

**Open Service Access (OSA);  
Application Programming Interface (API);  
Part 15: Multi-Media Messaging SCF  
(Parlay 5)**



---

Reference

RES/TISPAN-01029-15-OSA

---

Keywords

API, IDL, OSA, UML

**ETSI**

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

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

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

---

**Important notice**

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, please send your comment to one of the following services:

[http://portal.etsi.org/chaicor/ETSI\\_support.asp](http://portal.etsi.org/chaicor/ETSI_support.asp)

---

**Copyright Notification**

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2007.

© The Parlay Group 2007.

All rights reserved.

**DECT™**, **PLUGTESTS™** and **UMTS™** are Trade Marks of ETSI registered for the benefit of its Members.  
**TIPHON™** and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.  
**3GPP™** is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

# Contents

|  |    |
|--|----|
| Intellectual Property Rights .....                               | 7  |
| Foreword.....  | 7  |
| 1 Scope .....  | 8  |
| 2 References .....   | 8  |
| 3 Definitions and abbreviations.....                             | 8  |
| 3.1 Definitions .....  | 8  |
| 3.2 Abbreviations .....  | 8  |
| 4 Multi-Media Messaging SCF .....                                | 9  |
| 5 Sequence Diagrams .....  | 9  |
| 5.1 Sending messages and receiving delivery notification .....   | 9  |
| 5.2 Sending, and receiving messages in same context .....        | 11 |
| 5.3 Setting notification of received messages .....              | 12 |
| 5.4 Using Mailbox functions .....                                | 13 |
| 5.5 Using Mailbox to send and receive .....                      | 14 |
| 6 Class Diagrams.....  | 16 |
| 7 The Service Interface Specifications .....                     | 17 |
| 7.1 Interface Specification Format .....                         | 17 |
| 7.1.1 Interface Class .....                                      | 17 |
| 7.1.2 Method descriptions.....                                   | 17 |
| 7.1.3 Parameter descriptions.....                                | 17 |
| 7.1.4 State Model.....   | 17 |
| 7.2 Base Interface.....  | 17 |
| 7.2.1 Interface Class IpInterface .....                          | 17 |
| 7.3 Service Interfaces .....                                     | 18 |
| 7.3.1 Overview .....   | 18 |
| 7.4 Generic Service Interface .....                              | 18 |
| 7.4.1 Interface Class IpService .....                            | 18 |
| 7.4.1.1 Method setCallback() .....                               | 18 |
| 7.4.1.2 Method setCallbackWithSessionID().....                   | 18 |
| 8 Multi-Media Messaging Interface Classes .....                  | 19 |
| 8.1 Interface Class IpMultiMediaMessagingManager .....           | 19 |
| 8.1.1 Method openMailbox() .....                                 | 20 |
| 8.1.2 Method openMultiMediaMessaging() .....                     | 21 |
| 8.1.3 Method createNotification().....                           | 22 |
| 8.1.4 Method destroyNotification() .....                         | 22 |
| 8.1.5 Method changeNotification().....                           | 22 |
| 8.1.6 Method getNextNotification() .....                         | 23 |
| 8.1.7 Method enableNotifications() .....                         | 23 |
| 8.1.8 Method disableNotifications() .....                        | 24 |
| 8.2 Interface Class IpAppMultiMediaMessagingManager.....         | 24 |
| 8.2.1 Method mailboxTerminated() .....                           | 25 |
| 8.2.2 Method reportNotification().....                           | 25 |
| 8.2.3 Method notificationsInterrupted().....                     | 25 |
| 8.2.4 Method notificationsResumed().....                         | 26 |
| 8.2.5 Method multiMediaMessagingTerminated() .....               | 26 |
| 8.2.6 Method terminateMultipleMailboxes().....                   | 26 |
| 8.2.7 Method terminateMultipleMultiMediaMessagingSessions()..... | 26 |
| 8.3 Interface Class IpMailbox .....                              | 27 |
| 8.3.1 Method close() .....                                       | 28 |
| 8.3.2 Method getMessageInfoPropertiesReq() .....                 | 28 |
| 8.3.3 Method setMessageInfoPropertiesReq().....                  | 28 |

|        |  |    |
|--------|--|----|
| 8.3.4  | Method createFolderReq()                 | 29 |
| 8.3.5  | Method getFoldersReq()                   | 29 |
| 8.3.6  | Method deleteFolderReq()                 | 30 |
| 8.3.7  | Method copyFolderReq()                   | 30 |
| 8.3.8  | Method moveFolderReq()                   | 31 |
| 8.3.9  | Method putMessageReq()                   | 31 |
| 8.3.10 | Method copyMessageReq()                  | 32 |
| 8.3.11 | Method moveMessageReq()                  | 32 |
| 8.3.12 | Method deleteMessageReq()                | 33 |
| 8.3.13 | Method listMessagesReq()                 | 33 |
| 8.3.14 | Method listMessageBodyPartsReq()         | 34 |
| 8.3.15 | Method getMessageBodyPartsReq()          | 35 |
| 8.3.16 | Method getMessageHeadersReq()            | 35 |
| 8.3.17 | Method getMessageContentReq()            | 36 |
| 8.3.18 | Method getFullMessageReq()               | 36 |
| 8.3.19 | Method getMailboxInfoPropertiesReq()     | 37 |
| 8.3.20 | Method getFolderInfoPropertiesReq()      | 37 |
| 8.4    | Interface Class IpAppMailbox             | 38 |
| 8.4.1  | Method getMessageInfoPropertiesRes()     | 40 |
| 8.4.2  | Method setMessageInfoPropertiesRes()     | 40 |
| 8.4.3  | Method setMessageInfoPropertiesErr()     | 40 |
| 8.4.4  | Method getMailboxInfoPropertiesErr()     | 41 |
| 8.4.5  | Method getFolderInfoPropertiesErr()      | 41 |
| 8.4.6  | Method getMessageInfoPropertiesErr()     | 41 |
| 8.4.7  | Method createFolderRes()                 | 42 |
| 8.4.8  | Method createFolderErr()                 | 42 |
| 8.4.9  | Method getFoldersRes()                   | 43 |
| 8.4.10 | Method getFoldersErr()                   | 43 |
| 8.4.11 | Method deleteFolderRes()                 | 43 |
| 8.4.12 | Method deleteFolderErr()                 | 44 |
| 8.4.13 | Method copyFolderRes()                   | 44 |
| 8.4.14 | Method copyFolderErr()                   | 44 |
| 8.4.15 | Method moveFolderRes()                   | 45 |
| 8.4.16 | Method moveFolderErr()                   | 45 |
| 8.4.17 | Method putMessageRes()                   | 45 |
| 8.4.18 | Method putMessageErr()                   | 46 |
| 8.4.19 | Method copyMessageRes()                  | 46 |
| 8.4.20 | Method copyMessageErr()                  | 46 |
| 8.4.21 | Method moveMessageRes()                  | 47 |
| 8.4.22 | Method moveMessageErr()                  | 47 |
| 8.4.23 | Method deleteMessageRes()                | 47 |
| 8.4.24 | Method deleteMessageErr()                | 47 |
| 8.4.25 | Method listMessagesRes()                 | 48 |
| 8.4.26 | Method listMessagesErr()                 | 48 |
| 8.4.27 | Method listMessageBodyPartsRes()         | 49 |
| 8.4.28 | Method listMessageBodyPartsErr()         | 49 |
| 8.4.29 | Method getMessageBodyPartsRes()          | 49 |
| 8.4.30 | Method getMessageBodyPartsErr()          | 50 |
| 8.4.31 | Method getMessageHeadersRes()            | 50 |
| 8.4.32 | Method getMessageHeadersErr()            | 50 |
| 8.4.33 | Method getMessageContentRes()            | 51 |
| 8.4.34 | Method getMessageContentErr()            | 51 |
| 8.4.35 | Method getFullMessageRes()               | 52 |
| 8.4.36 | Method getFullMessageErr()               | 52 |
| 8.4.37 | Method getMailboxInfoPropertiesRes()     | 52 |
| 8.4.38 | Method getFolderInfoPropertiesRes()      | 53 |
| 8.5    | Interface Class IpMultiMediaMessaging    | 53 |
| 8.5.1  | Method sendMessageReq()                  | 53 |
| 8.5.2  | Method cancelMessageReq()                | 55 |
| 8.5.3  | Method queryStatusReq()                  | 55 |
| 8.5.4  | Method close()                           | 55 |
| 8.6    | Interface Class IpAppMultiMediaMessaging | 56 |

|         |  |    |
|---------|--|----|
| 8.6.1   | Method sendMessageRes() .....                  | 56 |
| 8.6.2   | Method sendMessageErr() .....                  | 57 |
| 8.6.3   | Method cancelMessageRes() .....                | 57 |
| 8.6.4   | Method cancelMessageErr() .....                | 57 |
| 8.6.5   | Method queryStatusRes() .....                  | 58 |
| 8.6.6   | Method queryStatusErr() .....                  | 58 |
| 8.6.7   | Method messageStatusReport() .....             | 58 |
| 8.6.8   | Method messageReceived() .....                 | 59 |
| 9       | State Transition Diagrams .....                | 59 |
| 10      | Multi-Media Messaging Service Properties ..... | 59 |
| 11      | Data Definitions .....                         | 60 |
| 11.1    | Multi-Media Messaging data definitions .....   | 60 |
| 11.1.1  | IpMultiMediaMessagingManager .....             | 60 |
| 11.1.2  | IpMultiMediaMessagingManagerRef .....          | 60 |
| 11.1.3  | IpAppMultiMediaMessagingManager .....          | 60 |
| 11.1.4  | IpAppMultiMediaMessagingManagerRef .....       | 60 |
| 11.1.5  | IpMailbox .....                                | 60 |
| 11.1.6  | IpMailboxRef .....                             | 60 |
| 11.1.7  | IpAppMailbox .....                             | 60 |
| 11.1.8  | IpAppMailboxRef .....                          | 60 |
| 11.1.9  | IpMultiMediaMessaging .....                    | 60 |
| 11.1.10 | IpMultiMediaMessagingRef .....                 | 60 |
| 11.1.11 | IpAppMultiMediaMessaging .....                 | 61 |
| 11.1.12 | IpAppMultiMediaMessagingRef .....              | 61 |
| 11.1.13 | TpBodyPartDescription .....                    | 61 |
| 11.1.14 | TpBodyPartDescriptionList .....                | 61 |
| 11.1.15 | TpBodyPart .....                               | 61 |
| 11.1.16 | TpBodyPartList .....                           | 61 |
| 11.1.17 | TpDeliveryTime .....                           | 62 |
| 11.1.18 | TpDeliveryTimeType .....                       | 62 |
| 11.1.19 | TpFolderInfoProperty .....                     | 62 |
| 11.1.20 | TpFolderInfoPropertyName .....                 | 62 |
| 11.1.21 | TpFolderInfoPropertySet .....                  | 62 |
| 11.1.22 | TpGenericHeaderField .....                     | 63 |
| 11.1.23 | TpListMessagesCriteria .....                   | 63 |
| 11.1.24 | TpMailboxFolderStatusInformation .....         | 63 |
| 11.1.25 | TpMailboxIdentifier .....                      | 63 |
| 11.1.26 | TpMailboxIdentifierSet .....                   | 63 |
| 11.1.27 | TpMailboxInfoProperty .....                    | 63 |
| 11.1.28 | TpMailboxInfoPropertyName .....                | 64 |
| 11.1.29 | TpMailboxInfoPropertySet .....                 | 64 |
| 11.1.30 | TpMailboxMessageStatus .....                   | 64 |
| 11.1.31 | TpMessageDeliveryType .....                    | 65 |
| 11.1.32 | TpMessageInfoProperty .....                    | 65 |
| 11.1.33 | TpMessageInfoPropertyName .....                | 65 |
| 11.1.34 | TpMessageInfoPropertySet .....                 | 65 |
| 11.1.35 | TpMessageHeaderFieldType .....                 | 66 |
| 11.1.36 | TpMessageHeaderField .....                     | 67 |
| 11.1.37 | TpMessageHeaderFieldSet .....                  | 67 |
| 11.1.38 | TpMessagePriority .....                        | 67 |
| 11.1.39 | TpMessageDeliveryReportType .....              | 68 |
| 11.1.40 | TpMessageTreatment .....                       | 68 |
| 11.1.41 | TpMessageTreatmentType .....                   | 68 |
| 11.1.42 | TpMessageTreatmentSet .....                    | 68 |
| 11.1.43 | TpMultiMediaMessagingIdentifier .....          | 68 |
| 11.1.44 | TpMultiMediaMessagingIdentifierSet .....       | 69 |
| 11.1.45 | TpQueryStatusReport .....                      | 69 |
| 11.1.46 | TpQueryStatusReportSet .....                   | 69 |
| 11.1.47 | TpTerminatingAddressList .....                 | 69 |
| 11.2    | Event Notification data definitions .....      | 69 |

|                               |  |           |
|-------------------------------|--|-----------|
| 11.2.1                        | TpMessagingEventName .....   | 69        |
| 11.2.2                        | TpMessagingEventCriteria .....                                     | 70        |
| 11.2.3                        | TpMessagingEventCriteriaSet .....                                  | 70        |
| 11.2.4                        | TpNewMailboxMessageArrivedCriteria .....                           | 70        |
| 11.2.5                        | TpNewMessageArrivedCriteria .....                                  | 70        |
| 11.2.6                        | TpMessagingEventInfo .....   | 70        |
| 11.2.7                        | TpMessagingEventInfoSet .....                                      | 71        |
| 11.2.8                        | TpNewMailboxMessageArrivedInfo .....                               | 71        |
| 11.2.9                        | TpNewMessageArrivedInfo .....                                      | 71        |
| 11.2.10                       | TpMessageDescription .....   | 71        |
| 11.2.11                       | TpMessageDescriptionList .....                                     | 71        |
| 11.2.12                       | TpMessagingNotificationRequested .....                             | 72        |
| 11.2.13                       | TpMessagingNotificationRequestedSet .....                          | 72        |
| 11.2.14                       | TpMessagingNotificationRequestedSetEntry .....                     | 72        |
| 11.3                          | Error type data definitions .....                                  | 72        |
| 11.3.1                        | TpMessageInfoPropertyError .....                                   | 72        |
| 11.3.2                        | TpMessagingError .....   | 73        |
| 11.3.3                        | TpMessageInfoPropertyErrorSet .....                                | 73        |
| 11.3.4                        | TpSetPropertyError .....   | 73        |
| 12                            | Exception Classes .....  | 74        |
| <b>Annex A (normative):</b>   | <b>OMG IDL Description of Multi-Media Messaging SCF .....</b>      | <b>75</b> |
| <b>Annex B (informative):</b> | <b>W3C WSDL Description of Multi-Media Messaging SCF .....</b>     | <b>76</b> |
| <b>Annex C (informative):</b> | <b>Java API Description of the Multi-Media Messaging SCF .....</b> | <b>77</b> |
| <b>Annex D (informative):</b> | <b>Contents of 3GPP OSA R6 Multi-Media Messaging .....</b>         | <b>78</b> |
| <b>Annex E (informative):</b> | <b>Record of changes .....</b>                                     | <b>79</b> |
| E.1                           | Interfaces .....   | 79        |
| E.1.1                         | New .....  | 79        |
| E.1.2                         | Deprecated .....   | 79        |
| E.1.3                         | Removed .....  | 79        |
| E.2                           | Methods .....  | 79        |
| E.2.1                         | New .....  | 79        |
| E.2.2                         | Deprecated .....   | 80        |
| E.2.3                         | Modified .....   | 80        |
| E.2.4                         | Removed .....  | 80        |
| E.3                           | Data Definitions .....   | 80        |
| E.3.1                         | New .....  | 80        |
| E.3.2                         | Modified .....   | 80        |
| E.3.3                         | Removed .....  | 81        |
| E.4                           | Service Properties .....   | 81        |
| E.4.1                         | New .....  | 81        |
| E.4.2                         | Deprecated .....   | 81        |
| E.4.3                         | Modified .....   | 81        |
| E.4.4                         | Removed .....  | 81        |
| E.5                           | Exceptions .....   | 82        |
| E.5.1                         | New .....  | 82        |
| E.5.2                         | Modified .....   | 82        |
| E.5.3                         | Removed .....  | 82        |
| E.6                           | Others .....   | 82        |
| History                       | .....  | 83        |

---

## Intellectual Property Rights

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

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

---

## Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN).

The present document is part 15 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 203 915) is structured in the following parts:

- Part 1: "Overview";
- Part 2: "Common Data Definitions";
- Part 3: "Framework";
- Part 4: "Call Control";
- Part 5: "User Interaction SCF";
- Part 6: "Mobility SCF";
- Part 7: "Terminal Capabilities SCF";
- Part 8: "Data Session Control SCF";
- Part 9: "Generic Messaging SCF";
- Part 10: "Connectivity Manager SCF";
- Part 11: "Account Management SCF";
- Part 12: "Charging SCF";
- Part 13: "Policy Management SCF";
- Part 14: "Presence and Availability Management SCF";
- Part 15: "Multi-Media Messaging SCF".**

The present document has been defined jointly between ETSI, The Parlay Group (<http://www.parlay.org>) and the 3GPP, in co-operation with a number of JAIN™ Community (<http://www.java.sun.com/products/jain>) member companies.

**The present document forms part of the Parlay 5.1 set of specifications.**

**The present document is equivalent to 3GPP TS 29.198-15 V6.3.0 (Release 6).**

---

# 1 Scope

The present document is part 15 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs.

The present document specifies the Multi-Media Messaging Service Capability Feature (SCF) aspects of the interface. All aspects of the Multi-Media Messaging SCF are defined here, these being:

- Sequence Diagrams.
- Class Diagrams.
- Interface specification plus detailed method descriptions.
- State Transition diagrams.
- Data Definitions.
- IDL Description of the interfaces.

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

---

# 2 References

The references listed in clause 2 of ES 203 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 203 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 5)".

---

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 203 915-1 apply.

## 3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 203 915-1 apply.



---

## 4 Multi-Media Messaging SCF

The following clauses describe each aspect of the Multi-Media Messaging Service Capability Feature (SCF).

The order is as follows:

- The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.
- The Class relationships clause shows how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.
- The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data Definitions clause shows a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part ES 203 915-2.

