and the second is necessary to change into "msg".

The usage of the OBEX Flags and Name header is summarized hereunder:

	Go back to root	Go down 1 level	Go up 1 level
Flags / bit 0:	0	0	1
bit 1 (*):	1	1	1
bit 2~7:	0	0	0
Name header	Mandatory empty	Mandatory Name of child folder	Optional: Name of child folder, omitted, or empty
Description	Reset to the root directory	Go down one level into this directory name, relative to the current directory	When name is not present or empty: go to parent directory (equivalent to “cd ..” on some systems) When name is present: Go up one level before applying name (equivalent to "cd ../name" on some systems)

(*) The creation of new directories on the MSE is not a feature of MAP so the MCE shall set bit1=1.

Accordingly, the name header shall be empty if the directory is shall be set to the parent directory (1 level up) or to the root directory.

5.4 GetFolderListing Function

If supported, the MCE shall use this function to retrieve the Folder-Listing object from the current folder of the MSE.

The request is formatted as follows:

Fields	Opcode	Get	M
Headers	Connection ID	Connection Identifier	M
	Type	x-obex/folder-listing	M
	Application parameters	MaxListCount	O
		ListStartOffset	O

The response is formatted as follows:

Fields	Response Code	Success or error code	M
Headers	Application Parameters	FolderListingSize	C
	Body/EndOfBody	Folder-Listing object	C

C: these parameters shall be present if the request has been successful

5.4.1 Connection ID

See chapter 5.1.1.

5.4.2 Type

The type header shall be used to indicate the type of object to be received.

<x-obex/folder-listing> Folder-Listing object.

5.4.3 Application Parameters

For further details, see 6.3.1.

5.4.3.1 MaxListCount

This header may be used to indicate the maximum number of folders listed in the **<x-obex/folder-listing >** object. The maximum number of entries shall be 1,024 if this header is not specified.

ListStartOffset:

This header may be used to indicate the offset of the first entry of the returned Folder-Listing object compared to the first entry of the Folder-Listing object that would be returned if the ListStartOffset parameter was not specified in the request. For example, if ListStartOffset is 5 then the first 5 listing entries are not delivered. The offset shall be 0 if this header is not specified.

FolderListingSize

This application parameter shall be used in the response if the value of MaxListCount in the request is 0. In this case the parameter shall report the actual number of accessible folders in the current folder of the MSE. If MaxListCount = 0, the MSE shall ignore all other application parameters that may be present in the request. The response shall not contain any Body header.

5.4.4 Body/EndOfBody

If the request gas been successful, these headers shall contain the **<x-obex/folder-listing>** object that is sent by the MSE device.

5.5 GetMessagesListing Function

If supported, the MCE shall use this function to retrieve Messages-Listing objects from the MSE.

The request is formatted as follows:

Fields	Opcode	Get	M
Headers	Connection ID	Connection Identifier	M
	Type	x-bt/MAP-msg-listing	M
	Name	Name of the Folder	M
	Application parameters	MaxListCount	O
		ListStartOffset	O
		SubjectLength	O
		ParameterMask	O
		FilterMessageType	O
		FilterPeriodBegin	O
		FilterPeriodEnd	O
		FilterReadStatus	O
		FilterRecipient	O
		FilterOriginator	O
		FilterPriority	O

The response is formatted as follows:

Fields	Response Code	Success or error code	M
Headers	Application parameters	NewMessage	C
		MSETime	C
		MessagesListingSize	C
	Body/EndOfBody	Messages-Listing object	C

C: These parameters shall be present if the request has been successful.

5.5.1 Connection ID

See chapter 5.1.1.

5.5.2 Name

This property shall be used to indicate the folder from which the Messages-Listing object is to be retrieved. The property shall be empty in case the desired listing is that of the current folder or shall be the name of a child folder. Thus, the value shall not include any path information.

5.5.3 Type

The type header shall be used to indicate the type of object to be retrieved:

<x-bt/MAP-msg-listing> Messages-Listing object

5.5.4 Application Parameters

For further details, see 6.3.1.

5.5.4.1 MaxListCount

This header may be used to indicate the maximum number of messages listed in the Messages-Listing object. The number of messages in the list shall not exceed the limit specified by this parameter. The maximum number of entries shall be 1,024 if this header is not specified. The MessageListing object shall contain MaxListCount entries if enough entries exist, starting from ListStartOffset. For example, a request with 'ListStartOffset' = 0 and 'MaxListCount'=100 delivers the 100 most recent messages in chronological descending order. If 'MaxListCount'=0 in the request, the MSE shall respond with the headers "NewMessage, MSETime", and "MessagesListingSize" only (see description of these headers below) and shall not deliver a Messages-Listing object.

ListStartOffset:

This header may be used to indicate the offset of the first entry of the returned MessagesListing object. For example, if ListStartOffset is 5, then the first 5 messages are not delivered. The offset shall be 0 if this header is not specified.

SubjectLength:

This header may be used to indicate the maximum string-length of the "subject" parameter in the entries of the Messages-Listing object. The value range shall be 1…255.

ParameterMask:

This header may be used to indicate the parameters contained in the requested Messages-Listing objects. The MCE can use this header to receive only the relevant content of the requested Messages-Listing objects.

If this header is not specified or carries the value 0 the MSE shall return all parameters of the Messages-Listing object DTD (chapter 3.1.6) labeled as "REQUIRED" and may return all other attributes. The message handle shall always be present.

bit 0	subject
bit 1	datetime
bit 2	sender_name
bit 3	sender_addressing
bit 4	recipient_name
bit 5	recipient_addressing
bit 6	type
bit 7	size
bit 8	reception_status
bit 9	text
bit 10	attachment_size
bit 11	priority
bit 12	read
bit 13	sent
bit 14	protected
bit 15	replyto_addressing
bit 16-31	Reserved for future use

Bit i=1 indicates that the parameter related to Bit i shall be present in the requested Messages-Listing. The reserved bits shall be set to 0 by MCE and discarded by MSE.

Annotation Filtering (attributes below)**:** The filter operations related to the filter parameters listed below shall be used as AND filtering, so the messages returned in the MessagesListing object have to fulfill all filter conditions defined in the request.
If list-segmentation with MaxListCount/ListStartOffset is requested in combination with filter-operations, then the filter-operations shall be applied first. The segmentation shall subsequently be applied onto the filtering result, which shall be a message-list, sorted chronologically in descending order.

FilterMessageType:

This application parameter may be used to filter by message type the messages that are returned in the Messages-Listing object.

FilterPeriodBegin, FilterPeriodEnd:

These application parameters may be used to filter by delivery date the messages that are returned in the Messages-Listing object.

The two parameters shall be formatted in "YYYYMMDDTHHMMSS" as defined for the OBEX "time" header in 2.2 of [10] (Local Time time basis). If "FilterPeriodBegin" is not specified the returned message listing shall include the messages older than "FilterPeriodEnd". If "FilterPeriodEnd" is not specified the returned message listing shall include the messages from "FilterPeriodBegin" to current time.

If both parameters are not specified no messages shall be filtered out by this parameters.

If the value of "FilterPeriodBegin" is larger than the value of "FilterPeriodEnd" no messages shall be delivered.

FilterReadStatus:

This application parameter may be used to filter by read-status the messages that are returned in the MessagesListing object. The parameter shall have one of the values "read", "unread" or "no-filtering".

FilterRecipient:

This application parameter may be used to filter by message-recipient the messages that are returned in the Messages-Listing object. A related bMessage shall be included in the listing if the delivered FilterRecipient string matches a sub-string of one of the vCard attributes "N", "TEL" and "EMAIL" of at least one <bmessage-recipient> contained by this message. This parameter shall not be used if the MCE intends to select messages with any recipient. The text string in this application parameter shall be coded in UTF-8.

FilterOriginator:

This application parameter may be used to filter by message-originator the messages that are returned in the Messages-Listing object. A related bMessage shall be included in the listing if the delivered FilterOriginator string matches a sub-string of one of the vCard attributes "N", "TEL" and "EMAIL" of the <bmessage-originator> contained by this message. This parameter shall not be used if the MCE intends to select messages with any originator. The text string in this application parameter shall be coded in UTF-8.

FilterPriority:

This application parameter may be used to filter by priority the messages that are returned in the Messages-Listing object. The parameter shall have one of the values "high", "non-high" and "no-filtering".

NewMessage:

If the request has been successful the MSE shall use this parameter to indicate the presence of unread messages in the listing using this parameter. The parameter shall have one of the values "Off" (meaning no unread messages) and "On" (meaning unread messages are present). For further details, see 6.3.1.

MSETime:

If the request has been successful, this application parameter shall be used in the response to report the Local Time basis of the MSE and its UTC offset, so the MCE is supported in interpreting the timestamps of the messages listing entries. The format shall be "YYYYMMDDTHHMMSS±hhmm". E.g., for Central European Time (CET) the offset is UTC+1h so the delivered value is "YYYYMMDDTHHMMSS+0100" where "YYYYMMDDTHHMMSS" is the current CET timestamp of the MSE. If the MSE device

has no information about the current UTC time, the offset value shall not be delivered so the delivered parameter is "YYYYMMDDTHHMMSS". For further details, see 6.3.1

MessagesListingSize:

If the request has been successful, this application parameter shall be used in the response to report the number of accessible messages in the corresponding folder fulfilling – if present – the filter parameters as described above. If MaxListCount = 0, the MSE shall ignore the request-parameters "ListStartOffset", "SubjectLength" and "ParameterMask". In this case the response shall not contain the Body header with the <x-bt/msg-listing> object.

5.5.5 Body/EndOfBody

If the request has been successful, these headers shall contain the **<x-bt/msg-listing>** object that is sent by the MSE device. If MaxListCount = 0 in the request this parameter shall not be present in the response.

5.6 GetMessage Function

This function retrieves a specific message from the MSE device. The request is formatted as follows:

Fields	Opcode	Get	M
Headers	Connection ID	Connection Identifier	M
	Name	Message Handle	M
	Type	x-bt/message	M
	Application Parameters	Attachment	M
		Charset	M
		FractionRequest	O

The response is formatted as follows:

Fields	Response Code	Success or error code	M
Headers	Application parameters	FractionDeliver	C1
	Body/EndOfBody	bMessage object or bBody object	C2

C1: this parameter shall be present if the request has been successful and the 'Fraction Request' parameter has been present in the request

C2: this parameter shall be present if the request has been successful

5.6.1 Connection ID

See chapter 5.1.1.

5.6.2 Name

The Name header shall be used to indicate the handle of the message to be retrieved. The handle shall be represented by a null-terminated Unicode text string with 16 hexadecimal digits.

5.6.3 Type

The type header shall be used to indicate the type of object to be retrieved. The value shall be set to one of the following settings:

<x-bt/message> bMessage object

5.6.4 Application parameters

For further details, see 6.3.1.

Charset:

This application parameter shall be used to determine the trans-coding of the textual parts of the delivered bMessage-content as described in chapter 3.1.3. The value shall be either

<native> if the message object shall be delivered without trans-coding or

<UTF-8> if the message text shall be trans-coded to UTF-8 by the before delivery

As described in 3.1.3 the following requirements shall be fulfilled:

- If the message is an email or a MMS the only value shall be <UTF-8> as textual parts of these message types shall be always trans-coded; The MSE shall reject a request with value <native> in this case.
- If the message is a SMS both values are possible:
  - **<native>** initiates the delivery of the overall SMS PDU, embedded in the bMessage object, without any trans-coding.
  - **<UTF-8>** initiates the delivery of the textual part of the SMS only, trans-coded to UTF-8. The MSE shall reject a request with value <UTF-8> if the message doesn't include textual content

Attachment:

This application parameter shall be used if the attachment(s) (if any) of a message are to be included in the bMessage object returned by the MSE. The value of this parameter is "Off" (no attachments) and "On" (attachments to be delivered).

FractionRequest:

This application parameter may be used if the message is a fractioned email as applied for some push-email services (see also 3.1.6 "reception_status") and shall not be used otherwise. In this case, several requests are required to retrieve the overall bMessage object. The value shall be either:

<first> when the first email fraction of the message object shall be delivered; this value shall be used by the MCE in its first GetMessage request on a particular bMessage or

<next> when the email fraction following the fraction of the previous GetMessage request shall be delivered by the MSE

The response of the MSE on a request with value <first> shall deliver a bMessage with all parameters as defined in 3.1.3 with exactly one bBody object containing the first fraction of the email.

The response of the MSE on a request with value <next> shall deliver only a bBody object as defined in 3.1.3 with containing the next fraction of the email.

If the message affected is a fractioned email but this application parameter is not present in the request the MSE shall load the overall message from the network and deliver the message completely.

FractionDeliver:

This application parameter shall be present if the 'Fraction Request' parameter has been present in the related GetMessage request and shall not be used otherwise. The value shall be either

<more> when an email fraction is delivered and there are other fractions following this one.

<last> when the last email fraction of the message object is delivered

5.6.5 Body/EndOfBody

If the request has been successful, these headers shall contain the body of the bMessage that is returned by the MSE device (see x-bt/message definition 3.1.3). The message may be in its original form, or it may have been stripped of all its attachments, depending on the value of the application parameter "Attachment" in the request.

In case of a request on a fractioned email with 'Fraction Request' value <next> (see above) this header shall return a bBody object only (see x-bt/message definition 3.1.3).

5.7 SetMessageStatus Function

The function allows the MCE to modify the status of a message on the MSE.

The request is formatted as follows:

Fields	Opcode	Put	M
Headers	Connection ID	Connection Identifier	M
	Name	Message Handle	M
	Type	x-bt/messageStatus	M
	Application Parameters	StatusIndicator	M
		StatusValue	M
	Body/EndOfBody	Filler-byte 0x30	M

The response is formatted as follows:

Fields	Response Code	Success or error code	M

5.7.1 Connection ID

See chapter 5.1.1.

5.7.2 Name

The Name header shall contain the handle of the message the status of which shall be modified. The handle shall be represented by a null-terminated Unicode text string with 16 hexadecimal digits.

5.7.3 Type

The type header shall be used to indicate the type of object to be sent.

<x-bt/messageStatus>

5.7.4 Application Parameters

StatusIndicator:

This header shall be used to indicate which status information is to be modified. The parameter value shall be either

- "readStatus" for the Read status or
- "deletedStatus" for the Deleted status

StatusValue:

This header shall be used to indicate the new value of the status indicator to be modified. The parameter shall have one of the values:

- "yes" (=read) or "no" (=unread) for the "readStatus" indicator
- "yes" (=deleted) or "no" (=undeleted) for the "deletedStatus" indicator

Changing the status of a message can initiate actions on the MSE. The following rules shall be applied after the modification:

- messages in any MSE folder whose deletedStatus have been set from "no" (non-deleted) to "yes" (deleted) shall be shifted by the MSE to the "Deleted" folder

messages in the 'Deleted' folder whose deletedStatus have been set from "yes" (deleted) to "no" (un-deleted) shall be shifted by the MSE to the "Inbox" folder

5.7.5 Body/EndOfBody

To avoid PUT with empty Body leading to a 'delete' of the related message these headers shall contain a filler byte. The value of this byte shall be set to 0x30 (="0").