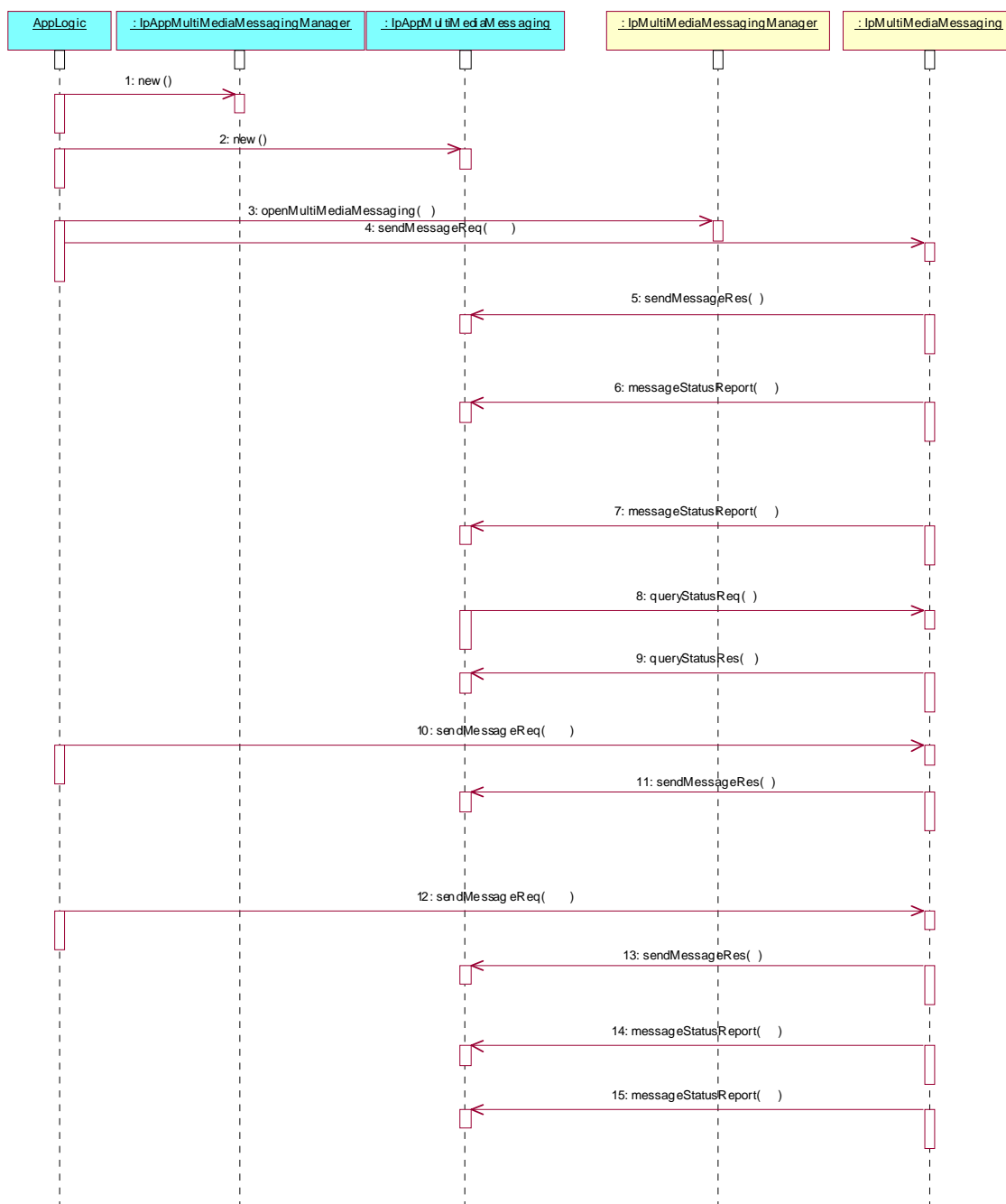
An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method. Where a method is not supported by an implementation of a Service interface, the exception `P_METHOD_NOT_SUPPORTED` shall be returned to any call of that method.

---

## 5 Sequence Diagrams

### 5.1 Sending messages and receiving delivery notification

This sequence diagram shows how the application can send messages on the `IpMultiMediaMessaging` interface with `sendMessageReq()`, and how the application can be informed about the delivery status of the message with `messageStatusReport()`. It also shows how the application can query the delivery status of a message, with `queryStatusReq()`.



3: Request the opening of a MultiMedia Messaging object. The application intends to use this object to send messages to multiple destinations, so it has not specified any defaultDestinationAddressList.

4: The application sends a message. The destination address is included in the destinationAddressList parameter. If the source address was not provided when the IpMultiMediaMessaging object was created, it can be provided in the sourceAddress parameter. The application has requested delivery receipt and read receipt in the messageTreatment parameter. The assignmentID received as a return parameter enables the application to match any message status information with this message.

5: This method indicates successful processing of the sendMessageReq by the SCF, and that the message has been sent. It does not indicate a delivery status.

6: This method contains a delivery receipt for the message just sent.

7: This method contains a read receipt for the message just sent.

8: The application queries the status of the message it has sent (to verify the read receipt? or it has discarded the read receipt?).

9: The status of the message is returned.

10: The application sends another message, this time to a different destination. It has requested a read receipt to be returned.

11: This method indicates successful processing of the sendMessageReq by the SCF, and that the message has been sent. It does not indicate a delivery status.

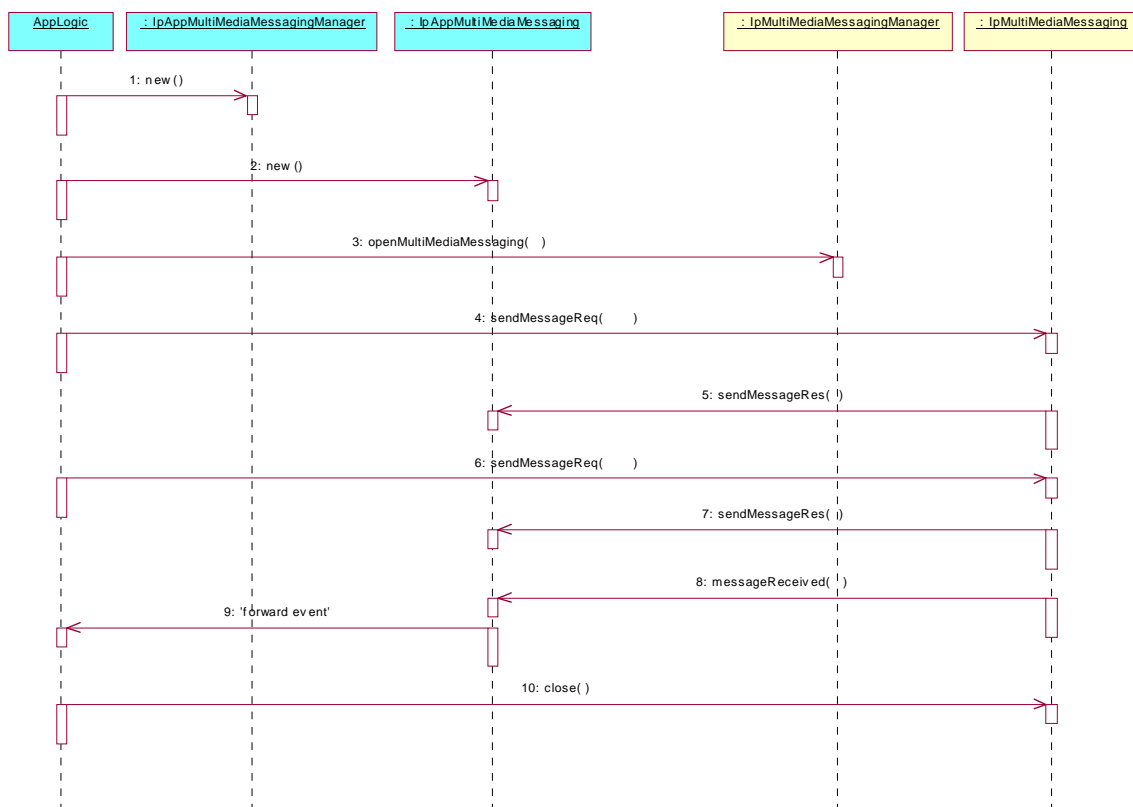
12: The application sends another message, to a different destination. It has requested a read receipt to be returned.

14: This method contains an indication that the previous message has been read.

15: This method contains an indication that the second message has been read. The assignmentID is used to match this report to the corresponding sendMessageReq().

## 5.2 Sending, and receiving messages in same context

This sequence diagram shows how the application can send and receive messages within the same communication context using sendMessageReq() on the IpMultiMediaMessaging interface and messageReceived() on the IpAppMultiMediaMessaging interface.



3: Request the opening of a MultiMedia Messaging object. The application intends to use this object to send messages to the same destination, so it has specified the defaultdestinationAddressList. The defaultSourceAddress is also specified.

4: The application sends a message. The application has not included a destination address in the destinationAddressList parameter, as a default value has already been supplied in the openMultiMediaMessaging() method. Likewise the default source address was provided when the IpMultiMediaMessaging object was created, so there is no need to provide the sourceAddress parameter. The application has not requested delivery receipt or read receipt in the messageTreatment parameter.

5: This method indicates successful processing of the sendMessageReq by the SCF, and that the message has been sent. It does not indicate a delivery status.

6: The application sends another message to the same destination, again using default values for the destination and source addresses.

7: This method indicates successful processing of the sendMessageReq by the SCF, and that the message has been sent. It does not indicate a delivery status.

8: A new message is received in this communication context. The full message contents are carried in this method. It is not specified how the SCF identifies that this message is to be delivered in this communication context. The SCF could use source or destination addresses, content type, time or subject, among other parameters, to identify the context.

10: The application closes the session, i.e. closes the communication context.

### 5.3 Setting notification of received messages

This sequence diagram shows how the application can subscribe to notifications, and how it can receive messages using reportNotifications() method.



3: The application requests the opening of a MultiMedia Messaging object.

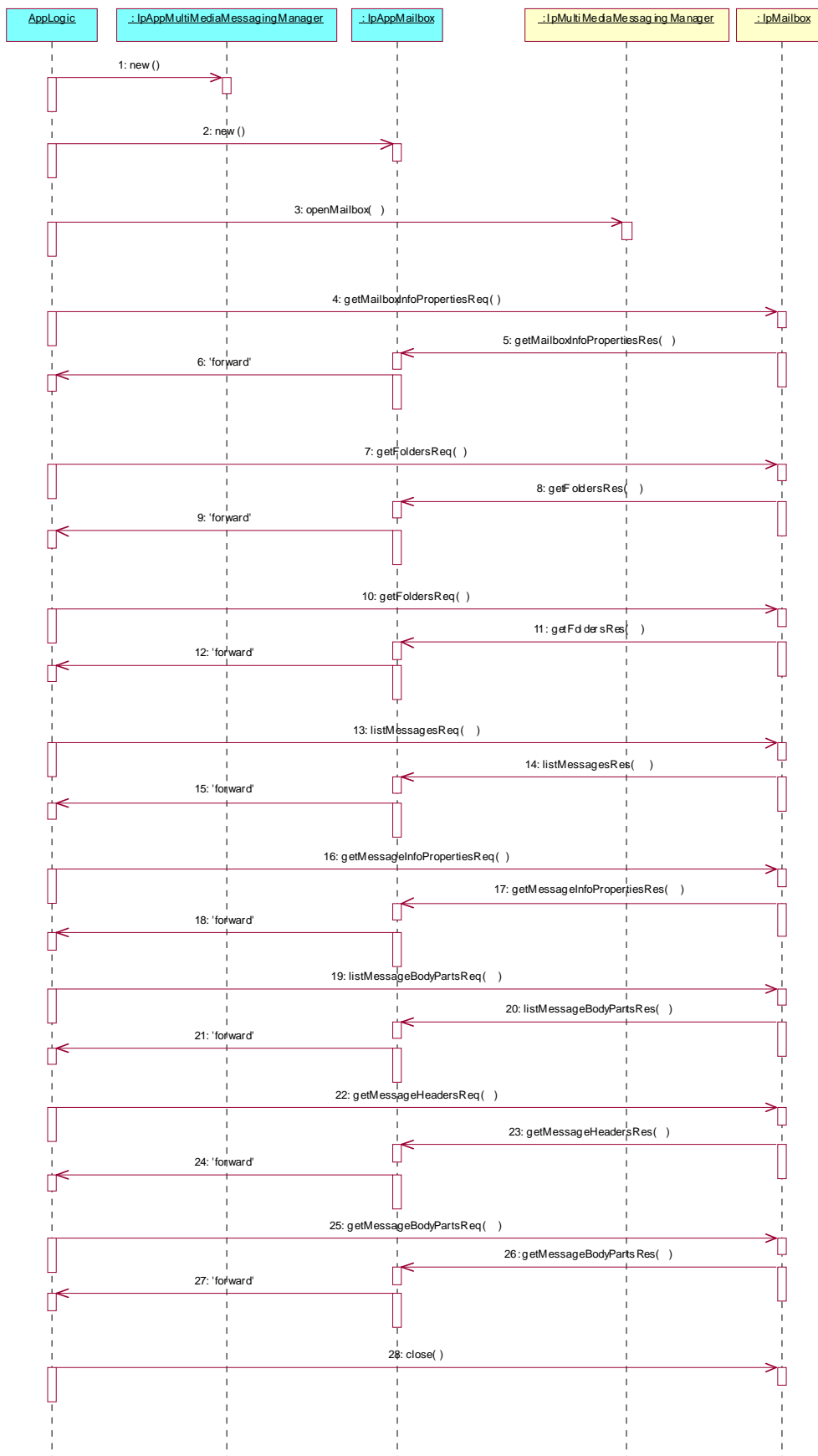
4: The application requests to be notified of any messages received for a particular destination address, using the P\_EVENT\_MSG\_NEW\_MESSAGE\_ARRIVED criteria. The application may request that a MultiMedia Messaging session is created upon receipt of a message.

5: A message is received for the destination address identified in the createNotification() method. In this type of event (P\_EVENT\_MSG\_NEW\_MESSAGE\_ARRIVED), the entire message contents are delivered in the reportNotification() method.

7: The application is no longer interested in receiving notifications of received messages. It de-subscribes from notification of received messages.

## 5.4 Using Mailbox functions

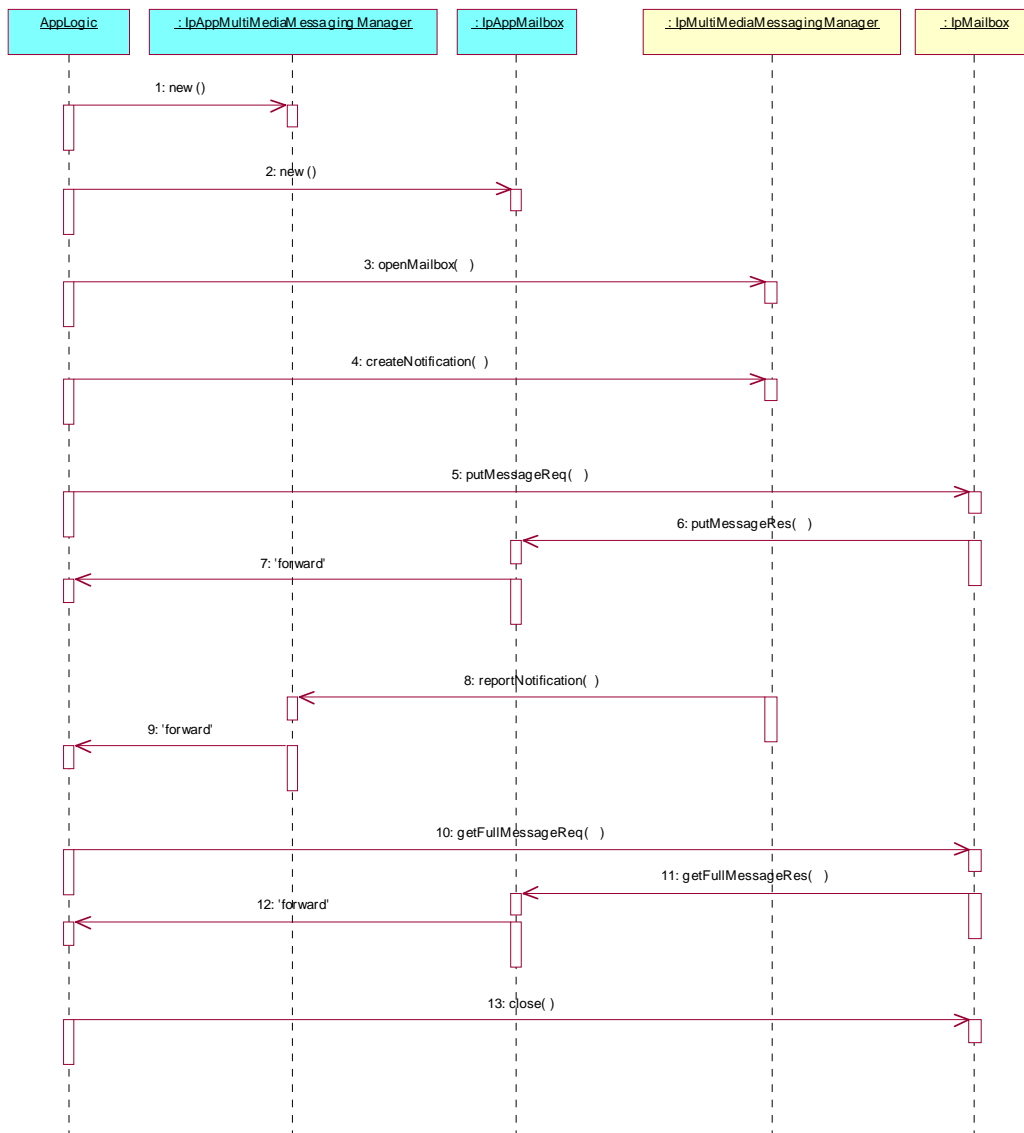
This sequence diagram shows how an application can retrieve message details from the mailbox.