5.8 PushMessage Function

This function allows an MCE to push a message to a folder of the MSE.

The request is formatted as follows:

Fields	Opcode	Put	M
Headers	Connection ID	Connection Identifier	M
	Type	x-bt/message	M
	Name	Name of the Folder	M
	Application parameters	Transparent	O
		Retry	O
		Charset	M
	Body/EndOfBody	bMessage	M

The response is formatted as follows:

Fields	Response Code	Success or error code	M
Headers	Name	Handle	C

C: This parameter shall be present if the request has been successful

5.8.1 Connection ID

See chapter 5.1.1.

5.8.2 Name

In a request:

This property shall be used to indicate the folder to which the Message object is to be pushed. The property shall be empty in case the desired listing is that of the current folder or shall be the name of a child folder. Thus, the value shall not include any path information.

In a response:

The Name header shall be used to contain the handle that was assigned by the MSE device to the message that was pushed by the MCE device. The handle shall be represented by a null-terminated Unicode text string with 16 hexadecimal digits.

5.8.3 Type

The type header shall be used to indicate the type of the pushed object.

<x-bt/message> bMessage object

5.8.4 Application parameters

For further details, see 6.3.1.

Transparent

This parameter may be used to indicate to the MSE that no copy of the message shall be kept in the 'Sent' folder after the message was sent. This is especially useful for telematics applications, e.g., frequent sending of car's speed and position for traffic measurements (floating car data). This application parameter is optional. The value of this parameter is "OFF" (keep messages in 'Sent' folder) and "ON" (don't keep messages in Sent' folder). If this parameter is not defined in the request the MSE shall use "OFF" as default value.

Retry

This parameter may be used to indicate whether successive attempts at sending the message shall be carried out in case the cellular network is not accessible when the message is sent from the MCE to the outbox of the MSE. This application parameter is optional. The value of this parameter is "OFF" (don't retry) and "ON" (retry). If this parameter is not defined in the request the MSE shall use "ON" as default value.

Charset:

This application parameter shall be used to determine the trans-coding of the textual parts of the delivered bMessage-content as described in chapter 3.1.3. The value shall be set to either

<native> if the message object shall be delivered without trans-coding (e.g., an SMS-Submit PDU), or

<UTF-8> if the message text shall be trans-coded to UTF-8 by the MSE before delivery

As described in section 3.1.3 the following requirements shall be fulfilled:

- If the message is an email or a MMS the only value shall be <UTF-8> as textual parts of these message types shall be always trans-coded; The MSE shall reject a request with value <native> in this case.
- If the message is an SMS both values are possible:
  - **<native>** indicates the overall SMS PDU is embedded in the bMessage object, without any trans-coding.
  - **<UTF-8>** indicates the bMessage includes the textual part of the SMS only, trans-coded to UTF-8. The MSE shall reject a request with value <UTF-8> if the message doesn't include textual content.

5.8.5 Body/EndOfBody

These headers shall contain the body of the bMessage that is delivered to the MSE device. If the textual parts of the message are coded in UTF-8 the MSE shall trans-code it to the required format as described in chapter 3.1.3.

5.9 UpdateInbox Function

This function allows the MCE to initiate an update of the MSE's inbox, i.e. the MSE shall contact the network (e.g., its mailbox) to retrieve new messages if available.

The request is formatted as follows:

Fields	Opcode	Put	M
Headers	Connection ID	Connection Identifier	M
	Type	x-bt/MAP-messageUpdate	M
	Body/EndOfBody	Filler-byte "0x30"	M

The response is formatted as follows:

Fields	Response Code	Success or error code	M

If the MSE does not allow the polling of its mailbox it shall answer with a 'Not implemented' error response (see 6.3.3).

5.9.1 Connection ID

See chapter 5.1.1.

5.9.2 Type

The type header shall be used to indicate the type of object to be sent.

<x-bt/MAP-messageUpdate> virtual object containing a filler byte 0x30 (="0")

5.9.3 Body/EndOfBody

To avoid PUT with empty Body leading to a 'delete' of the related message these headers shall contain a filler byte. The value of this byte shall be set to 0x30.

6 OBEX Services

6.1 OBEX Services Definition

This profile is based on GOEP [3] and makes use of the IrOBEX specification [10]. Within the scope of MAP, the following OBEX services are defined: The Message Access service (MAS) and the Message Notification Service (MNS).

The Message Access service is an OBEX service by which the MCE acts as an OBEX Client and connects to an MSE that acts as OBEX Server.

The Message Notification service is an OBEX service by which the MSE acts as OBEX Client and connects to the MCE that acts as an OBEX Server.

The MAP features are using the above defined OBEX services in the following way:

Feature	OBEX Service
Message Notification	Message Notification Service
Message Browsing	Message Access Service
Message Uploading	Message Access Service
Message Delete	Message Access Service
Notification Registration	Message Access Service

See details in chapter 7.

6.2 OBEX Operations Used

The following tables list the OBEX operations required by the Message Access Profile.

6.2.1 Message Access Service:

OBEX operations supported in a Message Access Service session:

OBEX operation	Ability to send (Client)	Ability to respond (Server)
	MCE	MSE
Connect	M	M
Disconnect	M	M
Put	O	M
Get	M	M
Abort	M	M
SetPath	M	M

Table 6.1: OBEX Operations- MAS

6.2.2 Message Notification Service:

OBEX operations supported in the Message Notification Service session:

OBEX operation	Ability to send (Client)	Ability to respond (Server)
	MSE	MCE
Connect	M	M
Disconnect	M	M
Put	M	M
Abort	M	M

Table 6.2: OBEX Operations- MNS

6.3 OBEX Headers

Table 6.3 lists the OBEX headers required by the Message Access Profile. Any other headers listed in GOEP should not be used in this profile.

OBEX header	MCE	MSE
Name	M	M
Type	M	M
Body	M	M
End of Body	M	M
Target	M	M
Who	M	M
Connection ID	M	M
Application Parameters	M	M

Table 6.3: OBEX Headers

Note that the profile does not exclude the headers that are not listed in Table 6-3. Some implementations may choose to support additional headers that enable added value services. Therefore unknown, or unsupported, headers shall always be skipped and ignored.

The following OBEX Target header UUID values shall be used for the OBEX services MAS and MNS:

OBEX service	Target header UUID value
MAS	bb582b40-420c-11db-b0de-0800200c9a66
MNS	bb582b41-420c-11db-b0de-0800200c9a66

Table 6.4: OBEX Target header UUID values

6.3.1 Application Parameters Header

The tag IDs used in the Application Parameters header are listed in Table 6.5.

Value	Tag ID	Length	Possible Values

Value	Tag ID	Length	Possible Values
MaxListCount	0x01	2 bytes	0x0000 to 0xFFFF
StartOffset	0x02	2 byte	0x0000 to 0xFFFF
FilterMessageType	0x03	1 byte	Bit mask: 0000xxx1 = "SMS_GSM" 0000xx1x = "SMS_CDMA" 0000x1xx = "EMAIL" 00001xxx = "MMS" all other values: rfu Where 0 = "no filtering, get this type" 1 = "filter out this type"
FilterPeriodBegin	0x04	variable	string with Begin of filter period See chapter 5.5.4
EndFilterPeriodEnd	0x05	variable	string with End of filter period See chapter 5.5.4
FilterReadStatus	0x06	1 byte	Bit mask: 00000001 = get unread messages only 00000010 = get read messages only 00000000 = no-filtering, get both read and unread messages; all other values: undefined
FilterRecipient	0x07	variable	Text (UTF-8) wildcards "*" may be used if required
FilterOriginator	0x08	variable	Text (UTF-8), wildcards "*" may be used if required
FilterPriority	0x09	1 byte	Bit mask: 00000000 = no-filtering 00000001 = get high priority messages only 00000010 = get non-high priority messages only; all other values: undefined
Attachment	0x0A	1 byte	Bit mask (*): xxxxxxx1 = "ON" xxxxxxx0 = "OFF"
Transparent	0x0B	1 byte	Bit mask (*): xxxxxxx1 = "ON" xxxxxxx0 = "OFF"

Value	Tag ID	Length	Possible Values
Retry	0x0C	1 byte	Bit mask (*): xxxxxxx1 = "ON" xxxxxxx0 = "OFF"
NewMessage	0x0D	1 byte	Bit mask (*): xxxxxxx1 = "ON" xxxxxxx0 = "OFF"
NotificationStatus	0x0E	1 byte	Bit mask (*): xxxxxxx0 = "Off" xxxxxxx1 = "On"
MASInstanceID	0x0F	1 byte	0…255
ParameterMask	0x10	4 bytes	Bit mask, settings see 5.5.4
FolderListingSize	0x11	2 bytes	0x0000 to 0xFFFF
MessagesListingSize	0x12	2 bytes	0x0000 to 0xFFFF
SubjectLength	0x13	1 byte	1…255
Charset	0x14	1 byte	Bit mask (*): xxxxxxx0 = "native" xxxxxxx1 = "UTF-8"
FractionRequest	0x15	1 byte	Bit mask: xxxxxxx0 = "first" xxxxxxx1 = "next"
FractionDeliver	0x16	1 byte	Bit mask (*): xxxxxxx0 = "more" xxxxxxx1 = "last"
StatusIndicator	0x17	1 byte	Bit mask (*): xxxxxxx0 = "readStatus" xxxxxxx1 = "deletedStatus"
StatusValue	0x18	1 byte	Bit mask (*): xxxxxxx1 = "yes" xxxxxxx0 = "no"
MSETime	0x19	variable	string with current time basis and UTC-offset of the MSE See chapter 5.5.4

Table 6.5: Tag IDs

(*) the bits marked by 'x" shall be set to zero; all other values are reserved for future use

All of the Application Parameter header values use big-endian byte ordering.

6.3.2 OBEX Headers in Multi-Packet Responses

In the case of multi-packet responses, there is a need to specify which packet contains the headers to be returned to the MCE or to the MSE. Although the IrOBEX

specification does not impose any restrictions in this area, the following rule shall be used in the Message Access Profile to encourage interoperability:

In the case of a multi-packet PUT response (i.e. the object being transported is large enough to require several packets), the headers shall be placed in the last packet or respectively in the last packets if one packet is not sufficient for the headers. All intermediate response packets shall contain only the Continue response code or (when necessary) one of the error codes. Note that if the OBEX Partial Content response code is used, it must be issued in the last response packet, after the object has been completely received.

In the case of multi-packet Get response (i.e. the object being transported is large enough to require several packets), all the headers other than the body header shall be placed in the first packet. If the first packet has enough room to include a portion of the object body, then the first packet shall end with the body header that carries this portion of object. Otherwise, the object shall be transferred in subsequent packets.

6.3.3 OBEX Error Codes

Message Access Service

Table 6.6 summarizes the error codes for the Message Access Service to be provided by the MSE and to be recognized by the MCE:

Error Code	OBEX Client (MCE) (interprets the Error Codes)	OBEX Server (MSE) (informs of Errors)	Meaning in the Message Access Profile
Bad Request	M*	M	Function not recognized or ill-formatted
Not implemented	M*	M	Function recognized but not supported
Unauthorized	M*	O	In operations with actual exchange of an object in the body header (either in the request or the response), indicates that the function was recognized and well formatted, but that the object to be handled is protected and access is not authorized (either temporarily or permanently).
Precondition Failed	M*	M	The function was recognized and well-formatted but there is a problem with one of the request's parameter values
Not Found	M*	M	The function was recognized and well-formatted and all the parameters are proper, but the bMessage handle or the message folder could not be found

Error Code	OBEX Client (MCE) (interprets the Error Codes)	OBEX Server (MSE) (informs of Errors)	Meaning in the Message Access Profile
Not Acceptable	M*	O	The request is recognized and well formatted and all the parameter values are legal, but there is a problem with a parameter value that indicates a request that cannot be met by the Server.
Service Unavailable	M*	M	The function was recognized and well formatted and is normally executable, but a system condition prevents it from being performed. It could be for instance that the message folders cannot be accessed when the MSE is involved in a call. Another example would be when the MCE sent a message to the Outbox of the MSE, but the cellular service is currently not available.
Forbidden	M*	O	Function recognized and correctly formatted but temporarily barred

Table 6.6: ERROR Codes

* Indicates that the Client shall recognize this response code as an error code.

On the MCE side, the entire response code list above must be recognized as error codes; how to handle these error codes is left to the implementer's discretion.

Message Notification Service

The table below summarizes the error codes for the Message Notification Service to be provided by the MSE and to be recognized by the MCE:

Error Code	OBEX Client (MSE) (interprets the Error Codes)	OBEX Server (MCE) (informs of Errors)	Meaning in the Message Notification Service
Bad Request	M*	M	Function not recognized or ill-formatted
Not implemented	M*	M	Function recognized but not supported
Unauthorized	M*	O	In operations with actual exchange of an object in the body header (either in the request or the response), indicates that the function was recognized and well formatted, but that the object to be handled is protected and access is not authorized (either temporarily or permanently).
Precondition Failed	M*	M	The function was recognized and well-formatted but there is a problem with one of the request's parameter values

Error Code	OBEX Client (MSE) (interprets the Error Codes)	OBEX Server (MCE) (informs of Errors)	Meaning in the Message Notification Service
Not Acceptable	M*	O	The request is recognized and well formatted and all the parameter values are legal, but there is a problem with a parameter value that indicates a request that cannot be met by the Server.
Service Unavailable	M*	O	The function was recognized and well formatted and is normally executable, but a system condition prevents it from being performed. It could be for instance that the event report cannot be delivered due to resource problems of the MCE.
Forbidden	M*	O	Function recognized and correctly formatted but temporarily barred

Table 6.7: ERROR Codes MAS

* Indicates that the Client shall recognize this response code as an error code.

On the MSE side, the entire response code list above shall be recognized as error codes; how to handle these error codes is left to the implementer's discretion.