- 3: The application requests to open the mailbox identified by the mailboxID parameter.
- 4: The application requests the properties of the Mailbox.
- 5: The property set of the mailbox is returned. The properties include the owner, date created, date changed and size of the mailbox.
- 7: The application requests a list of the top-level folders in the mailbox. The folderID parameter is left empty.
- 8: The list of top-level folders is returned to the application.
- 10: The application requests a list of the sub-folders in a folder returned earlier. The folderID parameter identifies the folder for which the list of sub-folders is requested.
- 11: The list of sub-folders is returned to the application.
- 13: The application requests a list of messages in a folder returned earlier. The folderID parameter identifies the folder for which the list of messages is requested.
- 14: The list of messages in the folder is returned.
- 16: The application requests the property set of a message returned earlier. The message is identified by its messageID, returned in the listMessagesRes().
- 17: The message properties are returned. These properties may include the date created, date received, date changed, size or status.
- 19: The application requests a list of the body parts of the message identified in the messageID parameter. The location of the message is identified in the folderID parameter.
- 20: The list of the message parts is returned.
- 22: The application requests the set of headers of the message, identified by the messageID parameter. The location of the message is identified in the folderID parameter.
- 23: The list of headers is returned to the application.
- 25: The application retrieves one or more parts of the message, as identified in the partIDs parameter.
- 26: The contents of the requested message parts are returned.
- 28: The application closes the mailbox session.

## 5.5 Using Mailbox to send and receive

This sequence diagram shows how an application can use a mailbox to send and receive messages, if this functionality is supported by the SCF.



3: The application requests to open the mailbox identified by the mailboxID parameter.

4: The application requests to be notified of any messages received in the specified mailbox, using the P\_EVENT\_MSG\_NEW\_MAILBOX\_MESSAGE\_ARRIVED criteria.

5: The application places a message in a folder in the mailbox. The message contents are specified in the message parameter, and the folder in which to place it is specified in the folderID. The application chooses to place the message in the folder identified in the service property P\_PUT\_MESSAGE\_FOLDER\_TO\_SEND. Any message placed in this folder is automatically sent. Typically it could be an Outbox folder.

6: This method indicates that the message has been successfully placed in the specified folder, and a messageID is returned. This does not necessarily indicate that the message has been sent, nor does it indicate that it has been delivered or received.

8: The application is notified that a message has been received in the mailbox. In this type of event (P\_EVENT\_MSG\_NEW\_MAILBOX\_MESSAGE\_ARRIVED), the messageID, location of the message and message description are delivered in the reportNotification() method, but the message contents are not. These need to be retrieved from the mailbox.

10: Using the messageID and folderID received in the reportNotification() method, the application requests to retrieve the full contents of the received message from the mailbox. The application could have chosen to retrieve individual parts of the message using getMessageBodyPartsReq(), or to retrieve just the headers using getMessageHeadersReq().

11: The full contents of the message are returned to the application.

13: The application closes the mailbox session.

## 6 Class Diagrams

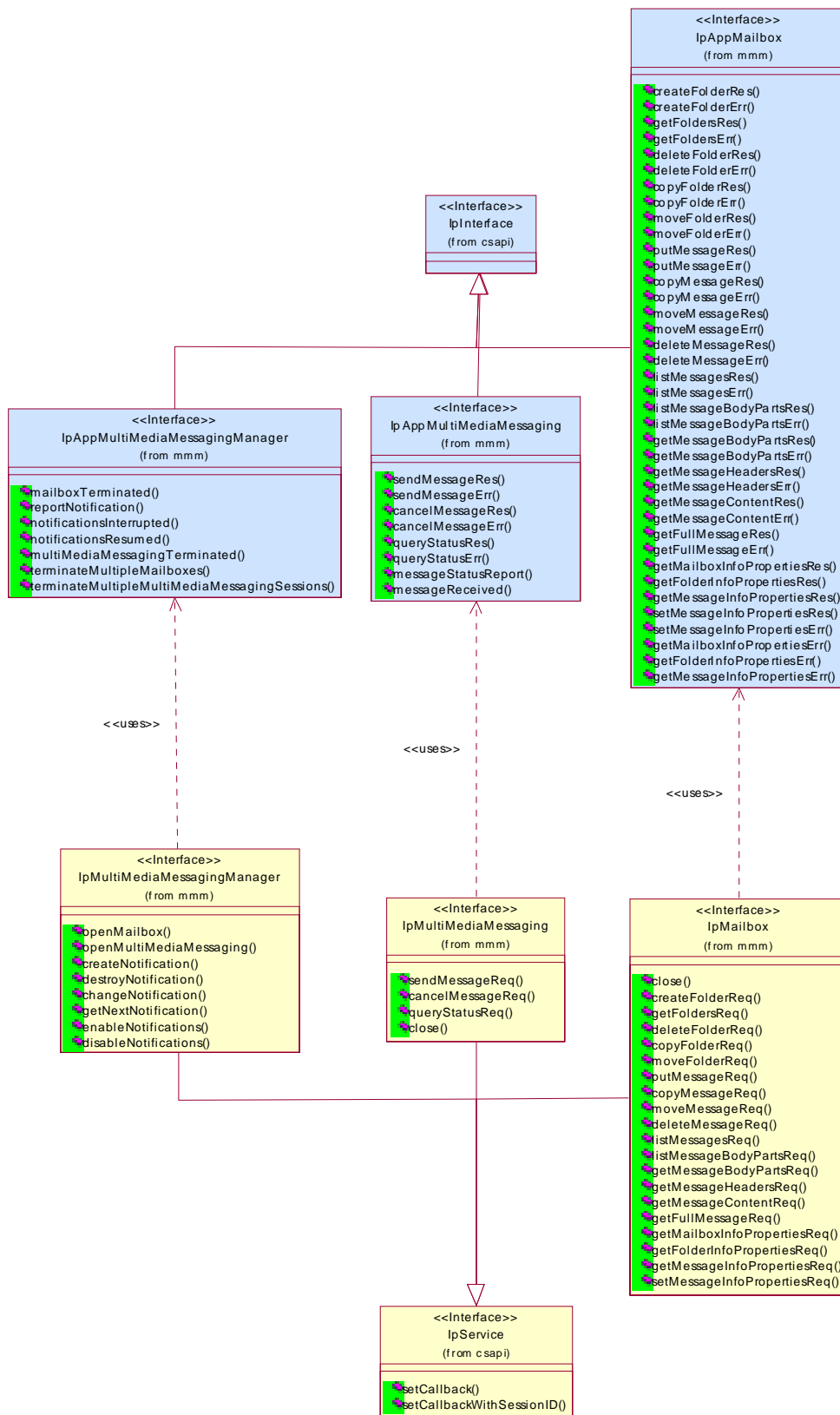


Figure 1: Messaging Interfaces Overview



## 7 The Service Interface Specifications

### 7.1 Interface Specification Format

This clause defines the interfaces, methods and parameters that form a part of the API specification. The Unified Modelling Language (UML) is used to specify the interface classes. The general format of an interface specification is described below.

#### 7.1.1 Interface Class

This shows a UML interface class description of the methods supported by that interface, and the relevant parameters and types. The Service and Framework interfaces for enterprise-based client applications are denoted by classes with name `Ip<name>`. The callback interfaces to the applications are denoted by classes with name `IpApp<name>`. For the interfaces between a Service and the Framework, the Service interfaces are typically denoted by classes with name `IpSvc<name>`, while the Framework interfaces are denoted by classes with name `IpFw<name>`.

#### 7.1.2 Method descriptions

Each method (API method 'call') is described. Both synchronous and asynchronous methods are used in the API. Asynchronous methods are identified by a 'Req' suffix for a method request, and, if applicable, are served by asynchronous methods identified by either a 'Res' or 'Err' suffix for method results and errors, respectively. To handle responses and reports, the application or service developer must implement the relevant `IpApp<name>` or `IpSvc<name>` interfaces to provide the callback mechanism.

#### 7.1.3 Parameter descriptions

Each method parameter and its possible values are described. Parameters described as 'in' represent those that must have a value when the method is called. Those described as 'out' are those that contain the return result of the method when the method returns.

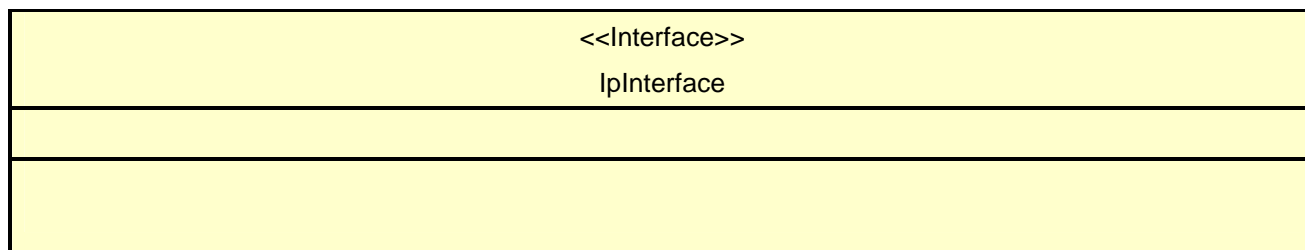
#### 7.1.4 State Model

If relevant, a state model is shown to illustrate the states of the objects that implement the described interface.

### 7.2 Base Interface

#### 7.2.1 Interface Class IpInterface

All application, framework and service interfaces inherit from the following interface. This API Base Interface does not provide any additional methods.



## 7.3 Service Interfaces

### 7.3.1 Overview

The Service Interfaces provide the interfaces into the capabilities of the underlying network - such as call control, user interaction, messaging, mobility and connectivity management.

The interfaces that are implemented by the services are denoted as 'Service Interface'. The corresponding interfaces that must be implemented by the application (e.g. for API callbacks) are denoted as 'Application Interface'.

## 7.4 Generic Service Interface

### 7.4.1 Interface Class IpService

Inherits from: IpInterface.

All service interfaces inherit from the following interface.

|   |
|---|
| <<Interface>><br>IpService  |
| <pre> setCallback (appInterface : in IpInterfaceRef) : void setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void </pre> |

#### 7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

#### *Parameters*

**appInterface : in IpInterfaceRef**

Specifies a reference to the application interface, which is used for callbacks.

#### *Raises*

**TpCommonExceptions, P\_INVALID\_INTERFACE\_TYPE**

#### 7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs. Multiple invocations of this method on an interface shall result in multiple callback references being specified. The SCS shall use the most recent callback interface provided by the application using this method. In the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

*Parameters***appInterface : in IpInterfaceRef**

Specifies a reference to the application interface, which is used for callbacks.

**sessionID : in TpSessionID**

Specifies the session for which the service can invoke the application's callback interface.

*Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_INTERFACE\_TYPE**

## 8 Multi-Media Messaging Interface Classes

The MultiMedia Messaging SCF (MMM SCF) is used by applications to send, store and receive messages either from within the context of a mailbox paradigm, or outside of it. MMM SCF also supports voice mail and electronic mail as the messaging mechanisms. The messaging service interface can be used by both.

The MMM SCF is represented by the IpMultiMediaMessagingManager, IpMailbox and IpMultiMediaMessaging interfaces to services provided by the network. To handle responses and reports, the developer must implement IpAppMultiMediaMessagingManager to provide the callback mechanism for the MultiMediaMessaging service manager.

The MMM SCF also supports messaging in the context of Instant Messaging (IM), SMS, MMS, GSM USSD etc.,. These contexts, IM in particular, may support communication in either the page mode or the session mode. The reader is encouraged to refer to "The Message Session Relay Protocol" (draft-ietf-simple-message-sessions-15.txt) work being done in the IETF for more details.

A messaging system that is conformant with the mailbox paradigm is assumed to have the following entities:

- Mailboxes. This is the application's main entry point to the messaging system. The framework may or may not need to authenticate an application before it accesses a mailbox.
- Folders. Folders may have sub-folders. The names of these sub-folders are appended to their parents' names with '/' as the delimiter. For instance, if there is a folder called INBOX and a sub-folder in INBOX called 'Personal' and a sub-folder in that folder called 'archive' then the fully qualified names, which are required for all operations, of the three folders are 'INBOX', 'INBOX/Personal', and 'INBOX/Personal/archive'. The names are case sensitive.
- Messages. Messages are stored in folders. Messages consist of a message header and message body.

Non-mailbox paradigm messaging is supported through the IpMultiMediaMessagingManager and IpMultiMediaMessaging interfaces.

### 8.1 Interface Class IpMultiMediaMessagingManager

Inherits from: IpService.

This interface is the 'service manager' interface for the MultiMedia Messaging Service. The service manager interface provides the management functions to the MultiMedia Messaging service. The application programmer can use this interface to open mailbox objects, MultiMedia Messaging objects, and also to enable or disable event notifications on them.

| <<Interface>><br>IpMultiMediaMessagingManager  |
|--|
| openMailbox (mailboxID : in TpString, authenticationInfo : in TpString, appMailbox : in IpAppMailboxRef) : TpMailboxIdentifier<br>openMultiMediaMessaging (defaultDestinationAddressList : in TpTerminatingAddressList, defaultSourceAddress : in TpAddress, appMultiMediaMessaging : in IpAppMultiMediaMessagingRef) : TpMultiMediaMessagingIdentifier<br>createNotification (appMultiMediaMessagingManager : in IpAppMultiMediaMessagingManagerRef, eventCriteria : in TpMessagingEventCriteriaSet) : TpAssignmentID<br>destroyNotification (assignmentID : in TpAssignmentID) : void<br>changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpMessagingEventCriteriaSet) : void<br>getNextNotification (reset : in TpBoolean) : TpMessagingNotificationRequestedSetEntry<br>enableNotifications (appMultiMediaMessagingManager : in IpAppMultiMediaMessagingManagerRef) : TpAssignmentID<br>disableNotifications () : void |

### 8.1.1 Method openMailbox()

This method opens a mailbox for the application. The session ID and reference to the IpMailbox interface for use by the application is returned. Authentication information may be needed to open the mailbox.

The application can open more than one mailbox at the same time. The application is not allowed to open the same mailbox more than once at the same time.

Returns: mailboxIdentifier.

Specifies the reference to the opened mailbox and the session ID.

#### *Parameters*

##### **mailboxID : in TpString**

Specifies the identity of the mailbox. If the mailbox chosen is invalid, the P\_MMM\_INVALID\_MAILBOX exception is thrown.

##### **authenticationInfo : in TpString**

Authentication information needed for the application to open a mailbox in the messaging system, such as a key or password. This is distinct from the authentication performed by the Framework. If the Framework authentication process is considered strong enough for the application to gain access to the mailbox, then this parameter will be an empty string. If the authentication information is not valid, the error code P\_MMM\_INVALID\_AUTHENTICATION\_INFORMATION is returned.

##### **appMailbox : in IpAppMailboxRef**

Specifies the client callback interface to be used with this mailbox session. P\_INVALID\_INTERFACE\_TYPE is thrown if the reference is not an IpAppMailbox.

*Returns***TpMailboxIdentifier***Raises***TpCommonExceptions, P\_MMM\_INVALID\_MAILBOX,  
P\_MMM\_INVALID\_AUTHENTICATION\_INFORMATION, P\_INVALID\_INTERFACE\_TYPE**

## 8.1.2 Method openMultiMediaMessaging()

This method is used to open a MultiMedia Messaging object that can be used for single-shot (page mode), or session mode messaging (e.g. instant-messaging).

The defaultDestinationAddressList is used to identify the default set of users that a message is to be sent to when using this interface. This default set is overridden by the information supplied in the sendMessageReq() method. If no destinationAddressList is present in the sendMessageReq(), then the default list supplied here applies. If no address is provided in the defaultDestinationAddressList, then the instance can be re-used for multiple invocations of sendMessageReq() to different destination addresses, with the destination address specified each time in the sendMessageReq() method.

The defaultSourceAddress is used to identify the default address to be used as the source of any message sent when using this interface. This default address is overridden by the information supplied in the sendMessageReq() method. If no sourceAddress is present in the sendMessageReq(), then the default address supplied here applies. If no address is provided in the defaultSourceAddress here, then the instance can be re-used for multiple invocations of sendMessageReq(), with a sourceAddress specified each time in the sendMessageReq() method.

Returns: multimediaMessagingIdentifier

Specifies the reference to the opened MultiMediaMessaging object and session ID.

*Parameters***defaultDestinationAddressList : in TpTerminatingAddressList**

Used to identify the default set of users that a message is to be sent to when using this interface. This default set is overridden by the information supplied in the sendMessageReq() method. If no destinationAddressList is present in the sendMessageReq(), then the default list supplied here applies. If no address is provided in the defaultDestinationAddressList, then the instance can be re-used for multiple invocations of sendMessageReq() to different destination addresses, with the destination address specified each time in the sendMessageReq() method.

**defaultSourceAddress : in TpAddress**

Used to identify the default address to be used as the source of any message sent when using this interface. This default address is overridden by the information supplied in the sendMessageReq() method. If no sourceAddress is present in the sendMessageReq(), then the default address supplied here applies. If no address is provided in the defaultSourceAddress here, then the instance can be re-used for multiple invocations of sendMessageReq(), with a sourceAddress specified each time in the sendMessageReq() method.

**appMultiMediaMessaging : in IpAppMultiMediaMessagingRef**

If this parameter is set (i.e. not NULL) it specifies a reference to the MultiMedia Messaging application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method. P\_INVALID\_INTERFACE\_TYPE is thrown if the reference is not an IpAppMultiMediaMessaging interface.

*Returns***TpMultiMediaMessagingIdentifier***Raises***TpCommonExceptions, P\_INVALID\_INTERFACE\_TYPE, P\_INVALID\_ADDRESS**

### 8.1.3 Method createNotification()

This method enables the application to indicate that it wishes to receive notifications of messaging related events (e.g. receipt of an incoming message).

If the same application invokes this method multiple times with exactly the same criteria but with different callback references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The gateway shall use the most recent callback interface provided by the application using this method. Therefore in the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Returns: assignmentID.

Specifies the ID assigned by the multimedia messaging manager interface for this newly-enabled event notification.

#### Parameters

**appMultiMediaMessagingManager : in IpAppMultiMediaMessagingManagerRef**

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

P\_INVALID\_INTERFACE\_TYPE is thrown if the reference is not an IpAppMultiMediaMessagingManager.

**eventCriteria : in TpMessagingEventCriteriaSet**

Specifies the event specific criteria used by the application to define the event required.

#### Returns

**TpAssignmentID**

#### Raises

**TpCommonExceptions, P\_INVALID\_CRITERIA, P\_INVALID\_INTERFACE\_TYPE**

### 8.1.4 Method destroyNotification()

This method is used by the application to destroy or delete a notification previously set using createNotification. The notification tied to the assignment ID is deleted by this operation.