Support for response codes indicated Table 6.6 and Table 6.7 as optional is recommended, because they are more informative and provide the MCE with a better indication of the nature of an error; this permits better error reporting. The "x complements y" relationship between response codes is illustrated below:

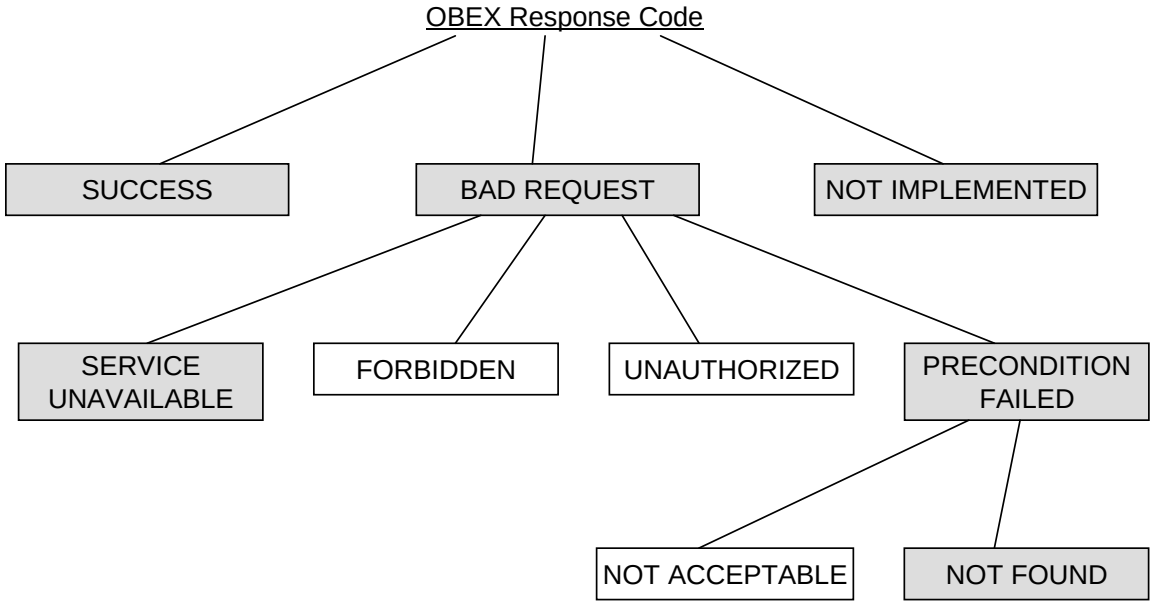

Figure 6.1: OBEX Response Codes

When multi-packet responses are used, response codes must be returned as early as possible, preferably in the first response packet. In some cases – for example, Service

Unavailable – it is possible that an error condition won't arise until the operation is underway, in which case it is acceptable to return a response code in a packet other than the first one.

6.4 Initializing a MAP session

OBEX sessions in MAP shall always be initiated by the MCE device.

However, the session initialization sequence to be used depends on the features that are to be used during the course of the MAP session:

The initialization sequence that applies to applications that use only the MSE's Message Access Service is described in 6.4.1.

For applications that use both the Message Access Service of the MSE and the Message Notification Service of the MCE, the initialization sequence that applies is described in 6.4.2.

For applications that use only the Message Notification Service of the MCE, the initialization sequence is that described in 6.4.3.

6.4.1 Initialization sequence for a MAP session that uses only the Message Access service

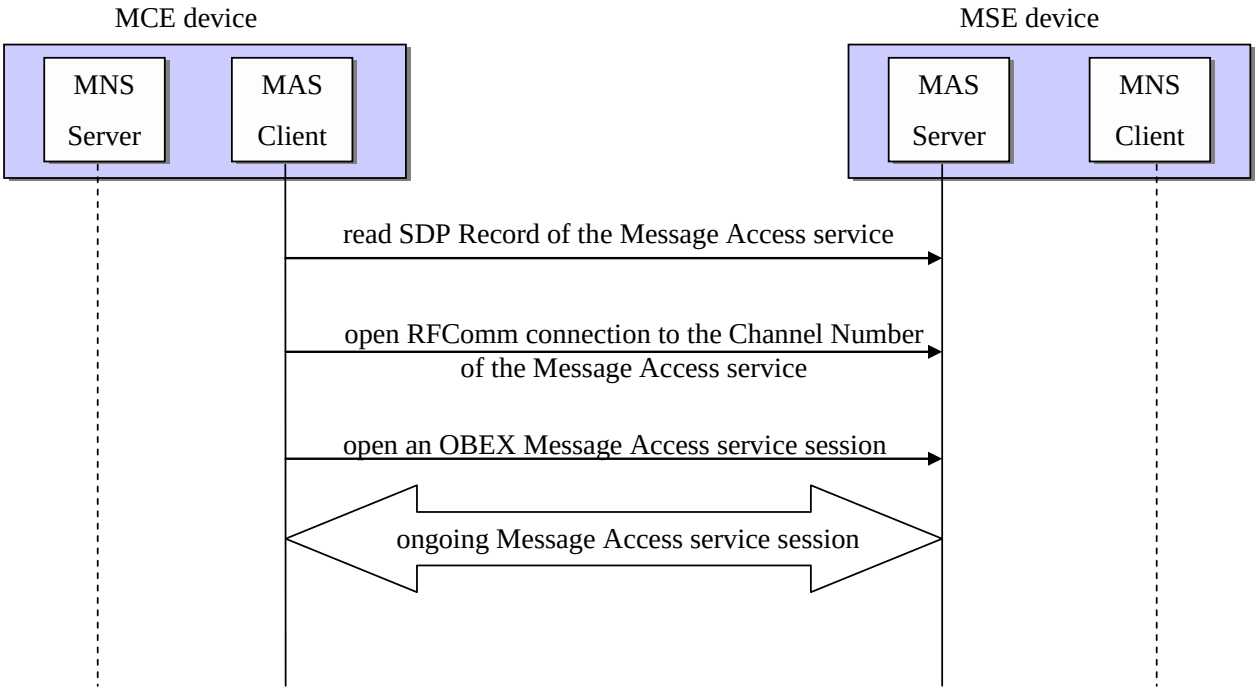

Figure 6.2: Establishment of MAP session (usage of Message Access Service only)

The establishment of a Message Access service session is done in accordance with [3] with the MCE as Client and the MSE as Server.

For the UUID value to be used for the Target header see 6.3.

6.4.2 Initialization sequence for a MAP session that uses both the Message Access service and the Message Notification service

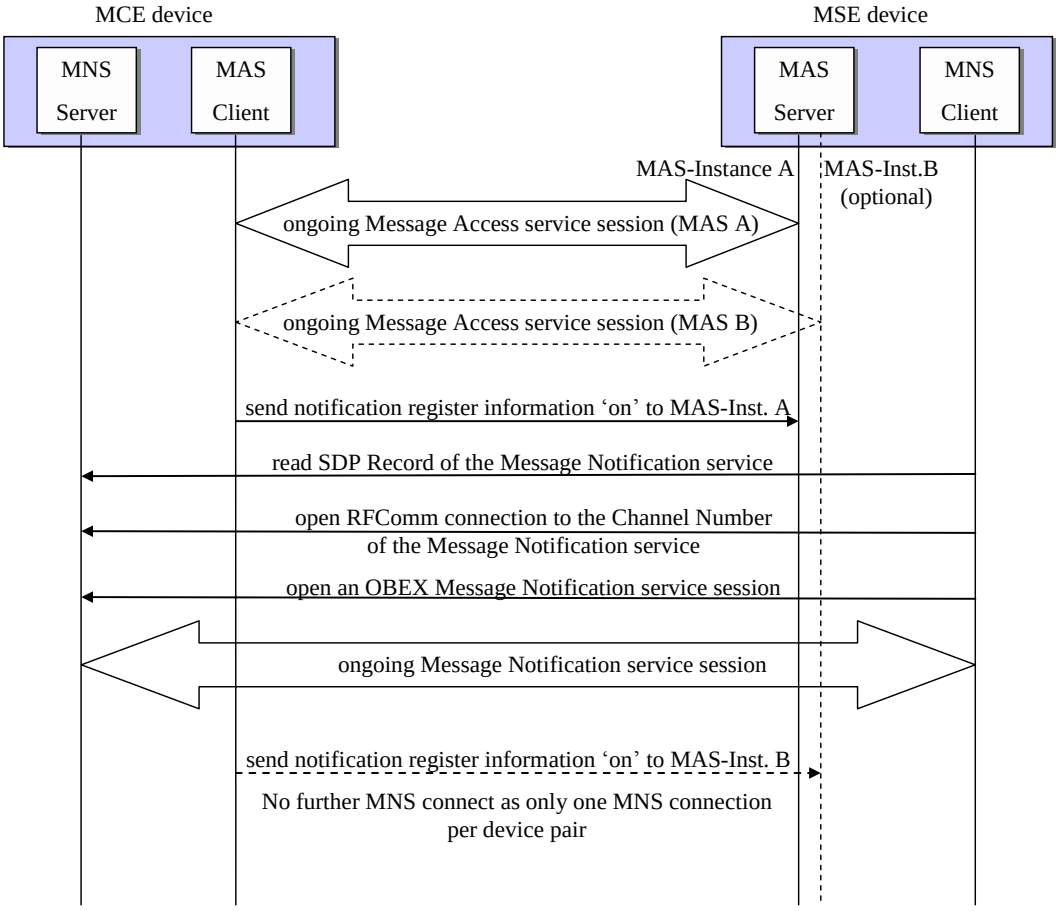

Figure 6.3: Establishment of MAP session (usage of Message Access Service and Message Notification Service)

The establishment of a Message Notification service session is done in accordance with [10] with the MCE as OBEX Server and the MSE as OBEX Client. For the UUID values to be used for the Target header see 6.3. The establishment of a Message Notification session requires the previous establishment of a Message Access service session as described in chapter 6.4.1.

The MCE device may establish one MAS connection for each MAS Instance provided by the MSE device while the MSE shall establish only one MNS connection (see also 4.5). The MNS connection shall be established by the first SetNotificationRegistration set to ON during a MAP session. Respectively, when the MNS connection is established all following SetNotificationRegistration set to ON by any MAS-Instance will cause no further MNS connection.

6.4.3 Initialization sequence for a MAP session that uses only the Message Notification service

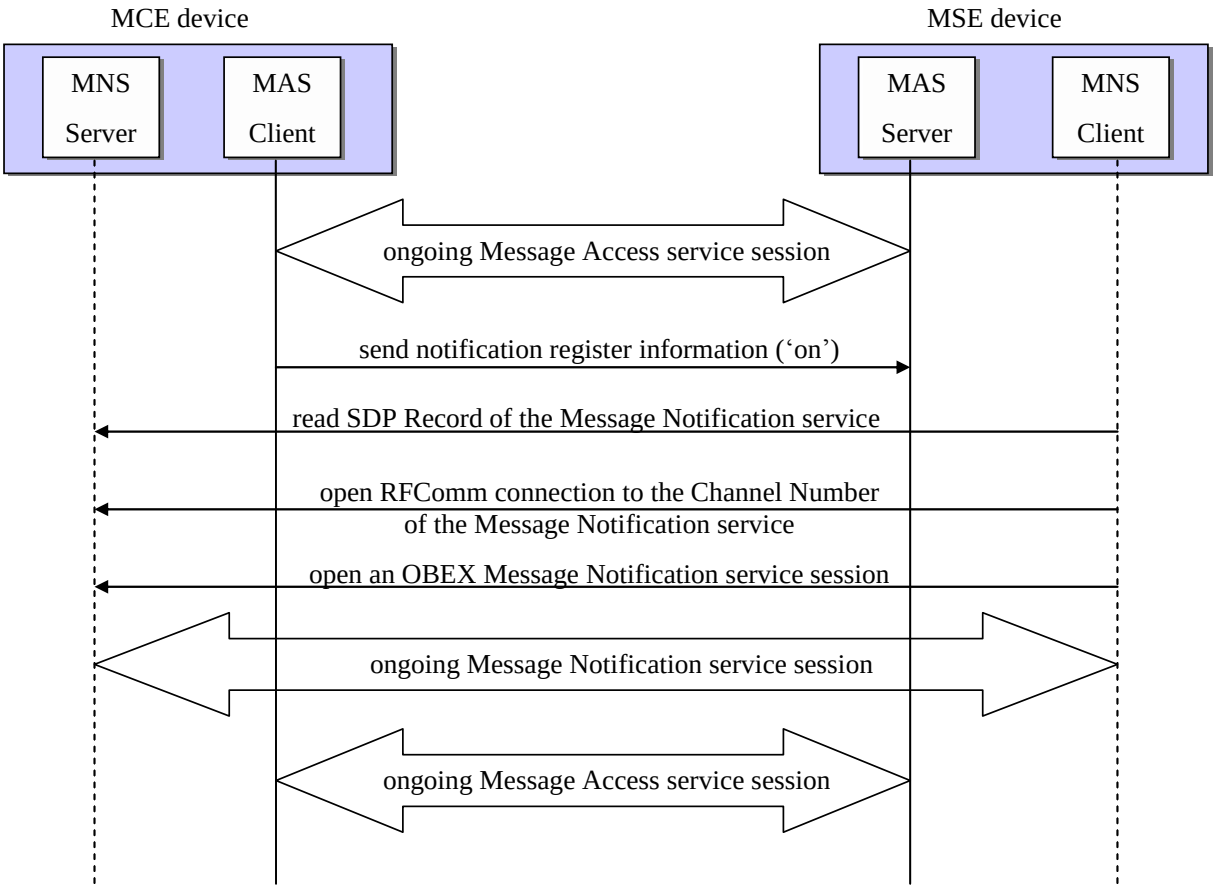

Figure 6.4: Establishment of MAP session (usage of Message Notification Service only)

The establishment of a Message Notification session is done in accordance with [10] with the MCE as OBEX Server and the MSE as OBEX Client.

The Message Access session shall be active before the establishment of the Message Notification session, but only the MAS Notification-Registration feature is required for this use case. If the Message Access session is disconnected after Message Notification session establishment, this will automatically indicate a MAS Notification-Deregistration for this MAS instance. For the UUID values to be used for the Target header see 6.3.

The establishment of a Message Notification session requires the previous establishment of a Message Access service session as described in chapter 6.4.1, to allow the MCE to register for a notification (function SetNotificationRegistration, 5.2).

6.4.4 Terminating a Message Access or Message Notification service session

The termination of a Message Access or a Message Notification service session is done in accordance with [3].