#### Parameters

**assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the generic communications manager interface when the previous createNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the service manager will return the error code P\_INVALID\_ASSIGNMENT\_ID.

#### Raises

**TpCommonExceptions, P\_INVALID\_ASSIGNMENT\_ID**

### 8.1.5 Method changeNotification()

This method is used by the application to modify or change a notification previously set using createNotification. The assignment ID tied to the original notification is not changed by this operation.

*Parameters***assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the multimedia messaging manager interface when the previous createNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the service manager will return the error code P\_INVALID\_ASSIGNMENT\_ID.

**eventCriteria : in TpMessagingEventCriteriaSet**

Specifies the event specific criteria used by the application to define the event required.

*Raises***TpCommonExceptions, P\_INVALID\_ASSIGNMENT\_ID, P\_INVALID\_CRITERIA**

### 8.1.6 Method getNextNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification. Since a lot of data can potentially be returned (which might cause problem in the middleware), this method must be used in an iterative way. Each method invocation may return part of the total set of notifications if the set is too large to return it at once. The reset parameter permits the application to indicate whether an invocation to getNextNotification is requesting more notifications from the total set of notifications or is requesting that the total set of notifications shall be returned from the beginning.

Returns messagingNotificationRequestedSetEntry: The set of notifications and an indication whether all off the notifications have been obtained or if more notifications are available that have not yet been obtained by the application. If no notifications exist, an empty set is returned and the final indication shall be set to TRUE.

Note that the (maximum) number of items provided to the application is determined by the gateway.

*Parameters***reset : in TpBoolean**

TRUE: indicates that the application intends to obtain the set of notifications starting at the beginning.

FALSE: indicates that the application requests the next set of notifications that have not (yet) been obtained since the last call to this method with this parameter set to TRUE.

The first time this method is invoked, reset shall be set to TRUE. Following the receipt of a final indication in TpNotificationRequestedSetEntry, for the next call to this method reset shall be set to TRUE. P\_TASK\_REFUSED may be thrown if these conditions are not met.

*Returns***TpMessagingNotificationRequestedSetEntry***Raises***TpCommonExceptions**

### 8.1.7 Method enableNotifications()

This method is used to indicate that the application is able to receive notifications which are provisioned from within the network (i.e. these notifications are NOT set using createNotification() but via, for instance, a network management system). If notifications provisioned for this application are created or changed, the application is unaware of this until the notification is reported.

If the same application invokes this method multiple times with different IpAppMultiMediaMessagingManager references, then these shall be treated as additional callback references. Each such notification request shall share the same assignmentID. The gateway shall use the most recent callback interface provided by the application using this method. Therefore in the event that a callback reference fails or is no longer available, the next most recent callback reference available shall be used.

When this method is used, it is still possible to use `createNotification()` for service provider provisioned notifications on the same interface as long as the criteria in the network and provided by `createNotification()` do not overlap. However, it is NOT recommended to use both mechanisms on the same service manager.

The methods `changeNotification()`, `getNotification()`, and `destroyNotification()` do not apply to notifications provisioned in the network and enabled using `enableNotifications()`. These only apply to notifications created using `createNotification()`.

Returns: `assignmentID`.

Specifies the ID assigned by the manager interface for this operation. This ID is contained in any `reportNotification()` that relates to notifications provisioned from within the network.

### *Parameters*

**appMultiMediaMessagingManager : in IpAppMultiMediaMessagingManagerRef**

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the `setCallback()` method.

`P_INVALID_INTERFACE_TYPE` is thrown if the reference is not an `IpAppMultiMediaMessagingManager`.

### *Returns*

**TpAssignmentID**

### *Raises*

**TpCommonExceptions, P\_INVALID\_INTERFACE\_TYPE**

## 8.1.8 Method `disableNotifications()`

This method is used to indicate that the application is not able to receive notifications for which the provisioning has been done from within the network. (i.e. these notifications that are NOT set using `createNotification()` but via, for instance, a network management system). After this method is called, no such notifications are reported anymore.

### *Parameters*

No Parameters were identified for this method.

### *Raises*

**TpCommonExceptions**

## 8.2 Interface Class `IpAppMultiMediaMessagingManager`

Inherits from: `IpInterface`.

The client application developer implements the multimedia messaging manager application interface to handle mailbox termination, mailbox fault and messaging notifications.



| <<Interface>><br>IpAppMultiMediaMessagingManager   |
|--|
| mailboxTerminated (mailboxIdentifier : in TpMailboxIdentifier) : void<br>reportNotification (assignmentID : in TpAssignmentID, eventInfo : in TpMessagingEventInfoSet) :<br>IpAppMultiMediaMessagingRef<br>notificationsInterrupted () : void<br>notificationsResumed () : void<br>multiMediaMessagingTerminated (multimediaMessagingIdentifier : in TpMultiMediaMessagingIdentifier) :<br>void<br>terminateMultipleMailboxes (mailboxSet : in TpMailboxIdentifierSet) : void<br>terminateMultipleMultiMediaMessagingSessions (multiMediaMessagingSet : in<br>TpMultiMediaMessagingIdentifierSet) : void |

### 8.2.1 Method mailboxTerminated()

This method indicates to the application that the mailbox has terminated or closed abnormally. No further communication will be possible between the mailbox and application.

#### *Parameters*

**mailboxIdentifier : in TpMailboxIdentifier**

Specifies the interface and session ID of the mailbox that has terminated.

### 8.2.2 Method reportNotification()

This method notifies the application of the arrival of messaging-related events.

Returns appMultiMediaMessaging: Specifies a reference to an IpAppMultiMediaMessaging interface, if the application has requested the creation of an IpMultiMediaMessaging session upon receipt of a new message arrived event outside the context of a mailbox, and the event reported is P\_EVENT\_MSG\_NEW\_MESSAGE\_ARRIVED. In all other cases this parameter will be null.

#### *Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment id which was returned by the createNotification() or enableNotifications() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

**eventInfo : in TpMessagingEventInfoSet**

Specifies data associated with the events.

#### *Returns*

**IpAppMultiMediaMessagingRef**

### 8.2.3 Method notificationsInterrupted()

This method indicates to the application that all event notifications have been temporarily interrupted (for example, due to faults detected in communication with underlying messaging system).

Note that more permanent failures are reported via the Framework (integrity management).

#### *Parameters*

No Parameters were identified for this method.

### 8.2.4 Method notificationsResumed()

This method indicates to the application that event notifications will again be possible.

#### *Parameters*

No Parameters were identified for this method.

### 8.2.5 Method multiMediaMessagingTerminated()

This method indicates to the application that a MultiMediamessaging session has terminated or closed abnormally. No further communication will be possible between the session and application.

#### *Parameters*

**multimediaMessagingIdentifier : in TpMultiMediaMessagingIdentifier**

Specifies the interface and session ID of the MultiMediaMessaging session that has terminated.

### 8.2.6 Method terminateMultipleMailboxes()

The service may invoke this method on the IpAppMultiMediaMessagingManager interface to indicate that a number of mailboxes have terminated or closed abnormally. No further communication will be possible between the application and the mailbox. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of mailboxes have failed. The service shall provide a set of TpMailboxIdentifiers, indicating to the application the interface references and sessionsIDs of the mailboxes that have aborted. In the case that the service invokes this method and provides an empty set of TpMailboxIdentifiers, this shall be used to indicate that all mailboxes previously active on the IpMultiMediaMessagingManager interface have been aborted.

#### *Parameters*

**mailboxSet : in TpMailboxIdentifierSet**

Specifies the set of interfaces and sessionIDs of the mailboxes that have aborted or terminated abnormally. The empty set shall be used to indicate that all mailboxes have aborted.

### 8.2.7 Method terminateMultipleMultiMediaMessagingSessions()

The service may invoke this method on the IpAppMultiMediaMessagingManager interface to indicate that a number of ongoing multi media messaging sessions have aborted or terminated abnormally. No further communication will be possible between the application and the multi media messaging sessions. This may be used for example in the event of service failure and recovery in order to instruct the application that a number of sessions have failed. The service shall provide a set of TpMultiMediaMessagingIdentifiers, indicating to the application the interface references and sessionsIDs of the multi media messaging sessions that have aborted. In the case that the service invokes this method and provides an empty set of TpMultiMediaMessagingIdentifiers, this shall be used to indicate that all multi media messaging sessions previously active on the IpMultiMediaMessaginManager interface have been aborted.

#### *Parameters*

**multiMediaMessagingSet : in TpMultiMediaMessagingIdentifierSet**

Specifies the set of interfaces and sessionIDs of the multi media messaging sessions that have aborted or terminated abnormally. The empty set shall be used to indicate that all multi media messaging sessions have aborted.

## 8.3 Interface Class IpMailbox

Inherits from: IpService.

This interface supports methods which enable the application to access and manipulate the contents of the mailbox. In order to send messages from the mailbox, the putMessageReq() method is used to put messages in a specified folder, which will automatically result in their being sent.

| <<Interface>><br>IpMailbox  |
|---|
| close (mailboxSessionID : in TpSessionID) : void<br>getMessageInfoPropertiesReq (mailboxSessionID : in TpSessionID, messageID : in TpString) : TpAssignmentID<br>setMessageInfoPropertiesReq (mailboxSessionID : in TpSessionID, messageID : in TpString, properties : in TpMessageInfoPropertySet) : TpAssignmentID<br>createFolderReq (mailboxSessionID : in TpSessionID, folderID : in TpString) : TpAssignmentID<br>getFoldersReq (mailboxSessionID : in TpSessionID, folderID : in TpString) : TpAssignmentID<br>deleteFolderReq (mailboxSessionID : in TpSessionID, folderID : in TpString) : TpAssignmentID<br>copyFolderReq (mailboxSessionID : in TpSessionID, sourceFolderID : in TpString, destinationFolderID : in TpString) : TpAssignmentID<br>moveFolderReq (mailboxSessionID : in TpSessionID, sourceFolderID : in TpString, destinationFolderID : in TpString) : TpAssignmentID<br>putMessageReq (mailboxSessionID : in TpSessionID, folderID : in TpString, message : in TpOctetSet) : TpAssignmentID<br>copyMessageReq (mailboxSessionID : in TpSessionID, fromFolderID : in TpString, toFolderID : in TpString, messageID : in TpString) : TpAssignmentID<br>moveMessageReq (mailboxSessionID : in TpSessionID, fromFolderID : in TpString, toFolderID : in TpString, messageID : in TpString) : TpAssignmentID<br>deleteMessageReq (mailboxSessionID : in TpSessionID, fromFolderID : in TpString, messageID : in TpString) : TpAssignmentID<br>listMessagesReq (mailboxSessionID : in TpSessionID, folderID : in TpString, criteria : in TpListMessagesCriteria, reset : in TpBoolean) : TpAssignmentID<br>listMessageBodyPartsReq (mailboxSessionID : in TpSessionID, folderID : in TpString, messageID : in TpString, maxNestingLevel : in TpInt32) : TpAssignmentID<br>getMessageBodyPartsReq (mailboxSessionID : in TpSessionID, folderID : in TpString, messageID : in TpString, partIDs : in TpStringList) : TpAssignmentID<br>getMessageHeadersReq (mailboxSessionID : in TpSessionID, folderID : in TpString, messageID : in TpString) : TpAssignmentID<br>getMessageContentReq (mailboxSessionID : in TpSessionID, folderID : in TpString, messageID : in TpString) : TpAssignmentID<br>getFullMessageReq (mailboxSessionID : in TpSessionID, folderID : in TpString, messageID : in TpString) : TpAssignmentID<br>getMailboxInfoPropertiesReq (mailboxSessionID : in TpSessionID) : TpAssignmentID<br>getFolderInfoPropertiesReq (mailboxSessionID : in TpSessionID, folderID : in TpString) : TpAssignmentID |

### 8.3.1 Method close()

This method closes the mailbox. After closing, the interfaces to the mailbox and any associated folders are automatically de-assigned and are no longer valid.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

The session ID of the open mailbox previously opened by openMailbox. From now on, the session ID is no longer valid. If by coincidence an identical session ID is returned by a subsequent openMailbox, the session ID will be associated with the new session and has nothing to do with the closed session. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

#### *Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID**

### 8.3.2 Method getMessageInfoPropertiesReq()

This asynchronous method requests the values of properties of a message. The response is provided in getMessageInfoPropertiesRes().

Returns: requestID:

A reference to the request for later use by the application.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**messageID : in TpString**

Identifies the message for which the properties are to be retrieved.

#### *Returns*

**TpAssignmentID**

#### *Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.3 Method setMessageInfoPropertiesReq()

This asynchronous method requests to set the properties of a message. In cases where more than one property was requested to be set, and where not all properties were successfully set, a setMessageInfoPropertiesRes() will be returned indicating those properties successfully set, and a setMessageInfoPropertiesErr() method will be returned as well, indicating those properties which were not set.

Returns: requestID:

A reference to the request for later use by the application.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**messageID : in TpString**

Identifies the message for which the properties are to be set.

**properties : in TpMessageInfoPropertySet**

The message properties (names and values) to be set.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_MESSAGE\_ID, P\_MMM\_INVALID\_PROPERTY**

### 8.3.4 Method createFolderReq()

This method creates a new folder in the mailbox opened by that Mailbox Session. The name of the new folder may be passed in. If the operation fails, a createFolderErr() is invoked by the SCF on the IpAppMailbox interface.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

This is the proposed fully qualified name of the new folder to be created.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID**

### 8.3.5 Method getFoldersReq()

This method requests a list of the (sub)folder names in the mailbox or a folder. The list is provided as a string list, returned in getFoldersRes(). Only the names of the 1st (top) level of folders is returned.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

This contains the fully qualified name of the folder for which the names of the top level of sub-folders are to be returned. If an emptystring is provided, the getFoldersReq requests the names of the top level of folders in the Mailbox.

*Returns* **TpAssignmentID** *Raises* **TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID** 

### 8.3.6 Method deleteFolderReq()

This method requests to remove a folder from the currently open mailbox session. The operation is carried out even for non-empty folders (i.e. folders with subtended folders or email messages within them).

Returns: requestID:

A reference to the request for later use by the application.

*Parameters* **mailboxSessionID : in TpSessionID** 

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

 **folderID : in TpString** 

This contains the fully qualified name of the folder to be deleted.

*Returns* **TpAssignmentID** *Raises* **TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID** 

### 8.3.7 Method copyFolderReq()

This method copies a folder from a mailbox session context, including all its contents into a new folder with the provided name but in the same mailbox context.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters* **mailboxSessionID : in TpSessionID** 

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

 **sourceFolderID : in TpString** 

The fully qualified name of the folder to be copied.

 **destinationFolderID : in TpString** 

The fully qualified name of the folder to be created, which will contain a copy of the source folder.

*Returns* **TpAssignmentID**

*Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID****8.3.8 Method moveFolderReq()**

This method moves a folder from a mailbox session context, including all its contents into a new folder with the provided name but in the same mailbox context.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**sourceFolderID : in TpString**

The fully qualified name of the folder to be moved.

**destinationFolderID : in TpString**

The fully qualified name of the folder to be created, which will contain a copy of the source folder.

*Returns***TpAssignmentID***Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID****8.3.9 Method putMessageReq()**

This method requests to put a message into an open mailbox folder. The message and the headers are transferred to the Messaging service. The message will be taken as is. No checking is done on the message. Furthermore, the message is assumed to be a simple message, that is, with no attachments. If the application knows the messaging system and understands the format to send attachments, it can do so. The service will not flag any inconsistencies if the formatting of the message is not correct.

In order to send messages from the mailbox, the application can use putMessageReq to place messages in a specified folder, from which they will be sent. The folder to use is indicated by the service property P\_PUT\_MESSAGE\_FOLDER\_TO\_SEND.

The messageID associated with the message is returned in putMessageRes().

Returns: requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

This is the fully qualified name of the folder in which the message should be placed.

**message : in TpOctetSet**

The message to put into the mailbox.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_MAX\_MESSAGE\_SIZE\_EXCEEDED**

### 8.3.10 Method copyMessageReq()

This method requests to copy a message with the specified ID from its current folder into the folder specified by the toFolderID. The message will be copied as is. This effects a deep copy, not just a cloning of pointers to the message.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**fromFolderID : in TpString**

This specifies the name of the folder from which the message is to be copied.

**toFolderID : in TpString**

This specifies the name of the folder into which the message is to be copied.

**messageID : in TpString**

The ID of the message to be copied into the specified folder.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.11 Method moveMessageReq()

This method requests to move a message with the specified ID from its current position into the folder specified by the toFolderID. The message will be moved as is.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.



**fromFolderID : in TpString**

This specifies the name of the folder from which the message is to be moved.

**toFolderID : in TpString**

This specifies the name of the folder into which the message is to be moved.

**messageID : in TpString**

The ID of the message to be moved into the specified folder.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.12 Method deleteMessageReq()

This method requests to delete the message with the specified ID from its current position in the folder. If the messaging system supports a file or folder structure with a "Trash" folder, the message may be put into that folder. In other systems, the message may be permanently deleted. If this method is invoked on a message in the trash folder, it is permanently deleted.

Returns: requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**fromFolderID : in TpString**

This specifies the name of the folder from which the message is to be deleted.

**messageID : in TpString**

The ID of the message to be deleted.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.13 Method listMessagesReq()

Request a list of messages in a mailbox folder, the list can be narrowed down by providing listing criteria. The maximum size of the list returned for a request is depending on the System configuration. In that case subsequent requests can be used to obtain the complete list, see also the description of the reset parameter.

Returns requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Specifies the identity of the folder for which a list of messages is requested. The folderID parameter is only relevant if the reset parameter is set to TRUE. If the reset parameter is set to FALSE it is ignored.

**criteria : in TpListMessagesCriteria**

Specifies the criteria that items to be listed need to conform to. The criteria parameter is only relevant if the reset parameter is set to TRUE. If the reset parameter is FALSE it is ignored.

**reset : in TpBoolean**

TRUE: Indicates that the application is intended to obtain the list of messages starting from the beginning.

FALSE: Indicates that the application requests the next part of the list that have not (yet) been obtained since the last call to this method with this parameter set to TRUE.

The first time this method is invoked, reset shall be set to TRUE. Following the receipt of a final indication in the ListMessagesRes(), for the next call to this method reset shall be set to TRUE. P\_TASK\_REFUSED may be thrown if these conditions are not met.

*Returns***TpAssignmentID***Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_INVALID\_CRITERIA**

### 8.3.14 Method listMessageBodyPartsReq()

Request a list of body parts that are contained in a message. This is especially useful with MIME multipart messages. The Application shall indicate up to what nesting level it wants the structure presented. When an Application wants to download only a specific attachment with the GetBodyPartReq() method this method needs to be invoked first in order to find out the partID of the attachment to be retrieved.

Returns requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Specifies the identity of the folder in which the targeted message is contained.

**messageID : in TpString**

Identifies the exact message on which to perform this operation.

**maxNestingLevel : in TpInt32**

Parts of a multipart message can be multipart structures themselves. The Application can indicate what is the maximum nesting level it wants the structure to be reported. A nesting level of 0 means that only the message content itself will be reported. A nesting level of 1 means that the parts of a multipart type on level 0 will be reported as well. In general a maxNestingLevel of nl=n means that the parts of a multipart type on level nl-1 will be reported, for every nl in the range 0..n.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.15 Method getMessageBodyPartsReq()

Request for retrieval of one or more parts of a multipart message. The targeted message is identified by its messageID and the folderID to know where in the mailbox the message is stored. The messageID is typically obtained either by having received a notification of new message arrival or from the result of a list-message request. The partIDs that identify which parts of the message shall be retrieved is obtained by listing the messages body parts first.

Returns requestID:

A reference to the request for later use by the application.

*Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Specifies the identity of the folder in which the targeted message is contained.

**messageID : in TpString**

Identifies the exact message on which to perform this operation.

**partIDs : in TpStringList**

Identifies the parts of the message to retrieve. If an emptystring is provided, all parts are to be retrieved.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID, P\_MMM\_INVALID\_PART\_ID**

### 8.3.16 Method getMessageHeadersReq()

Request the headers of a message. The targeted message is identified by its messageID and the folderID to know where in the mailbox the message is stored. The messageID is typically obtained either by having received a notification of new message arrival or from the result of a list-message request.

Returns requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Specifies the identity of the folder in which the targeted message is contained.

**messageID : in TpString**

Identifies the exact message on which to perform this operation.

*Returns***TpAssignmentID***Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.17 Method getMessageContentReq()

Request the entire body of a message. The targeted message is identified by its messageID and the folderID to know where in the mailbox the message is stored. The messageID is obtained either by having received a notification of new message arrival or from the result of a list-message request.

Returns requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Specifies the identity of the folder in which the targeted message is contained.

**messageID : in TpString**

Identifies the exact message on which to perform this operation.

*Returns***TpAssignmentID***Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.18 Method getFullMessageReq()

Request the entire message, including headers and body. The targeted message is identified by its messageID and the folderID to know where in the mailbox the message is stored. The messageID is typically obtained either by having received a notification of new message arrival or from the result of a list-message request.

Returns requestID:

A reference to the request for later use by the application.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Specifies the identity of the folder in which the targeted message is contained.

**messageID : in TpString**

Identifies the exact message on which to perform this operation.

#### *Returns*

**TpAssignmentID**

#### *Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID, P\_MMM\_INVALID\_MESSAGE\_ID**

### 8.3.19 Method getMailboxInfoPropertiesReq()

This asynchronous method requests the values of properties of a mailbox. The response is provided in getMailboxInfoPropertiesRes().

Returns: requestID:

A reference to the request for later use by the application.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

#### *Returns*

**TpAssignmentID**

#### *Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID**

### 8.3.20 Method getFolderInfoPropertiesReq()

This asynchronous method requests the values of properties of a folder. The response is provided in getFolderInfoPropertiesRes().

Returns: requestID:

A reference to the request for later use by the application.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**folderID : in TpString**

Identifies the folder for which the properties are to be retrieved.

*Returns***TpAssignmentID***Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_MMM\_INVALID\_FOLDER\_ID**

## 8.4 Interface Class IpAppMailbox

Inherits from: IpInterface.

| <<Interface>><br>IpAppMailbox  |
|--|
| <pre> getMessageInfoPropertiesRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID,   messageID : in TpString, returnedProperties : in TpMessageInfoPropertySet) : void setMessageInfoPropertiesRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID,   messageID : in TpString, propertiesUpdated : in TpMessageInfoPropertySet) : void setMessageInfoPropertiesErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID,   messageID : in TpString, propertiesNotUpdated : in TpMessageInfoPropertyErrorSet) : void getMailboxInfoPropertiesErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in   TpMessagingError, errorDetails : in TpString) : void getFolderInfoPropertiesErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in   TpMessagingError, errorDetails : in TpString) : void getMessageInfoPropertiesErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in   TpMessagingError, errorDetails : in TpString) : void createFolderRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, folderID : in   TpString) : void createFolderErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in   TpMessagingError, errorDetails : in TpString) : void getFoldersRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, folderID : in TpString,   folderNames : in TpStringList) : void getFoldersErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, folderID : in TpString,   error : in TpMessagingError, errorDetails : in TpString) : void deleteFolderRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID) : void deleteFolderErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in   TpMessagingError, errorDetails : in TpString) : void copyFolderRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID) : void copyFolderErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in   TpMessagingError, errorDetails : in TpString) : void </pre> |

```

moveFolderRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID) : void
moveFolderErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
putMessageRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, messageID : in
  TpString) : void
putMessageErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
copyMessageRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID) : void
copyMessageErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
moveMessageRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID) : void
moveMessageErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
deleteMessageRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID) : void
deleteMessageErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
listMessagesRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, messageList : in
  TpMessageDescriptionList, mailboxStatusInfo : in TpMailboxFolderStatusInformation, final : in
  TpBoolean) : void
listMessagesErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
listMessageBodyPartsRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, partsList :
  in TpBodyPartDescriptionList) : void
listMessageBodyPartsErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
getMessageBodyPartsRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, bodyParts
  : in TpBodyPartList) : void
getMessageBodyPartsErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
getMessageHeadersRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, headers : in
  TpMessageHeaderFieldSet) : void
getMessageHeadersErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
getMessageContentRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, contentType :
  in TpString, contentTransferEncoding : in TpString, content : in TpOctetSet) : void
getMessageContentErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
getFullMessageRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, message : in
  TpOctetSet) : void
getFullMessageErr (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, error : in
  TpMessagingError, errorDetails : in TpString) : void
getMailboxInfoPropertiesRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID,
  returnedProperties : in TpMailboxInfoPropertySet) : void
getFolderInfoPropertiesRes (mailboxSessionID : in TpSessionID, requestID : in TpAssignmentID, folderID :
  in TpString, returnedProperties : in TpFolderInfoPropertySet) : void

```

### 8.4.1 Method getMessageInfoPropertiesRes()

This method returns the properties of a message, requested by getMessageInfoPropertiesReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageInfoPropertiesReq() that was previously invoked by the client application.

**messageID : in TpString**

Identifies the message for which the properties are being returned.

**returnedProperties : in TpMessageInfoPropertySet**

The message properties (names and values).

### 8.4.2 Method setMessageInfoPropertiesRes()

This method returns the list of properties updated following a setMessageInfoPropertiesReq() invoked by the application. In cases where more than one property was requested to be set, and where not all properties were successfully set, a setMessageInfoPropertiesRes() will be returned indicating those properties successfully set, and a setMessageInfoPropertiesErr() method will be returned as well, indicating those properties which were not set.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding setMessageInfoPropertiesReq() that was previously invoked by the client application.

**messageID : in TpString**

Identifies the message for which the properties were set.

**propertiesUpdated : in TpMessageInfoPropertySet**

This identifies the properties whose values were successfully updated as a result of invoking setMessageInfoPropertiesReq().

### 8.4.3 Method setMessageInfoPropertiesErr()

This method returns the list of properties which were not updated, and a reason why not, following a setMessageInfoPropertiesReq() invoked by the application. In cases where more than one property was requested to be set, and where not all properties were successfully set, a setMessageInfoPropertiesReq() will be returned indicating those properties successfully set, and a setMessageInfoPropertiesErr() method will be returned as well, indicating those properties which were not set.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.



**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding setMessageInfoPropertiesReq() that was previously invoked by the client application.

**messageID : in TpString**

Identifies the message for which the properties were requested to be set.

**propertiesNotUpdated : in TpMessageInfoPropertyErrorSet**

This identifies the names of the properties whose values were not updated as a result of invoking setMessageInfoPropertiesReq(), and the reason why each property was not updated.

#### 8.4.4 Method getMailboxInfoPropertiesErr()

This method indicates that the getMailboxInfoPropertiesReq() was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMailboxInfoPropertiesReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

#### 8.4.5 Method getFolderInfoPropertiesErr()

This method indicates that the getFolderInfoPropertiesReq() was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getFolderInfoPropertiesReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

#### 8.4.6 Method getMessageInfoPropertiesErr()

This method indicates that the getMessageInfoPropertiesReq() was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageInfoPropertiesReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.7 Method createFolderRes()

This method indicates the successful creation of a folder, requested by createFolderReq().

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding createFolderReq() that was previously invoked by the client application.

**folderID : in TpString**

This is the name of the new folder which has been created.

### 8.4.8 Method createFolderErr()

This method indicates that the attempt to create a folder, requested by createFolderReq(), was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding createFolderReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.9 Method getFoldersRes()

This method returns the list of the (sub)folder names in the mailbox or a folder. Only the names of the 1st (top) level of folders is returned.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getFoldersReq() that was previously invoked by the client application.

**folderID : in TpString**

This contains the name of the folder for which the names of the top level of sub-folders are returned. If an emptystring is provided, the list contains the names of the top level of folders in the Mailbox.

**folderNames : in TpStringList**

Contains the list of names of folders contained within the mailbox or folder.

### 8.4.10 Method getFoldersErr()

This method indicates that the request for a list of folder names, requested by getFoldersReq(), has failed.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getFoldersReq() that was previously invoked by the client application.

**folderID : in TpString**

This contains the name of the folder for which the names of the top level of sub-folders are requested. If an emptystring is provided, the request was for the names of the top level of folders in the Mailbox.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.11 Method deleteFolderRes()

This method indicates the successful deletion of a folder, requested by deleteFolderReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding deleteFolderReq() that was previously invoked by the client application.

### 8.4.12 Method deleteFolderErr()

This method indicates that the attempt to delete a folder, requested by deleteFolderReq(), was unsuccessful.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding deleteFolderReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.13 Method copyFolderRes()

This method indicates the successful copying of a folder, requested by copyFolderReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding copyFolderReq() that was previously invoked by the client application.

### 8.4.14 Method copyFolderErr()

This method indicates that the attempt to copy a folder, requested by copyFolderReq(), was unsuccessful.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding copyFolderReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.15 Method moveFolderRes()

This method indicates the successful move of a folder, requested by moveFolderReq().

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding moveFolderReq() that was previously invoked by the client application.

### 8.4.16 Method moveFolderErr()

This method indicates that the attempt to move a folder, requested by moveFolderReq(), was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding moveFolderReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.17 Method putMessageRes()

This method indicates the successful placing of a message in a folder, requested by putMessageReq().

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding putMessageReq() that was previously invoked by the client application.

**messageID : in TpString**

The new ID of the message which has been placed in the folder as requested.

### 8.4.18 Method putMessageErr()

This method indicates that the attempt to put a message in a folder, requested by putMessageReq(), was unsuccessful.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding putMessageReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.19 Method copyMessageRes()

This method indicates the successful copying of a message from one folder to another, as requested by copyMessageReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding copyMessageReq() that was previously invoked by the client application.

### 8.4.20 Method copyMessageErr()

This method indicates that the attempt to copy a message from one folder to another, requested by copyMessageReq(), was unsuccessful.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding copyMessageReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.21 Method moveMessageRes()

This method indicates the successful move of a message from one folder to another, as requested by moveMessageReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding moveMessageReq() that was previously invoked by the client application.

### 8.4.22 Method moveMessageErr()

This method indicates that the attempt to move a message from one folder to another, requested by moveMessageReq(), was unsuccessful.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding moveMessageReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.23 Method deleteMessageRes()

This method indicates the successful deletion of a message, as requested by deleteMessageReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding deleteMessageReq() that was previously invoked by the client application.

### 8.4.24 Method deleteMessageErr()

This method indicates that the attempt to delete a message, requested by deleteMessageReq(), was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding deleteMessageReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.25 Method listMessagesRes()

This method delivers the result of a completed list messages request. Whether there are still more messages that can be listed yet will be indicated with the final parameter.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding listMessagesReq() that was previously invoked by the client application.

**messageList : in TpMessageDescriptionList**

A list with each entry giving a short description of the message.

**mailboxStatusInfo : in TpMailboxFolderStatusInformation**

Gives some information about the status of the mailbox regarding the number of messages it holds, how many new messages etc.

**final : in TpBoolean**

Indication whether the returned list is the final part of the complete list (TRUE) or if there are still parts of the list to retrieve (FALSE).

### 8.4.26 Method listMessagesErr()

This method indicates that the list messages request was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding listMessagesReq() that was previously invoked by the client application.



**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.27 Method listMessageBodyPartsRes()

This method delivers the result of a completed list message body parts request.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding listMessageBodyPartsReq() that was previously invoked by the client application.

**partsList : in TpBodyPartDescriptionList**

Specifies the structure of the message up to the requested nesting level.

### 8.4.28 Method listMessageBodyPartsErr()

This method indicates that the list message body parts request was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding listMessageBodyPartsReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.29 Method getMessageBodyPartsRes()

This method delivers the result of a completed get message body parts request.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageBodyPartsReq() that was previously invoked by the client application.

**bodyParts : in TpBodyPartList**

Contains the details and content of the requested Body Parts of the message.

### 8.4.30 Method getMessageBodyPartsErr()

This method indicates that the get message body parts request was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageBodyPartsReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.31 Method getMessageHeadersRes()

This method delivers the result of a completed get message headers request.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageHeadersReq() that was previously invoked by the client application.

**headers : in TpMessageHeaderFieldSet**

Carries the headers of the message.

### 8.4.32 Method getMessageHeadersErr()

This method indicates that the get message headers request was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageHeadersReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.33 Method getMessageContentRes()

This method delivers the result of a completed get message content request.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageContentReq() that was previously invoked by the client application.

**contentType : in TpString**

Specifies the content type value according to the RFC 2045 format.

**contentTransferEncoding : in TpString**

Specifies the content transfer encoding value according to the RFC2045 format.

**content : in TpOctetSet**

Contains the body of the message.

### 8.4.34 Method getMessageContentErr()

This method indicates that the get message content request was unsuccessful.

*Parameters***mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMessageContentReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.35 Method getFullMessageRes()

This method delivers the result of a completed get full message request.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getFullMessageReq() that was previously invoked by the client application.

**message : in TpOctetSet**

Contains the entire message (headers and body) in unstructured format.

### 8.4.36 Method getFullMessageErr()

This method indicates that the get full message request was unsuccessful.

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getFullMessageReq() that was previously invoked by the client application.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.4.37 Method getMailboxInfoPropertiesRes()

This method returns the properties of a mailbox, requested by getMailboxInfoPropertiesReq().

#### *Parameters*

**mailboxSessionID : in TpSessionID**

This is the session ID of the open mailbox.

**requestID : in TpAssignmentID**

This specifies the requestID associated with the corresponding getMailboxInfoPropertiesReq() that was previously invoked by the client application.

**returnedProperties : in TpMailboxInfoPropertySet**

The mailbox properties (names and values).

### 8.4.38 Method getFolderInfoPropertiesRes()

This method returns the properties of a folder, requested by getFolderInfoPropertiesReq().

#### Parameters

**mailboxSessionID** : in TpSessionID

This is the session ID of the open mailbox.

**requestID** : in TpAssignmentID

This specifies the requestID associated with the corresponding getFolderInfoPropertiesReq() that was previously invoked by the client application.

**folderID** : in TpString

Identifies the folder for which the properties are being returned.

**returnedProperties** : in TpFolderInfoPropertySet

The folder properties (names and values).

## 8.5 Interface Class IpMultiMediaMessaging

Inherits from: IpService.

This interface supports methods that enable messages to be sent or received when the mailbox paradigm is not in use. Mechanisms such as SMS, MMS, GSM USSD, etc., could be used in this context for either single-shot (page mode), or session mode messaging (e.g. instant-messaging). Default source and destination addresses can be provided by the application when an instance of IpMultiMediaMessaging is created. These addresses are overridden by including source or destination addresses in the sendMessageReq() method. If no default source or destination address is provided when an instance of IpMultiMediaMessaging is created, then the instance can be reused for multiple invocations of sendMessageReq() to different targets or from different sources, with the addresses specified each time in the sendMessageReq() method.

| <<Interface>><br>IpMultiMediaMessaging   |
|--|
| sendMessageReq (sessionID : in TpSessionID, sourceAddress : in TpAddress, destinationAddressList : in TpTerminatingAddressList, deliveryType : in TpMessageDeliveryType, messageTreatment : in TpMessageTreatmentSet, message : in TpOctetSet, additionalHeaders : in TpMessageHeaderFieldSet) : TpAssignmentID<br><br>cancelMessageReq (sessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void<br><br>queryStatusReq (sessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void<br><br>close (sessionID : in TpSessionID) : void |

### 8.5.1 Method sendMessageReq()

This method requests the underlying network infrastructure to send the message being passed in through the message parameter as one of the data elements, to the set of identified targets specified using the supported addressing schemes from the specification.

As a response to this method invocation, the SCF will respond with either a `sendMessageRes()`, or a `sendMessageErr()`, indicating that the SCF has or has not succeeded to send the message.

If the application requests further reports about the message status, such as successful delivery, read receipt, or wishes to be notified of non-delivery, it must request this explicitly in the `messageTreatment` parameter. These reports are delivered in the `messageStatusReport()` method on `IpAppMultiMediaMessaging`.

Returns: `assignmentID`.

A reference to the request for later use by the application.

### *Parameters*

#### **sessionID : in TpSessionID**

This is the session ID of the open multimedia messaging session. If the session ID is not a valid session ID, the error code `P_INVALID_SESSION_ID` is returned.

#### **sourceAddress : in TpAddress**

The address that is used to represent the sender of the message. For alphanumeric SMS addresses the address plan `P_ADDRESS_PLAN_UNDEFINED` shall be used.