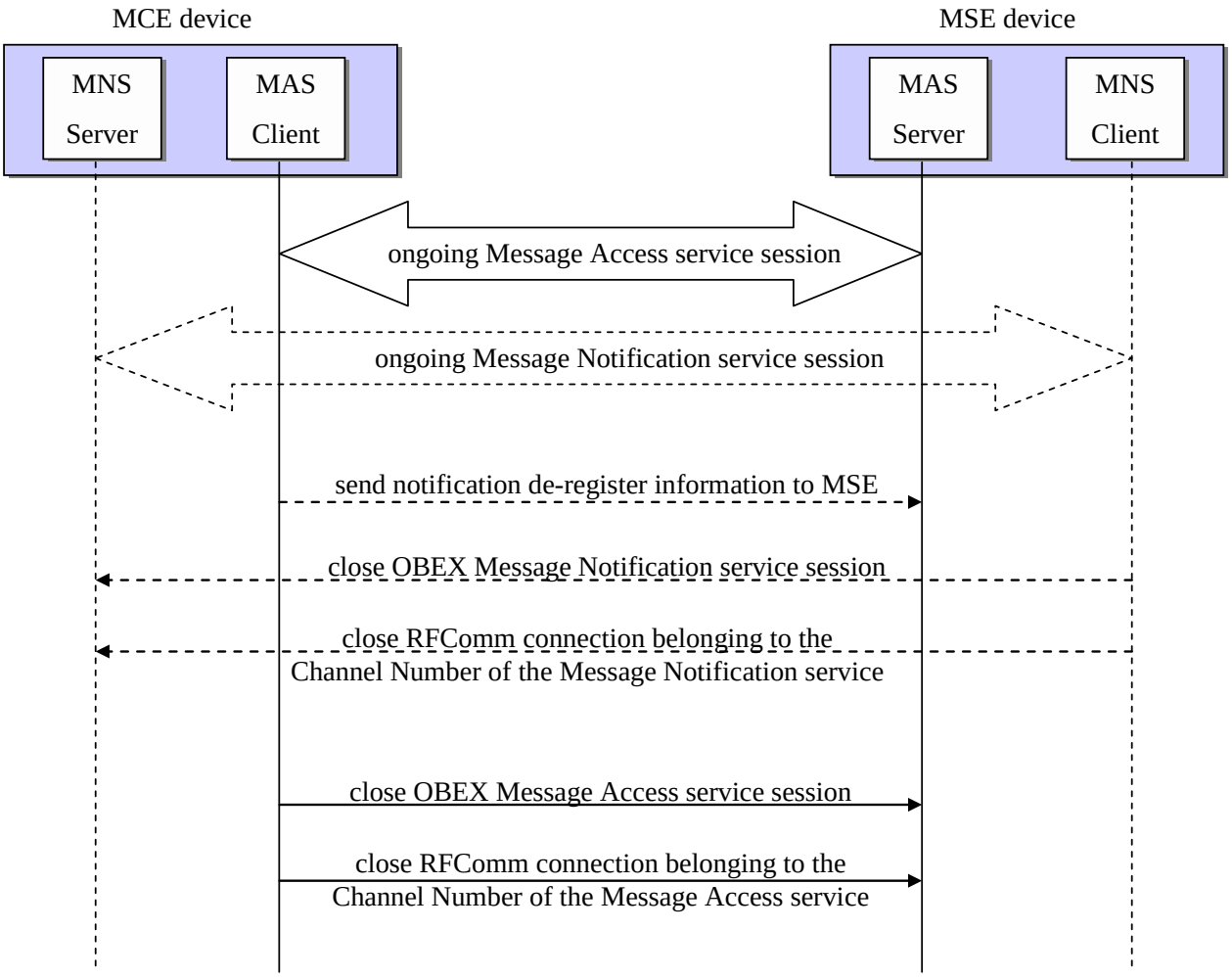

Figure 6.5: Termination of a MAP session (usage of Message Access and optional Message Notification Service)

The termination of multiple MAS sessions (MCE connected to several MAS instances) and an ongoing Message Notification service is described in the figure below. The MSE shall terminate the MNS service if the MCE has de-registered the MNS for all MAS-instances OR has terminated the MAS connections for all MAS-instances.

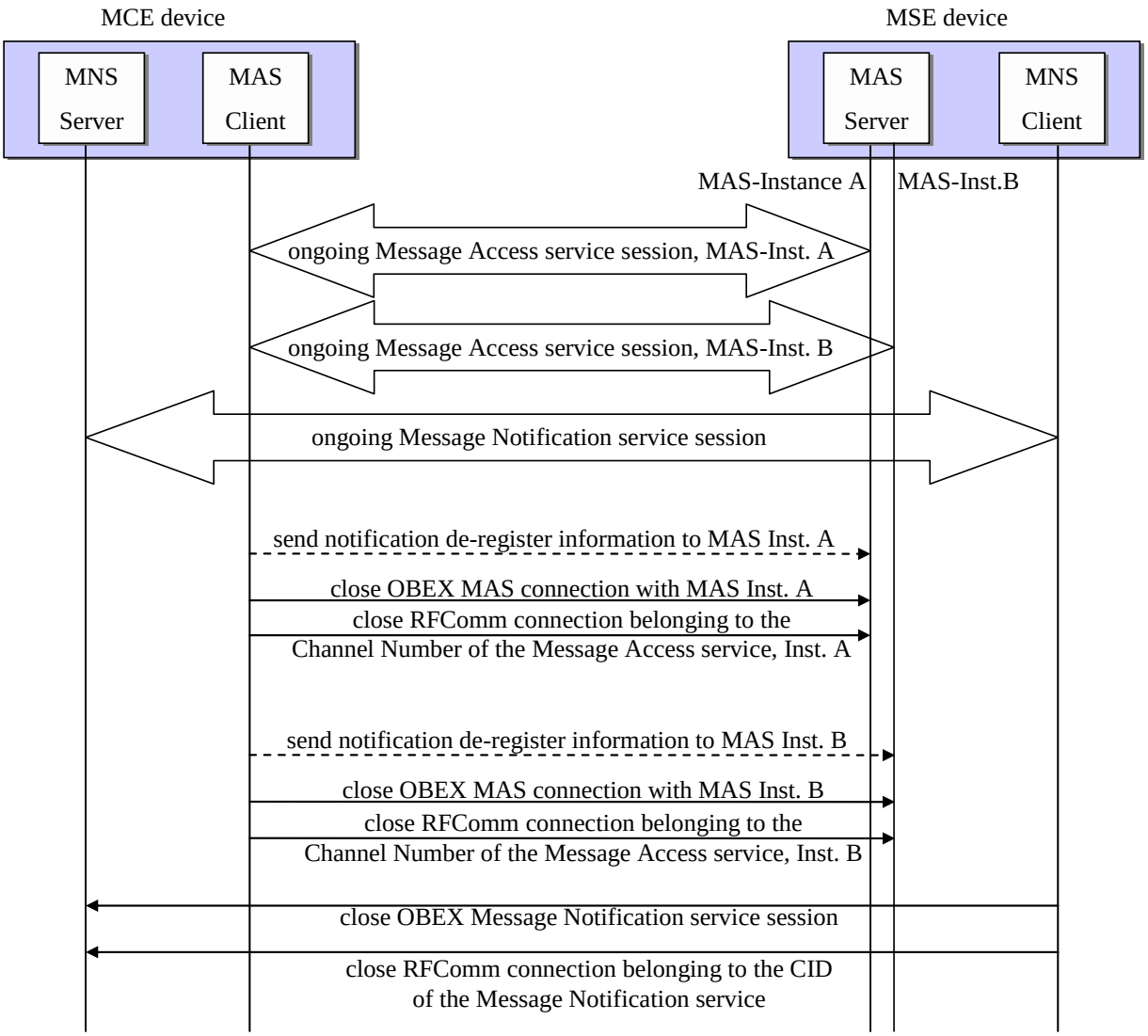

Figure 6.6: Termination of a MAP session with connections to multiple MAS instances and established Message Notification Service

6.4.5 Authentication

OBEX authentication shall not be used within MAP to avoid redundancy with the authentication on link level (see also chapter 2.6).

7 Service Discovery

7.1 SDP Interoperability Requirements

The following service records are defined for the Message Access Profile. There shall be one service record per MCE device (Message Notification Service, server role) and on the MSE device one for each MAS instance (Message Access Service, server role, see also 3.1.8). Up to 12 MAS Instances may be supported by a MSE device.

If a MCE requires connection to several MAS Instances it has to establish separate MAS connections to each of these MAS Instances where it shall use dedicated OBEX channels for the particular MAS connections. The MCE can differentiate the particular MAS Instances by its SDP Service record attributes ServiceName and MASInstanceID. Both values shall be unique for all SDP records of a MSE device. The value range of the MASInstanceID. shall be 0..255 (see 7.1.1).