The address provided here overrides the default address provided in the `openMultiMediaMessaging()` method, if one was provided then. If this parameter is empty, then the default address is used.

#### **destinationAddressList : in TpTerminatingAddressList**

A list of addresses of users to whom the message will be sent. A `terminatingAddressList` contains a TO, CC and BCC address list. When the underlying network technology can not distinguish these all addresses can be concatenated.

The address list provided here overrides the default address list provided in the `openMultiMediaMessaging()` method, if one was provided then. If this parameter is empty, then the default address is used.

#### **deliveryType : in TpMessageDeliveryType**

Specifies what delivery method shall be used to deliver the message to the user. If an unsupported delivery type is specified, the exception `P_MMM_INVALID_DELIVERY_TYPE` is returned.

#### **messageTreatment : in TpMessageTreatmentSet**

This parameter contains instructions to the messaging system about how to process and send the message. These instructions can include a request for report of delivery, read receipt, message expiry or non-delivery of the message.

#### **message : in TpOctetSet**

The actual message that needs to be sent.

#### **additionalHeaders : in TpMessageHeaderFieldSet**

This parameter contains additional header information which is intended to be sent as part of the message. This information could have been provided in the raw message, if correctly formulated. Information contained in the additional headers may duplicate information provided in the `sourceAddress` and `destinationAddressList` parameters of the `sendMessageReq()`. In case of conflict, the SCF will take as priority the information provided in the `sourceAddress` and the `destinationAddressList` parameters.

### *Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_ADDRESS,  
 P\_MMM\_INVALID\_DELIVERY\_TYPE, P\_MMM\_MAX\_MESSAGE\_SIZE\_EXCEEDED,  
 P\_MMM\_DELIVERY\_TYPE\_ADDRESS\_TYPE\_MISMATCH,  
 P\_MMM\_DELIVERY\_TYPE\_MESSAGE\_TYPE\_MISMATCH, P\_MMM\_INVALID\_DELIVERY\_TIME,  
 P\_MMM\_INVALID\_VALIDITY\_TIME, P\_MMM\_MAX\_SUBJECT\_SIZE\_EXCEEDED,  
 P\_MMM\_INVALID\_HEADER**

**8.5.2 Method cancelMessageReq()**

This method requests the underlying network infrastructure to cancel an undelivered message previously sent in a `sendMessageReq`.

*Parameters*

**sessionID : in TpSessionID**

This is the session ID of the open multimedia messaging session. If the session ID is not a valid session ID, the error code `P_INVALID_SESSION_ID` is returned.

**assignmentID : in TpAssignmentID**

This specifies the assignmentID associated with the `sendMessageReq` that was previously invoked by the client application on the SCS and which now needs to be cancelled. If an invalid assignment ID is used, a `P_INVALID_ASSIGNMENT_ID` exception is thrown.

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_ASSIGNMENT\_ID,  
 P\_INVALID\_NETWORK\_STATE, P\_MMM\_CANNOT\_CANCEL**

**8.5.3 Method queryStatusReq()**

This method requests the underlying network infrastructure to query the status of messages already sent using the `sendMessageReq()`. A session ID and an assignment ID are used to identify individual messages. Implementations may choose to use the same assignment ID in different sessions to reference different messages, but the specification merely requires that assignment IDs in individual sessions be unique.

*Parameters*

**sessionID : in TpSessionID**

This is the session ID of the open communications session. If the session ID is not a valid session ID, the exception `P_INVALID_SESSION_ID` is returned.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the `sendMessageReq` that was previously invoked by the client application on the SCS and which now needs to be cancelled. If an invalid assignment ID is issued, a `P_INVALID_ASSIGNMENT_ID` exception is thrown.

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_ASSIGNMENT\_ID,  
 P\_MMM\_INFORMATION\_NOT\_AVAILABLE**

**8.5.4 Method close()**

This method requests the SCS to close a previously opened session. Once closed, the application can no longer send new messages, query the status of already sent messages, nor cancel the transmission of pending messages on that session.

*Parameters***sessionID : in TpSessionID**

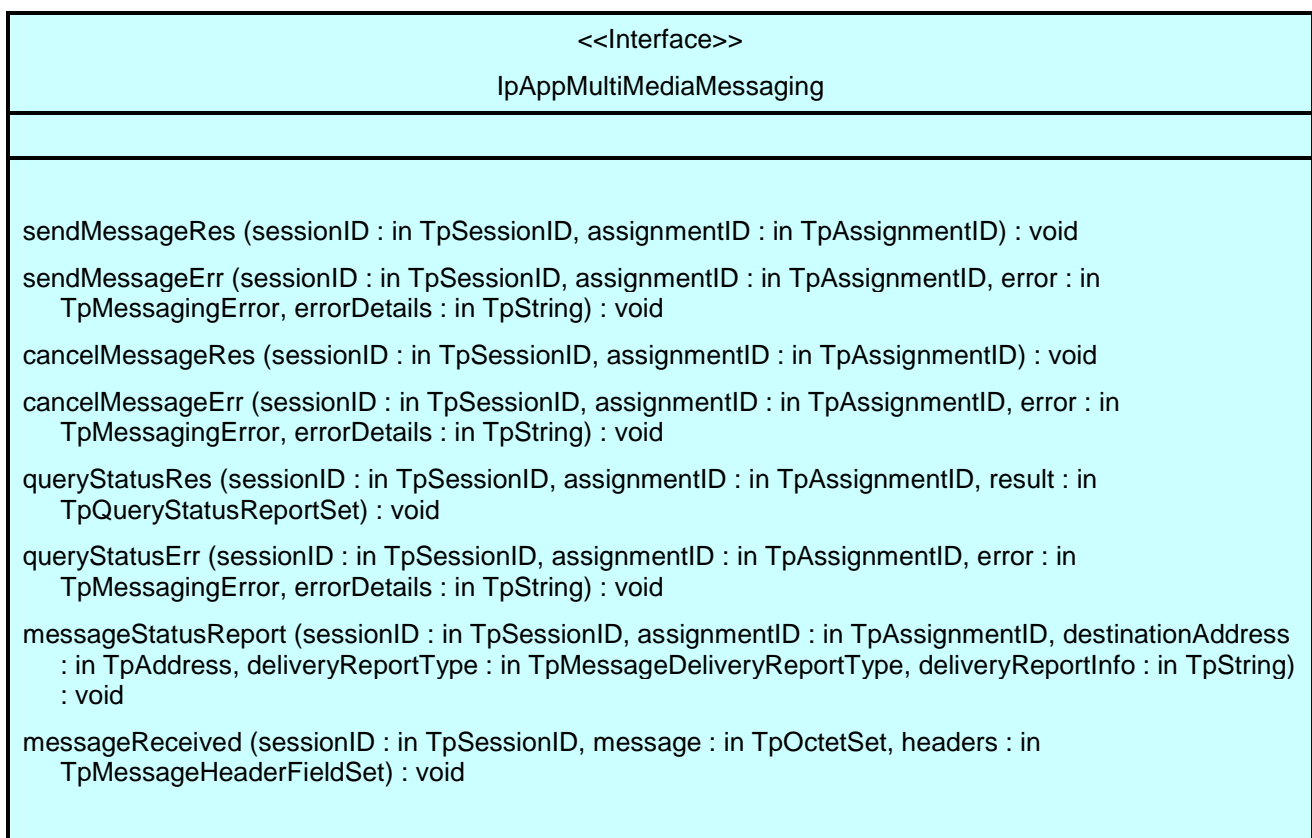
This is the session ID of the open multi media messaging session. If the session ID is not a valid session ID, the error code P\_INVALID\_SESSION\_ID is returned.

*Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID**

## 8.6 Interface Class IpAppMultiMediaMessaging

Inherits from: IpInterface.

This interface provides methods that may be invoked by the SCS on the client application as callbacks to asynchronously inform it of the status of pending requests, etc., for requests issued within the context of non-mailbox messaging systems employed for either single-shot or session-based messaging.



### 8.6.1 Method sendMessageRes()

This asynchronous method informs the application about the completion of a sendMessageReq(). Receipt of this method indicates that the SCF has successfully processed the sendMessageReq() method and successfully sent the message. It does not indicate that the message was delivered or read.

*Parameters***sessionID : in TpSessionID**

This is the session ID of the multimedia messaging session.



**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the sendMessageReq that was previously invoked by the client application on the SCS.

## 8.6.2 Method sendMessageErr()

This asynchronous method indicates that the request to send a message was unsuccessful. The SCF was unable to process the sendMessageReq() or was unable to send the message. Further details are provided in the error parameter.

*Parameters***sessionId : in TpSessionID**

This is the session ID of the multimedia messaging session.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the sendMessageReq that was previously invoked by the client application on the SCS.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

## 8.6.3 Method cancelMessageRes()

This method indicates successful execution of a cancelMessageReq() method by the SCS.

*Parameters***sessionId : in TpSessionID**

This is the session ID of the multimedia messaging session.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the cancelMessageReq that was previously invoked by the client application on the SCS.

## 8.6.4 Method cancelMessageErr()

This asynchronous method indicates that the request to cancel a message was unsuccessful.

*Parameters***sessionId : in TpSessionID**

This is the session ID of the multimedia messaging session.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the cancelMessageReq that was previously invoked by the client application on the SCS.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.6.5 Method queryStatusRes()

This method indicates successful execution of a queryStatusReq() method by the SCS, and provides the status along with the response indication.

*Parameters***sessionID : in TpSessionID**

This is the session ID of the multimedia messaging session.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the queryStatusReq that was previously invoked by the client application on the SCS.

**result : in TpQueryStatusReportSet**

This is a list of each destination address of the message, together with the reported status from that address.

### 8.6.6 Method queryStatusErr()

This method indicates a failure in the execution of the queryStatusReq() method by the SCS.

*Parameters***sessionID : in TpSessionID**

This is the session ID of the multimedia messaging session.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the queryStatusReq that was previously invoked by the client application on the SCS.

**error : in TpMessagingError**

Indicates the error that occurred.

**errorDetails : in TpString**

Provides additional information which may help to locate the source of the error. There is no specified format for this information.

### 8.6.7 Method messageStatusReport()

This method is used to provide the application with any read or delivery etc. reports it has explicitly requested in the sendMessageReq() method. This method may be received more than once for a single invocation of sendMessageReq(), if multiple report types have been requested, or if the message had multiple destination addresses.

*Parameters***sessionID : in TpSessionID**

This is the session ID of the multimedia messaging session.

**assignmentID : in TpAssignmentID**

This specifies the assignment ID associated with the sendMessageReq that was previously invoked by the client application on the SCS.

**destinationAddress : in TpAddress**

Identifies from which destination address of the original message that this report arrives.

**deliveryReportType : in TpMessageDeliveryReportType**

Indicates the type of report carried by this messageStatusReport method. If more than one report type needs to be delivered to the application, the SCF will invoke a separate messageStatusReport for each report type.

**deliveryReportInfo : in TpString**

Additional information which may have been provided by the messaging system in its read, delivery etc. report, and which the SCF passes to the application. The format of this information is not specified.

## 8.6.8 Method messageReceived()

This method is used to inform the application that a message has been received for a remote party within the context of the conversation or session currently active. The full contents of the message are provided. The message may be, but is not necessarily in reply to a message sent by the application using the sendMessageReq().

Messages which are received outside the context of the conversation or session are notified to the application using the IpAppMultiMediaMessagingManager.reportNotification() method.

It is not specified how the SCF is expected to identify that a received message is inside or outside of the context of a currently active conversation or session.

*Parameters***sessionID : in TpSessionID**

This is the session ID of the multimedia messaging session.

**message : in TpOctetSet**

The actual message received. Contains the entire message (headers and body) in unstructured format.

**headers : in TpMessageHeaderFieldSet**

This parameter contains header information which was sent as part of the message. This information could be duplicated in the raw message, depending on its format.

---

## 9 State Transition Diagrams

There are no State Transition Diagrams for the Multi-Media Messaging SCF.

---

## 10 Multi-Media Messaging Service Properties

The following table lists properties relevant for the Multi-Media Messaging API.

| Property                     | Type       | Description  |
|------------------------------|------------|--|
| P_MESSAGE_DELIVERY_TYPE      | STRING_SET | Specifies the message delivery capabilities which the SCS supports. Values are defined by TpMessageDeliveryType. |
| P_PUT_MESSAGE_FOLDER_TO_SEND | STRING_SET | Identifies the folder in the mailbox into which messages should be placed in order to be sent.                   |

The previous table lists properties related to capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

| Property                      | Type                  | Description   |
|-------------------------------|-----------------------|---|
| P_NOTIFICATION_ADDRESS_RANGES | XML_ADDRESS_RANGE_SET | Indicates for which numbers notifications may be set. More than one range may be present. |

## 11 Data Definitions

All data types referenced but not defined in this clause are common data definitions which may be found in ES 203 915-2.

### 11.1 Multi-Media Messaging data definitions

#### 11.1.1 IpMultiMediaMessagingManager

Defines the address of an IpMultiMediaMessagingManager Interface.

#### 11.1.2 IpMultiMediaMessagingManagerRef

Defines a Reference to type IpMultiMediaMessagingManager.

#### 11.1.3 IpAppMultiMediaMessagingManager

Defines the address of an IpAppMultiMediaMessagingManager Interface.

#### 11.1.4 IpAppMultiMediaMessagingManagerRef

Defines a Reference to type IpAppMultiMediaMessagingManager.

#### 11.1.5 IpMailbox

Defines the address of an IpMailbox Interface.

#### 11.1.6 IpMailboxRef

Defines a Reference to type IpMailbox.

#### 11.1.7 IpAppMailbox

Defines the address of an IpAppMailbox Interface.

#### 11.1.8 IpAppMailboxRef

Defines a Reference to type IpAppMailbox.

#### 11.1.9 IpMultiMediaMessaging

Defines the address of an IpMultiMediaMessaging Interface.

#### 11.1.10 IpMultiMediaMessagingRef

Defines a Reference to type IpMultiMediaMessaging.

### 11.1.11 IpAppMultiMediaMessaging

Defines the address of an IpAppMultiMediaMessaging Interface.

### 11.1.12 IpAppMultiMediaMessagingRef

Defines a Reference to type IpAppMultiMediaMessaging.

### 11.1.13 TpBodyPartDescription

Defines a sequence of data elements that specify the properties of a body part.

| Sequence Element Name   | Sequence Element Type | Description   |
|-------------------------|-----------------------|---|
| ContentDescription      | TpString              | The contents of the field shall be interpreted as the RFC 2045 Content-Description field body.  |
| ContentSize             | TpInt32               | This field specifies the length of the body part content in bytes.  |
| ContentType             | TpString              | The contents of the field shall be interpreted as the RFC 2045 Content-Type field body.   |
| ContentTransferEncoding | TpString              | The contents of the field shall be interpreted as the RFC 2045 Content-Transfer-Encoding field body.  |
| ContentID               | TpString              | The contents of the field shall be interpreted as the RFC 2045 Content-ID field body.   |
| ContentDisposition      | TpString              | The contents of the field shall be interpreted as the RFC 2183 Content-Disposition field body.  |
| PartID                  | TpString              | Identifies the body part uniquely within the message. This identifier is created by the System and shall always be the same for the same message, whether a structure was listed with a nesting level of n or n+1 should not make a difference. |
| NestingLevel            | TpInt32               | Specifies how deep the part is nested within the structure.   |

### 11.1.14 TpBodyPartDescriptionList

Defines a numbered list of data elements of TpBodyPartDescription.

### 11.1.15 TpBodyPart

Defines a sequence of data elements that specify a body part.

| Sequence Element Name | Sequence Element Type | Description  |
|-----------------------|-----------------------|--|
| BodyPartHeader        | TpBodyPartDescription | Specifies details about the body part that are needed in order to interpret the content correctly. |
| BodyPartContent       | TpOctetSet            | Contains the content of a body part.   |

### 11.1.16 TpBodyPartList

Defines a numbered list of data elements of TpBodyPart.

### 11.1.17 TpDeliveryTime

Defines the Tagged Choice of Data Elements that specifies when the message shall be delivered.

|  | Tag Element Type   |  |
|--|--------------------|--|
|  | TpDeliveryTimeType |  |

| Tag Element Value    | Choice Element Type | Choice Element Name |
|----------------------|---------------------|---------------------|
| P_MMM_SEND_IMMEDIATE | NULL                | Undefined           |
| P_MMM_DELIVERY_TIME  | TpDateAndTime       | DeliveryTime        |

### 11.1.18 TpDeliveryTimeType

Defines whether a message shall be delivered instantly or at some specified time.

| Name                 | Value | Description   |
|----------------------|-------|---|
| P_MMM_SEND_IMMEDIATE | 0     | The message shall be delivered as soon as possible.         |
| P_MMM_DELIVERY_TIME  | 1     | The message shall be delivered at a certain specified time. |

### 11.1.19 TpFolderInfoProperty

Defines the Tagged Choice of Data Elements that specify the information properties of a folder.

|  | Tag Element Type         |  |
|--|--------------------------|--|
|  | TpFolderInfoPropertyName |  |

| Tag Element Value               | Choice Element Type | Choice Element Name    |
|---------------------------------|---------------------|------------------------|
| P_MMM_FOLDER_DATE_CREATED       | TpDateAndTime       | FolderDateCreated      |
| P_MMM_FOLDER_DATE_CHANGED       | TpDateAndTime       | FolderDateChanged      |
| P_MMM_FOLDER_SIZE               | TpInt32             | FolderSize             |
| P_MMM_FOLDER_NUMBER_OF_MESSAGES | TpInt32             | FolderNumberOfMessages |

### 11.1.20 TpFolderInfoPropertyName

Defines a specific folder information property name.

| Name                            | Value | Description  |
|---------------------------------|-------|--|
| P_MMM_FOLDER_UNDEFINED          | 0     | Undefined  |
| P_MMM_FOLDER_DATE_CREATED       | 1     | Indicates the date created. The application cannot modify this property.                     |
| P_MMM_FOLDER_DATE_CHANGED       | 2     | Indicates the date last changed. The application cannot modify this property.                |
| P_MMM_FOLDER_SIZE               | 3     | Indicates the size of the Folder in Bytes. The application cannot modify this property.      |
| P_MMM_FOLDER_NUMBER_OF_MESSAGES | 4     | Indicates the number of messages in the Folder. The application cannot modify this property. |

### 11.1.21 TpFolderInfoPropertySet

Defines a Numbered Set of Data Elements of TpFolderInfoProperty.

### 11.1.22 TpGenericHeaderField

Specifies the name and value of a header field.

| Sequence Element Name | Sequence Element Type | Description   |
|-----------------------|-----------------------|---|
| FieldName             | TpString              | Contains the field name of an RFC 822 header field. |
| FieldValue            | TpString              | Contains the field body of a RFC 822 header field.  |

### 11.1.23 TpListMessagesCriteria

The list message criteria can be used to narrow down the list of messages reported to the Application by specifying extra criteria that the listed messages need to conform to.

| Sequence Element Name | Sequence Element Type | Description   |
|-----------------------|-----------------------|---|
| OnlyUnreadMessages    | TpBoolean             | When this field is TRUE only unread messages shall be reported. |

### 11.1.24 TpMailboxFolderStatusInformation

Describes the status of a mailbox folder.

| Sequence Element Name | Sequence Element Type | Description   |
|-----------------------|-----------------------|---|
| TotalMessageCount     | TpInt32               | Specifies the total number of messages in the mailbox folder. |

### 11.1.25 TpMailboxIdentifier

Defines the Sequence of Data Elements that identify a mailbox.

| Sequence Element Name | Sequence Element Type |
|-----------------------|-----------------------|
| Mailbox               | IpMailboxRef          |
| SessionID             | TpSessionID           |

### 11.1.26 TpMailboxIdentifierSet

Defines a Numbered Set of Data Elements of TpMailboxIdentifier.

### 11.1.27 TpMailboxInfoProperty

Defines the Tagged Choice of Data Elements that specify the information properties of a mailbox.

| Tag Element Type          |
|---------------------------|
| TpMailboxInfoPropertyName |

| Tag Element Value          | Choice Element Type | Choice Element Name |
|----------------------------|---------------------|---------------------|
| P_MMM_MAILBOX_OWNER        | TpString            | MailboxOwner        |
| P_MMM_MAILBOX_DATE_CREATED | TpDateAndTime       | MailboxDateCreated  |
| P_MMM_MAILBOX_DATE_CHANGED | TpDateAndTime       | MailboxDateChanged  |
| P_MMM_MAILBOX_SIZE         | TpInt32             | MailboxSize         |

### 11.1.28 TpMailboxInfoPropertyName

Defines a specific mailbox information property name.

| Name                       | Value | Description  |
|----------------------------|-------|--|
| P_MMM_MAILBOX_UNDEFINED    | 0     | Undefined.   |
| P_MMM_MAILBOX_OWNER        | 1     | The owner of the mailbox. The application cannot modify this property.                   |
| P_MMM_MAILBOX_DATE_CREATED | 2     | Indicates the date created. The application cannot modify this property.                 |
| P_MMM_MAILBOX_DATE_CHANGED | 3     | Indicates the date last changed. The application cannot modify this property.            |
| P_MMM_MAILBOX_SIZE         | 4     | Indicates the size of the Mailbox in Bytes. The application cannot modify this property. |

### 11.1.29 TpMailboxInfoPropertySet

Defines a Numbered Set of Data Elements of TpMailboxInfoProperty.

### 11.1.30 TpMailboxMessageStatus

Defines the status of messages stored in the Mailbox.

The first 4 status items are applicable to messages which have been received.

The 5<sup>th</sup> status item refers to messages which have been created or put in the Mailbox and not yet sent.

The remaining items are applicable to messages which have been sent from the mailbox, and for which a status is recorded and updated depending on any delivery reports which have been received.

| Name                                   | Value | Description  |
|--|-------|--|
| P_MMM_RECEIVED_MSG_STATUS_READ         | 0     | The message received has been read   |
| P_MMM_RECEIVED_MSG_STATUS_UNREAD       | 1     | The message received has not been read   |
| P_MMM_RECEIVED_MSG_STATUS_FORWARDED    | 2     | The message received has been forwarded  |
| P_MMM_RECEIVED_MSG_STATUS_REPLIED_TO   | 3     | The message received has been replied to   |
| P_MMM_DRAFT_MSG_STATUS_SAVED_OR_UNSENT | 4     | The message has been saved as a draft or is unsent   |
| P_MMM_SENT_MSG_STATUS_SENT             | 5     | The message has been sent. No further information is available, because no delivery report was received, or none was requested |
| P_MMM_SENT_MSG_STATUS_DELIVERED        | 6     | The message sent has been delivered  |
| P_MMM_SENT_MSG_STATUS_READ             | 7     | The message sent has been read   |
| P_MMM_SENT_MSG_STATUS_DELETED_UNREAD   | 8     | The message sent has been deleted without being read   |
| P_MMM_SENT_MSG_STATUS_NOT_DELIVERABLE  | 9     | The message sent was undeliverable   |
| P_MMM_SENT_MSG_STATUS_EXPIRED          | 10    | The message sent has expired before it was delivered   |



### 11.1.31 TpMessageDeliveryType

This data type is identical to a TpString, and is defined as a string of characters that identify the names of the modes of delivery for a message. Other Network operator specific names may also be used, but should be preceded by the string "SP\_". The following values are defined.

| Name             | Description   |
|------------------|---|
| P_MMM_SMS        | The message shall be delivered as a Short Message.        |
| P_MMM_SMS_BINARY | The message shall be delivered as a Binary Short Message. |
| P_MMM_MMS        | The message shall be delivered as a Multimedia Message.   |
| P_MMM_WAP_PUSH   | The message shall be delivered as a WAP-push Message.     |
| P_MMM_EMAIL      | The message shall be delivered as an e-mail Message.      |

### 11.1.32 TpMessageInfoProperty

Defines the Tagged Choice of Data Elements that specify the information properties of a message.

| Tag Element Type          |
|---------------------------|
| TpMessageInfoPropertyName |

| Tag Element Value           | Choice Element Type    | Choice Element Name |
|-----------------------------|------------------------|---------------------|
| P_MMM_MESSAGE_DATE_CREATED  | TpDateAndTime          | MessageDateCreated  |
| P_MMM_MESSAGE_DATE_RECEIVED | TpDateAndTime          | MessageDateReceived |
| P_MMM_MESSAGE_DATE_CHANGED  | TpDateAndTime          | MessageDateChanged  |
| P_MMM_MESSAGE_SIZE          | TpInt32                | MessageSize         |
| P_MMM_MESSAGE_STATUS        | TpMailboxMessageStatus | MessageStatus       |

### 11.1.33 TpMessageInfoPropertyName

Defines a specific message information property name.

| Name                        | Value | Description  |
|-----------------------------|-------|--|
| P_MMM_MESSAGE_UNDEFINED     | 0     | Undefined.   |
| P_MMM_MESSAGE_CREATED       | 1     | Indicates the date created. The application cannot modify this property.                 |
| P_MMM_MESSAGE_DATE_RECEIVED | 2     | Indicates the date received. The application cannot modify this property.                |
| P_MMM_MESSAGE_DATE_CHANGED  | 3     | Indicates the date last changed. The application cannot modify this property.            |
| P_MMM_MESSAGE_SIZE          | 4     | Indicates the size of the message in bytes. The application cannot modify this property. |
| P_MMM_MESSAGE_STATUS        | 5     | The status of the message. This property can be modified by the application.             |

### 11.1.34 TpMessageInfoPropertySet

Defines a Numbered Set of Data Elements of TpMessageInfoProperty.

### 11.1.35 TpMessageHeaderFieldType

Specifies the different types of header fields recognized.

| Name                        | Value | Description  |
|-----------------------------|-------|--|
| P_MESSAGE_DATE_SENT         | 0     | The origination date specifies the date and time at which the creator of the message indicated that the message was complete and ready to enter the mail delivery system.  |
| P_MESSAGE_SENT_FROM         | 1     | Specifies the author(s) of the message, that is, the mailbox address(es) of the person(s) or system(s) responsible for the writing of the message.   |
| P_MESSAGE_SENDER            | 2     | Specifies the mailbox address of the agent responsible for the actual transmission of the message.   |
| P_MESSAGE_REPLY_TO          | 3     | Indicates the mailbox address(es) to which the author of the message suggests that replies be sent.  |
| P_MESSAGE_SENT_TO           | 4     | Specifies the address(es) of the primary recipient(s) of the message.  |
| P_MESSAGE_CC_TO             | 5     | Specifies the addresses of others who are to receive the message, though the content of the message may not be directed at them.   |
| P_MESSAGE_BCC_TO            | 6     | Specifies addresses of recipients of the message whose addresses are not to be revealed to other recipients of the message.  |
| P_MESSAGE_RFC822_MESSAGE_ID | 7     | Specifies a unique message identifier that refers to a particular version of a particular message. This field has the same semantics as the RFC 822 / RFC 2822 "Message-ID:" field.<br>Note that this message ID can not be used on the messaging interface to address this specific message. See TpMessageDescription for more information. |
| P_MESSAGE_IN_REPLY_TO       | 8     | May be used to identify the message (or messages) to which the new message is a reply. The messages are referred by their RFC 822 / RFC 2822 Message-ID.   |
| P_MESSAGE_REFERENCES        | 9     | May be used to identify the message (or messages) with which this message forms a thread of conversation. The messages are referred by their RFC 822 / RFC 2822 Message-ID.  |
| P_MESSAGE_SUBJECT           | 10    | A short string identifying the topic of the message.   |
| P_MESSAGE_COMMENTS          | 11    | This field has the same semantics as the RFC 822 / RFC 2822 "Comments:" field.   |
| P_MESSAGE_KEYWORDS          | 12    | This field has the same semantics as the RFC 822 / RFC 2822 "Keywords:" field.   |
| P_MESSAGE_TRACE_FIELD       | 13    | All trace fields like for example RFC 822 / RFC 2822 "Return-Path:" and "Received:" will be gathered under this flag.  |
| P_MESSAGE_RESENT_FIELD      | 14    | All RFC 822 / RFC 2822 resent fields will be gathered under this flag.   |
| P_MESSAGE_MIME_VERSION      | 15    | Declare the version of the Internet message body format standard in use. This field has the same semantics as the RFC 2045 "MIME-Version:" field.  |
| P_MESSAGE_MIME_CONTENT      | 16    | The Content-Type header field specifies the nature of the data in the body of an entity by giving media type and subtype identifiers, and by providing auxiliary information that may be required for certain media types. This field has the same semantics as the RFC 2045 "Content-Type:" field.  |
| P_MESSAGE_MIME_ENCODING     | 17    | This field's value is a single token specifying the type of encoding. This field has the same semantics as the RFC 2045 "Content-Transfer-Encoding:" field.  |
| P_MESSAGE_MIME_ID           | 18    | When present uniquely identifies a MIME entity. This field has the same semantics as the RFC 2045 "Content-ID:" field.   |
| P_MESSAGE_MIME_DESCRIPTION  | 19    | Specifies some descriptive information about the MIME entity. This field has the same semantics as the RFC 2045 "Content-Description:" field.  |

| Name                           | Value | Description   |
|--------------------------------|-------|---|
| P_MESSAGE_MIME_DISPOSITION     | 20    | Specifies how the MIME entity shall be presented, inline or as an attachment. This field has the same semantics as the RFC 2183 "Content-Disposition:" field. |
| P_MESSAGE_MIME_EXTENSION_FIELD | 21    | Any RFC 822 header field which begins with the string "Content-" and does not match the description of the other MIME fields.                                 |
| P_MESSAGE_EXTENSION_FIELD      | 22    | Any header field that does not match any of the of the other field types in this datatype.  |
| P_MESSAGE_PRIORITY             | 23    | The Priority of the message   |

### 11.1.36 TpMessageHeaderField

Carries the contents of one message header field.

| Tag Element Type         |
|--------------------------|
| TpMessageHeaderFieldType |

| Tag Element Value              | Choice Element Type  | Choice Element Name |
|--------------------------------|----------------------|---------------------|
| P_MESSAGE_DATE_SENT            | TpDateAndTime        | DateSent            |
| P_MESSAGE_SENT_FROM            | TpAddressSet         | From                |
| P_MESSAGE_SENDER               | TpAddress            | Sender              |
| P_MESSAGE_REPLY_TO             | TpAddressSet         | ReplyTo             |
| P_MESSAGE_SENT_TO              | TpAddressSet         | To                  |
| P_MESSAGE_CC_TO                | TpAddressSet         | Cc                  |
| P_MESSAGE_BCC_TO               | TpAddressSet         | Bcc                 |
| P_MESSAGE_RFC822_MESSAGE_ID    | TpString             | RFC822MessageID     |
| P_MESSAGE_IN_REPLY_TO          | TpStringSet          | InReplyTo           |
| P_MESSAGE_REFERENCES           | TpStringSet          | References          |
| P_MESSAGE_SUBJECT              | TpString             | Subject             |
| P_MESSAGE_COMMENTS             | TpString             | Comments            |
| P_MESSAGE_KEYWORDS             | TpStringSet          | Keywords            |
| P_MESSAGE_TRACE_FIELD          | TpGenericHeaderField | TraceField          |
| P_MESSAGE_RESENT_FIELD         | TpGenericHeaderField | ResentField         |
| P_MESSAGE_MIME_VERSION         | TpString             | MimeVersion         |
| P_MESSAGE_MIME_CONTENT         | TpString             | MimeContent         |
| P_MESSAGE_MIME_ENCODING        | TpString             | MimeEncoding        |
| P_MESSAGE_MIME_ID              | TpString             | MimeID              |
| P_MESSAGE_MIME_DESCRIPTION     | TpString             | MimeDescription     |
| P_MESSAGE_MIME_DISPOSITION     | TpString             | MimeDisposition     |
| P_MESSAGE_MIME_EXTENSION_FIELD | TpGenericHeaderField | MimeExtensionField  |
| P_MESSAGE_EXTENSION_FIELD      | TpGenericHeaderField | ExtensionField      |
| P_MESSAGE_PRIORITY             | TpMessagePriority    | Priority            |

### 11.1.37 TpMessageHeaderFieldSet

Defines a numbered set of data elements of TpMessageHeaderField.

### 11.1.38 TpMessagePriority

Defines the priority of a message.

| Name                             | Value | Description      |
|----------------------------------|-------|------------------|
| P_MMM_MESSAGE_PRIORITY_UNDEFINED | 0     | Undefined/Normal |
| P_MMM_MESSAGE_PRIORITY_HIGH      | 1     | High priority    |
| P_MMM_MESSAGE_PRIORITY_LOW       | 2     | Low priority     |

### 11.1.39 TpMessageDeliveryReportType

Defines the type of message delivery report requested or reported. In order to request multiple report types at once, the values can be added together (i.e. a logical OR function). So a value of 3 identifies both Read and Delivery reports.

| Name                                | Value | Description   |
|-------------------------------------|-------|---|
| P_MESSAGE_REPORT_DELIVERY_UNDEFINED | 0     | Undefined / status unknown.   |
| P_MESSAGE_REPORT_DELIVERED          | 1     | A report that the message has been successfully delivered.  |
| P_MESSAGE_REPORT_READ               | 2     | A report that the message was read.   |
| P_MESSAGE_REPORT_DELETED_UNREAD     | 4     | A report that the message was deleted without being read.   |
| P_MESSAGE_REPORT_NOT_DELIVERABLE    | 8     | A report that the message could not be successfully delivered.                                    |
| P_MESSAGE_REPORT_EXPIRED            | 16    | A report that the message could not be delivered before the validity time of the message expired. |

### 11.1.40 TpMessageTreatment

Defines the specific message treatment which an application requests of the messaging system when sending a message.

| Tag Element Type       |
|------------------------|
| TpMessageTreatmentType |

| Tag Element Value                | Choice Element Type         | Choice Element Name |
|----------------------------------|-----------------------------|---------------------|
| P_MMM_TREATMENT_REPORT_REQUESTED | TpMessageDeliveryReportType | DeliveryReport      |
| P_MMM_TREATMENT_BILLING_ID       | TpString                    | BillingID           |
| P_MMM_TREATMENT_DELIVERY_TIME    | TpDeliveryTime              | DeliveryTime        |
| P_MMM_TREATMENT_VALIDITY_TIME    | TpDateAndTime               | ValidityTime        |

### 11.1.41 TpMessageTreatmentType

Specifies the various types of message treatment which the application may request when sending a message.

| Name                             | Value | Description  |
|----------------------------------|-------|--|
| P_MMM_TREATMENT_UNDEFINED        | 0     | Undefined  |
| P_MMM_TREATMENT_REPORT_REQUESTED | 1     | Request various delivery notifications   |
| P_MMM_TREATMENT_BILLING_ID       | 2     | A Billing Identifier/Charging Code that can be used to indicate how the costs for this transaction shall be charged.   |
| P_MMM_TREATMENT_DELIVERY_TIME    | 3     | Specifies whether the message shall be delivered instantly or at some specified time.  |
| P_MMM_TREATMENT_VALIDITY_TIME    | 4     | Specifies the time within which the message keeps its validity. When the message is not delivered before the validity time passes, the message is dropped by the messaging system. |

### 11.1.42 TpMessageTreatmentSet

Defines a Numbered Set of Data Elements of TpMessageTreatment.

### 11.1.43 TpMultiMediaMessagingIdentifier

Defines the Sequence of Data Elements that identify a mailbox.

| Sequence Element Name | Sequence Element Type    |
|-----------------------|--------------------------|
| MultiMediaMessaging   | IpMultiMediaMessagingRef |
| SessionID             | TpSessionID              |

#### 11.1.44 TpMultiMediaMessagingIdentifierSet

Defines a Numbered Set of Data Elements of TpMultiMediaMessagingIdentifier.

#### 11.1.45 TpQueryStatusReport

Defines the Sequence of Data Elements that identify the status of a message, as reported in messageStatusReport. If a message is sent to more than one addressee, a different status can apply to the message sent to each addressee.

| Sequence Element Name | Sequence Element Type       |
|-----------------------|-----------------------------|
| DestinationAddress    | TpAddress                   |
| ReportedStatus        | TpMessageDeliveryReportType |

#### 11.1.46 TpQueryStatusReportSet

Defines a numbered set of data elements of TpQueryStatusReport.