7.1.1 SDP record for the Message Access service on the MSE device

Item	Definition	Type	Value	Status	Default
ServiceClassID List				M	
ServiceClass #0		UUID	Message Access Server	M	
Protocol Descriptor List				M	
Protocol #0		UUID	L2CAP	M	
Protocol #1		UUID	RFCOMM	M	
Param #0	Channel number	Uint8	N= Channel number	M	
Protocol #2		UUID	OBEX	M	
ServiceName	Displayable text name	String	Service-provider defined	M	"MAP MAS-name"
Bluetooth Profile Descriptor List				M	
Profile #0	Supported Profiles	UUID	Message Access Profile	M	
Param #0	Profile Version	Unit16		M	0x0101
MASInstanceID		Uint8		M	
SupportedMessageTypes		Uint8	Bit 0 = EMAIL	M	
			Bit 1 = SMS_GSM		
			Bit 2 = SMS_CDMA		
			Bit 3 = MMS		
			Bit 4 ~ 7 Reserved		

For MAP SDP UUID values please refer to the Bluetooth Assigned Numbers section of the Bluetooth SIG website.

7.1.2 SDP record for the Message Notification service on the MCE device

Item	Definition	Type	Value	Status	Default
ServiceClassID List				M	
ServiceClass #0		UUID	Message Notification Server	M	
Protocol Descriptor List				M	
Protocol #0		UUID	L2CAP	M	
Protocol #1		UUID	RFCOMM	M	
Param #0	Channel number	Uint8	N= Channel number	M	
Protocol #2		UUID	OBEX	M	
ServiceName	Displayable text name	String	Service-provider defined	M	"MAP MNS-name"
Bluetooth Profile Descriptor List				M	
Profile #0	Supported profiles	UUID	Message Access Profile	M	
Param #0	Profile version	Unit16		M	0x0101

For MAP SDP UUID values please refer to the Bluetooth Assigned Numbers section of the Bluetooth website.

7.2 Link Manager (LM) Interoperability Requirements

For the Link Manager layer, there are no additions to the requirements as stated in the Serial Port Profile.

7.3 Link Control (LC) Interoperability Requirements

In Table 7.1 changes to the support status as listed in the Serial Port Profile [16], Section 8.1, Table 8.1 are listed. The specification allows both MCEand MSE to support Inquiry and Inquiry scan. It is mandatory for MSE to support Inquiry and MCE to support Inquiry scan. It is optional for the MSE to support Inquiry Scan and for the MCE to support Inquiry though this support is strongly recommended.

Capability	Support in MSE	Support in MCE
Inquiry	M	O
Inquiry scan	O	M

Table 7.1: LC capabilities

7.3.1 Class of Device/Service Field

There is no obvious correlation between the Class of Device/Service Field and the support for the Message Access Profile. It is expected that many types of devices will implement the Message Access Profile.

8 Generic Access Profile

This profile requires compliance to the Generic Access Profile.

This section defines the support requirements for the capabilities as defined in the "Generic Access Profile" [2]. All “Mandatory,” “Optional,” “Excluded,” and “Conditional” properties are inherited from GAP. This chapter lists only those properties that differ from the requirements in GAP.

8.1 Modes

Table 8.1 shows the differences in the support status for GAP Modes in this profile.

	Procedure	Support in MCE	Support in MSE
	Discoverability modes		
1	General discoverable mode	M	M
	Pairing modes		
2	Pairable mode	M	M

Table 8.1: GAP Modes

8.2 Security Aspects

There is no change to the requirements as stated in the General Access Profile [2].

8.3 Idle Mode Procedures

The following table shows the differences to in the support status for the GAP Idle mode procedures within this profile:

	Procedure	Support in MCE	Support in MSE
1	General inquiry	O	M
2	Limited inquiry	O	O

Table 8.3: Idle mode procedures

9 List of Acronyms and Abbreviations

Abbreviation or Acronym	Meaning
3GPP	3rd Generation Partnership Project
bMessage	Bluetooth message format
CDMA	Code Division Multiple Access
CID	OBEX Connection Identifier
EDR	Enhanced Data Rate
GAP	Generic Access Profile
GSM	Global System for Mobile communication
HFP	Hands-Free-Profile
IO	Input/Output
L2CAP	Logical Link Control and Adaptation Protocol
LM	Link Manager
LMP	Link Manager Protocol
LMP Link	A Link Manager (LM) level link over which only Link Manager Protocol (LMP) commands are conveyed
MAP	Message Access Profile
MCE	Message Client Equipment
MMS	Multimedia Message Service
MSE	Message Server Equipment
MSISDN	Mobile Subscriber Integrated Services Digital Network Number (also appears as "Mobile Station International ISDN Number(s))
OBEX	Object Exchange
RFCOMM	Serial port transport protocol over L2CAP
SDP	Service Discovery Protocol
SMS	Short Message Service
UUID	Universally Unique Identifier

10 References

[1] Bluetooth Specification [vol3]; Core System Package [host volume]; Part B Service Discovery Protocol specification

[2] Bluetooth Specification [vol3]; Core System Package [host volume]; Part C Generic Access Profile specification

[3] Specification of the Bluetooth System, Bluetooth Generic Object Exchange Profile specification

[4] Specification of Bluetooth System, Bluetooth SIG Assigned Numbers

[5] vCard The Electronic Business Card, version 2.1, September 18th

[6] A MIME Content-Type for Directory Information, IETF, Network Working Group, RFC 2425, September 1998

[7] vCard MIME Directory Profile, IETF, Network Working Group, RFC 2426, September 1998

[8] Specification for Ir Mobile Communication (IrMC), Infrared Data Association, March 01, 1999

[9] Multipurpose Internet Mail Extensions (MIME):
RFC 2045 Part 1: Format of Internet Message Bodies,
RFC 2046 Part 2: Media Types,
RFC 2047,Part 3: Message Header Extensions for Non-ASCII Text

[10] IrDA Object Exchange Protocol OBEX™ with Published Errata, Version 1.2, April 1999, Counterpoint Systems Foundry Inc. and Microsoft Corporation

[11] Specification of the Bluetooth System, Bluetooth Service Discovery Application Profile specification

[12] Technical Specification 3rd Generation Partnership Project; Technical Specification Group Terminals; Technical realization of the Short Message Service (SMS), 3GPP TS 23.040 V8.20.0 (2008-06)

[13] Technical Specification 3rd Generation Partnership Project; Technical Specification Group Terminals; Technical realization of the Multimedia Messaging Service (MMS), 3GPP TS 23.140 V7.0.0 (2008-03)

[14] TIA/EIA-637-A, Short Message Service, 3GPP2 C.S0015-0

[15] Bluetooth Specification [vol0]; Part E IEEE Language

[16] Bluetooth Specification [vol7]; Core System Package [host volume]; Part B, Serial Port Profile specification

[17] Administration of Parameter Value Assignments for cdma2000 Spread Spectrum Standards, Release A (TSB58-A), 3GPP2 C.R1001-A 2.0

[18] 3rd Generation Partnership Project, Technical Specification Group Core Network and Terminals; Alphabets and language-specific information (Release 8), 3GPP TS 23.038 V8.1.0 (2008-06)

[19] RFC 2822, Internet Message Format

[20] Extensible Markup Language (XML) 1.0 (Fourth Edition) W3C Recommendation 29 September 2006, Definitions, http://www.w3.org/TR/xml/

[21] Bluetooth Specification [vol1], Architecture & Terminology Overview;
Part A, Architecture

[22] Bluetooth Specification [vol2], Core System Package [Controller volume];
Part H, Security Specification

[23] Technical Specification 3rd Generation Partnership Project; Technical realization of the Short Message Service (SMS): 3GPP TS 04.11 version 7.1.0 (2000-09)

[24] International Reference Alphabet (IRA) (formerly International Alphabet No. 5 or IA5)—Information Technology 7-Bit Coded Character Set for Information Interchange, recommendation T.50,International Telegraph and Telephone Consultative Committee, September, 1992