#### 11.1.47 TpTerminatingAddressList

Defines the Sequence of Data Elements that specify a send message request.

| Sequence Element Name | Sequence Element Type | Description  |
|-----------------------|-----------------------|--|
| ToAddressList         | TpAddressSet          | The list of addresses that are the main target of the message.   |
| CcAddressList         | TpAddressSet          | The list of addresses that receive a copy of the message.  |
| BccAddressList        | TpAddressSet          | The list of addresses that receive a copy of the message, without this being visible to other receivers. |

## 11.2 Event Notification data definitions

### 11.2.1 TpMessagingEventName

Defines the names of the messaging events which can be notified.

| Name                                    | Value | Description   |
|---|-------|---|
| P_EVENT_MSG_NAME_UNDEFINED              | 0     | Undefined.  |
| P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED | 1     | New message arrived in the mailbox. The message contents are not requested/delivered - the message can be retrieved from the mailbox. |
| P_EVENT_MSG_NEW_MESSAGE_ARRIVED         | 2     | New message arrived. The Message contents are requested/delivered with this event.  |

## 11.2.2 TpMessagingEventCriteria

Defines the Tagged Choice of Data Elements that specify the criteria for an event notification to be generated.

|  | Tag Element Type     |  |
|--|----------------------|--|
|  | TpMessagingEventName |  |

| Tag Element Value                       | Choice Element Type                | Choice Element Name           |
|---|------------------------------------|-------------------------------|
| P_EVENT_MSG_NAME_UNDEFINED              | NULL                               | Undefined                     |
| P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED | TpNewMailboxMessageArrivedCriteria | EventNewMailboxMessageArrived |
| P_EVENT_MSG_NEW_MESSAGE_ARRIVED         | TpNewMessageArrivedCriteria        | EventNewMessageArrived        |

## 11.2.3 TpMessagingEventCriteriaSet

Defines a numbered set of data elements of TpMessagingEventCriteria.

## 11.2.4 TpNewMailboxMessageArrivedCriteria

Defines the Sequence of Data Elements that specify the criteria for a New Mailbox Message Arrived event.

| Sequence Element Name | Sequence Element Type |
|-----------------------|-----------------------|
| MailboxID             | TpString              |
| AuthenticationInfo    | TpString              |

## 11.2.5 TpNewMessageArrivedCriteria

Defines the Sequence of Data Elements that specify the criteria for a New Message Arrived event, for messages which are received outside the context of a mailbox.

| Sequence Element Name            | Sequence Element Type | Description  |
|----------------------------------|-----------------------|--|
| SourceAddress                    | TpAddressRange        | Defines the source address for which the notification is requested.  |
| DestinationAddress               | TpAddressRange        | Defines the destination address or address range for which the notification is requested.  |
| CreateMultiMediaMessagingSession | TpBoolean             | Used by the application to indicate if the SCF should create an instance of IpMultiMediaMessaging upon receipt of an event which matches these criteria. |

## 11.2.6 TpMessagingEventInfo

Defines the Tagged Choice of Data Elements that specify the information returned to the application in an event notification.

|  | Tag Element Type     |  |
|--|----------------------|--|
|  | TpMessagingEventName |  |

| Tag Element Value                       | Choice Element Type            | Choice Element Name           |
|---|--------------------------------|-------------------------------|
| P_EVENT_MSG_NAME_UNDEFINED              | TpString                       | EventNameUndefined            |
| P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED | TpNewMailboxMessageArrivedInfo | EventNewMailboxMessageArrived |
| P_EVENT_MSG_NEW_MESSAGE_ARRIVED         | TpNewMessageArrivedInfo        | EventNewMessageArrived        |

### 11.2.7 TpMessagingEventInfoSet

Defines a numbered set of data elements of TpMessagingEventInfo.

### 11.2.8 TpNewMailboxMessageArrivedInfo

Defines the Sequence of Data Elements that specify the information returned to the application in a New Mailbox Message Arrived event. Basic information is provided in the Message Description field. If further information is to be provided, it is included in the ExtendedHeaderInformation field.

| Sequence Element Name     | Sequence Element Type   |
|---------------------------|-------------------------|
| MailboxID                 | TpString                |
| FolderID                  | TpString                |
| MessageDescription        | TpMessageDescription    |
| ExtendedHeaderInformation | TpMessageHeaderFieldSet |

### 11.2.9 TpNewMessageArrivedInfo

Defines the Sequence of Data Elements that specify the information returned to the application in a New Message Arrived event. The Source and Destination addresses are included, as they are used in the criteria for this event. The actual message received is included in the Message field. This contains the entire message (headers and body) in unstructured format. The Headers field contains header information which was sent as part of the message. This information could be duplicated in the raw message, depending on its format.

| Sequence Element Name         | Sequence Element Type           |
|-------------------------------|---------------------------------|
| SourceAddress                 | TpAddress                       |
| DestinationAddressSet         | TpAddressSet                    |
| Message                       | TpOctetSet                      |
| Headers                       | TpMessageHeaderFieldSet         |
| MultiMediaMessagingIdentifier | TpMultiMediaMessagingIdentifier |

### 11.2.10 TpMessageDescription

Specifies the properties of a message.

| Sequence Element Name | Sequence Element Type | Description  |
|-----------------------|-----------------------|--|
| MessageID             | TpString              | This is the messageID as it is used on the API to identify a message. This should not be confused with the RFC 822 "Message-ID" field body. The latter can be obtained with getMessageHeadersReq(). The messageID used on the API is persistent over sessions and at least unique within the context of a mailbox. |
| From                  | TpAddress             | The address of the sender of the message.  |
| To                    | TpAddressSet          | A set of addresses representing the primary recipients of the message.   |
| Subject               | TpString              | Specifies the subject of the message.  |
| ReceivedDate          | TpDateAndTime         | Specifies the date/time that the message was received by the messaging system.   |
| Size                  | TpInt32               | Specifies the size of the message in bytes.  |

### 11.2.11 TpMessageDescriptionList

Defines a numbered list of data elements of TpMessageDescription.

### 11.2.12 TpMessagingNotificationRequested

Defines the Sequence of Data Elements that specify the criteria relating to event requests.

| Sequence Element Name | Sequence Element Type       |
|-----------------------|-----------------------------|
| EventCriteria         | TpMessagingEventCriteriaSet |
| AssignmentID          | TpInt32                     |

### 11.2.13 TpMessagingNotificationRequestedSet

Defines a numbered Set of Data Elements of TpMessagingNotificationRequested.

### 11.2.14 TpMessagingNotificationRequestedSetEntry

Defines the Sequence of Data Elements that specify a set of requested notifications and an indication whether more notifications can be requested.

| Sequence Element Name             | Sequence Element Type               | Description   |
|-----------------------------------|-------------------------------------|---|
| MessagingNotificationRequestedSet | TpMessagingNotificationRequestedSet | Numbered set of requested notifications.  |
| Final                             | TpBoolean                           | Indication whether the set of notifications is the final set (TRUE) or if there are more notifications available (FALSE). |

## 11.3 Error type data definitions

### 11.3.1 TpMessageInfoPropertyError

Defines the Sequence of Data Elements that identify a message property and the reason why it was not set.

| Sequence Element Name | Sequence Element Type     |
|-----------------------|---------------------------|
| MessagePropertyName   | TpMessageInfoPropertyName |
| Error                 | TpSetPropertyError        |



### 11.3.2 TpMessagingError

A set of general error identifiers to include in Err methods.

| Name  | Value | Description   |
|---|-------|---|
| P_MMM_ERROR_UNDEFINED                           | 0     | Undefined   |
| P_MMM_ERROR_INVALID_AUTHENTICATION_INFORMATION  | 1     | Authentication Information is not valid   |
| P_MMM_ERROR_INVALID_MAILBOX                     | 2     | Chosen Mailbox Address is invalid   |
| P_MMM_ERROR_INVALID_DELIVERY_TYPE               | 3     | The delivery mechanism specified is invalid or is not supported   |
| P_MMM_ERROR_MAX_MESSAGE_SIZE_EXCEEDED           | 4     | The maximum size of a message has been exceeded   |
| P_MMM_ERROR_INVALID_FOLDER_ID                   | 5     | The folder ID is invalid  |
| P_MMM_ERROR_INVALID_MESSAGE_ID                  | 6     | The message ID is invalid   |
| P_MMM_ERROR_INVALID_PART_ID                     | 7     | The part ID is invalid  |
| P_MMM_ERROR_DELIVERY_TYPE_ADDRESS_TYPE_MISMATCH | 8     | The address type does not match the message delivery type   |
| P_MMM_ERROR_DELIVERY_TYPE_MESSAGE_TYPE_MISMATCH | 9     | The message format does not match the message delivery type   |
| P_MMM_ERROR_INVALID_DELIVERY_TIME               | 10    | The requested delivery time is invalid  |
| P_MMM_ERROR_INVALID_VALIDITY_TIME               | 11    | The requested validity time is invalid  |
| P_MMM_ERROR_MAX_SUBJECT_SIZE_EXCEEDED           | 12    | The maximum size of the subject field has been exceeded   |
| P_MMM_ERROR_INVALID_ID                          | 13    | The BillingID is invalid  |
| P_MMM_ERROR_INVALID_NESTING_LEVEL               | 14    | The nesting level is invalid  |
| P_MMM_ERROR_INVALID_CRITERIA                    | 15    | The criteria specified are invalid  |
| P_MMM_ERROR_INFORMATION_NOT_AVAILABLE           | 16    | The requested information is not available, e.g. message status information                                 |
| P_MMM_ERROR_CANNOT_CANCEL                       | 17    | The message cannot be cancelled   |
| P_MMM_ERROR_INVALID_HEADER                      | 18    | The message header is invalid   |
| P_MMM_INVALID_NETWORK_STATE                     | 19    | Although the sequence of method calls is allowed by the gateway, the underlying protocol cannot support it. |
| P_MMM_ERROR_RESOURCE_UNAVAILABLE                | 20    | The information resources used by the messaging service are unavailable, e.g. due to an overload situation. |
| P_MMM_ERROR_RESOURCE_TIMEOUT                    | 21    | The request has been accepted by the resource but it did not report a result.                               |

### 11.3.3 TpMessageInfoPropertyErrorSet

Defines a Numbered Set of Data Elements of TpMessageInfoPropertyError.

### 11.3.4 TpSetPropertyError

Identifies the reason why a property was not set.

| Name                                  | Value | Description  |
|---------------------------------------|-------|--|
| P_MMM_PROPERTY_NOT_SET                | 0     | The property was not set. No further information is available.     |
| P_MMM_PROPERTY_READONLY               | 1     | The property is readonly and cannot be modified by an application. |
| P_MMM_PROPERTY_INSUFFICIENT_PRIVILEGE | 2     | The application is not authorised to modify the property.          |
| P_MMM_PROPERTY_NAME_UNKNOWN           | 3     | The property name is unknown.                                      |

## 12 Exception Classes

The following are the list of exception classes which are used in this interface of the API.

| Name                                      | Description  |
|---|--|
| P_MMM_INVALID_AUTHENTICATION_INFORMATION  | Authentication Information is not valid.                                     |
| P_MMM_INVALID_MAILBOX                     | Chosen Mailbox Address is invalid.   |
| P_MMM_INVALID_DELIVERY_TYPE               | The delivery mechanism specified is invalid or is not supported.             |
| P_MMM_MAX_MESSAGE_SIZE_EXCEEDED           | The maximum size of a message has been exceeded.                             |
| P_MMM_INVALID_FOLDER_ID                   | The folder ID is invalid.  |
| P_MMM_INVALID_MESSAGE_ID                  | The message ID is invalid.   |
| P_MMM_INVALID_PART_ID                     | The part ID is invalid.  |
| P_MMM_DELIVERY_TYPE_ADDRESS_TYPE_MISMATCH | The address type does not match the message delivery type.                   |
| P_MMM_DELIVERY_TYPE_MESSAGE_TYPE_MISMATCH | The message format does not match the message delivery type.                 |
| P_MMM_INVALID_PROPERTY                    | The property name is invalid.  |
| P_MMM_INVALID_DELIVERY_TIME               | The requested delivery time is invalid.                                      |
| P_MMM_INVALID_VALIDITY_TIME               | The requested validity time is invalid.                                      |
| P_MMM_MAX_SUBJECT_SIZE_EXCEEDED           | The maximum size of the subject field has been exceeded.                     |
| P_MMM_INFORMATION_NOT_AVAILABLE           | The requested information is not available, e.g. message status information. |
| P_MMM_CANNOT_CANCEL                       | The message cannot be cancelled.   |
| P_MMM_INVALID_HEADER                      | The message header is invalid.   |

Each exception class contains the following structure.

| Structure Element Name | Structure Element Type | Structure Element Description  |
|------------------------|------------------------|--|
| ExtraInformation       | TpString               | Carries extra information to help identify the source of the exception, e.g. a parameter name. |

---

## Annex A (normative): OMG IDL Description of Multi-Media Messaging SCF

The OMG IDL representation of this interface specification is contained in a text file `mmm.idl` contained in archive `es_20391515IDL.ZIP`.

This archive can be found in `es_20391515v010201p0.zip` which accompanies the present document.

---

## Annex B (informative): W3C WSDL Description of Multi-Media Messaging SCF

The W3C WSDL representation of this interface specification is contained in archive es\_20391515WSDL.ZIP.

This archive can be found in es\_20391515v010201p0.zip which accompanies the present document.

---

## Annex C (informative): Java API Description of the Multi-Media Messaging SCF

The Java API realisation of this interface specification is produced in accordance with the Java Realisation rules defined in ES 203 915-1. These rules aim to deliver for Java, a developer API, provided as a realisation, supporting a Java API that represents the UML specifications. The rules support the production of both J2SE and J2EE versions of the API from the common UML specifications.

The J2SE representation of this interface specification is provided as Java Code, contained in archive 20391515J2SE.ZIP.

The J2EE representation of this interface specification is provided as Java Code, contained in archive 20391515J2EE.ZIP.

Both these archives can be found in es\_20391515v010201p0.zip which accompanies the present document.

---

## Annex D (informative): Contents of 3GPP OSA R6 Multi-Media Messaging

All of the present document is relevant for TS 129 198-15 V6 (Release 6).

---

## Annex E (informative): Record of changes

The following is a list of the changes made to the present document for each release. The list contains the names of all changed, deprecated, added or removed items in the specifications and not the actual changes. Any type of change information that is important to the reader is put in the final clause of this annex.

Changes are specified as changes to the prior major release, but every minor release will have its own part of the table allowing the reader to know when the actual change was made.

---

### E.1 Interfaces

#### E.1.1 New

| Identifier   | Comments |
|--|----------|
| Interfaces added in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Interfaces added in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

#### E.1.2 Deprecated

| Identifier  | Comments |
|---|----------|
| Interfaces deprecated in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|   |          |
| Interfaces deprecated in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|   |          |

#### E.1.3 Removed

| Identifier   | Comments |
|--|----------|
| Interfaces removed in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Interfaces removed in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

---

### E.2 Methods

#### E.2.1 New

| Identifier  | Comments |
|---|----------|
| Methods added in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|   |          |
| Methods added in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|   |          |

## E.2.2 Deprecated

| Identifier   | Comments |
|--|----------|
| Methods deprecated in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Methods deprecated in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

## E.2.3 Modified

| Identifier   | Comments |
|--|----------|
| Methods modified in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Methods modified in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

## E.2.4 Removed

| Identifier  | Comments |
|---|----------|
| Methods removed in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|   |          |
| Methods removed in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|   |          |

---

## E.3 Data Definitions

### E.3.1 New

| Identifier   | Comments |
|--|----------|
| Data Definitions added in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Data Definitions added in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

### E.3.2 Modified

| Identifier  | Comments                           |
|---|------------------------------------|
| Data Definitions modified in ES 203 915-15 version 1.1.1 (Parlay 5.0) |                                    |
|   |                                    |
| Data Definitions modified in ES 203 915-15 version 1.2.1 (Parlay 5.1) |                                    |
| TpMessageTreatment  | Default Dummy element added in IDL |



### E.3.3 Removed

| Identifier | Comments   |
|------------|--|
|            | Data Definitions removed in ES 203 915-15 version 1.1.1 (Parlay 5.0) |
|            |  |
|            | Data Definitions removed in ES 203 915-15 version 1.2.1 (Parlay 5.1) |
|            |  |

---

## E.4 Service Properties

### E.4.1 New

| Identifier | Comments   |
|------------|--|
|            | Service Properties added in ES 203 915-15 version 1.1.1 (Parlay 5.0) |
|            |  |
|            | Service Properties added in ES 203 915-15 version 1.2.1 (Parlay 5.1) |
|            |  |

### E.4.2 Deprecated

| Identifier | Comments  |
|------------|---|
|            | Service Properties deprecated in ES 203 915-15 version 1.1.1 (Parlay 5.0) |
|            |   |
|            | Service Properties deprecated in ES 203 915-15 version 1.2.1 (Parlay 5.1) |
|            |   |

### E.4.3 Modified

| Identifier | Comments  |
|------------|---|
|            | Service Properties modified in ES 203 915-15 version 1.1.1 (Parlay 5.0) |
|            |   |
|            | Service Properties modified in ES 203 915-15 version 1.2.1 (Parlay 5.1) |
|            |   |

### E.4.4 Removed

| Identifier | Comments   |
|------------|--|
|            | Service Properties removed in ES 203 915-15 version 1.1.1 (Parlay 5.0) |
|            |  |
|            | Service Properties removed in ES 203 915-15 version 1.2.1 (Parlay 5.1) |
|            |  |

---

## E.5 Exceptions

### E.5.1 New

| Identifier   | Comments |
|--|----------|
| Exceptions added in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Exceptions added in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

### E.5.2 Modified

| Identifier  | Comments |
|---|----------|
| Exceptions modified in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|   |          |
| Exceptions modified in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|   |          |

### E.5.3 Removed

| Identifier   | Comments |
|--|----------|
| Exceptions removed in ES 203 915-15 version 1.1.1 (Parlay 5.0) |          |
|  |          |
| Exceptions removed in ES 203 915-15 version 1.2.1 (Parlay 5.1) |          |
|  |          |

---

## E.6 Others

ES 203 915-15 (V1.2.1):

WSDL code reworked and replaced in annex B.

---

## History

| <b>Document history</b> |              |  |
|-------------------------|--------------|--|
| V1.1.1                  | Avril 2005   | Publication  |
| V1.2.1                  | October 2006 | Membership Approval Procedure    MV 20061222: 2006-10-24 to 2006-12-22 |
| V1.2.1                  | January 2007 | Publication  |
|                         |              |  |
|                         |              |  |