3GPP TSG CT Plenary Meeting #28 **CP-~~050158~~050220**
01-03 June 2005, Quebec, CANADA

Source:	**CT5 (OSA)**
Title:	**2 Rel-6 CR 29.198-15**
Agenda item:	**9.7 (OSA Enhancements [OSA3])**
Document for:	**APPROVAL**

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Doc-2nd-Level	Workitem
CP-~~050158~~050220	29.198-15	0002	~~-~~1	Rel-6	Clarification of Multi Media Messaging using Sequence Diagrams	F	6.1.1	~~C5-050228~~	OSA3
CP-~~050158~~050220	29.198-15	0003	-	Rel-6	Correction to TpMessageTreatment in IDL	F	6.1.1	C5-050231	OSA3

CR-Form-v7.1

CHANGE REQUEST

⌘ **29.198-15 CR 0002** ⌘ **rev** ~~-~~1 ⌘ Current version: **6.1.1** ⌘

For HELP on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Clarification of Multi Media Messaging using Sequence Diagrams		
Source:	⌘	~~CT5 Ultan Mulligan, ETSI Secretariat~~CT#28		
Work item code:	⌘	OSA3	***Date:*** ⌘	~~18/04/05~~02/05/2005
Category:	⌘	**F**	***Release:*** ⌘	Rel-6
		Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *Ph2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)* *Rel-7 (Release 7)*

Reason for change:	⌘	TS 29.198-15 has been developed without including any sequence diagrams. Sequence diagrams are an essential part of the specification, necessary in order to understand how the specification is intended to work. While they do not change the functionality in the specification, they clarify the intended message exchange, and reduce the possibility of mis-interpretation of the specification, and therefore reduce the possibility of interoperability problems.
Summary of change:	⌘	Sequence diagrams for the following scenarios are introduced in clause 5: - Sending messages and receiving delivery notification - Sending, and receiving messages in same context - Setting notification of received messages - Using Mailbox functions - Using Mailbox to send and receive
Consequences if not approved:	⌘	Failure to include sequence diagrams in this specification may result in mis-interpretation of the specification, confusion among developers, and potential interoperability problems. This may eventually lead to lack of adoption of the specification by developers.

Clauses affected:	⌘	5				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

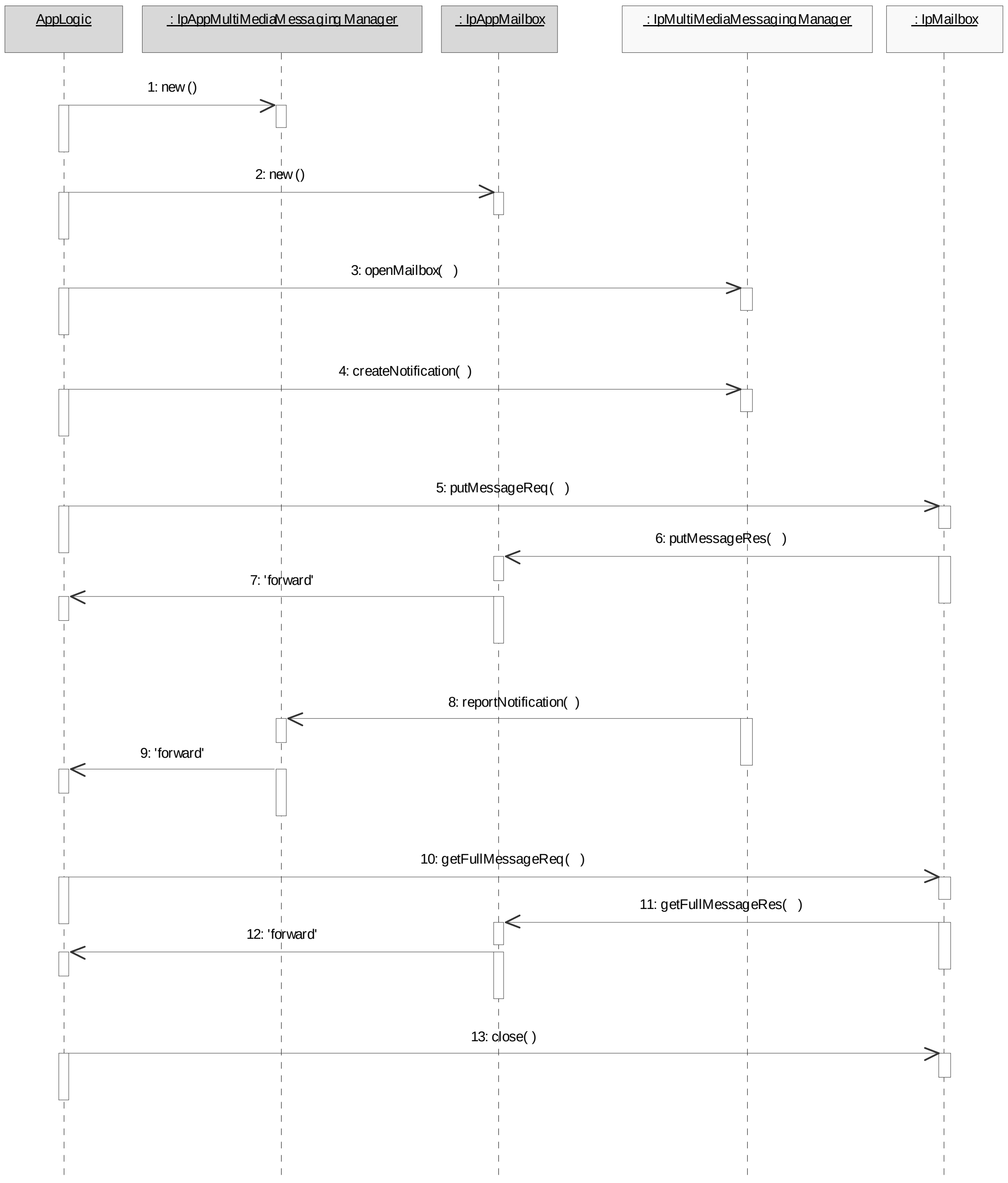
Change in Clause 5

5 Sequence Diagrams

~~There are no Sequence Diagrams for the Multi-Media Messaging SCF.~~

5.1 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.

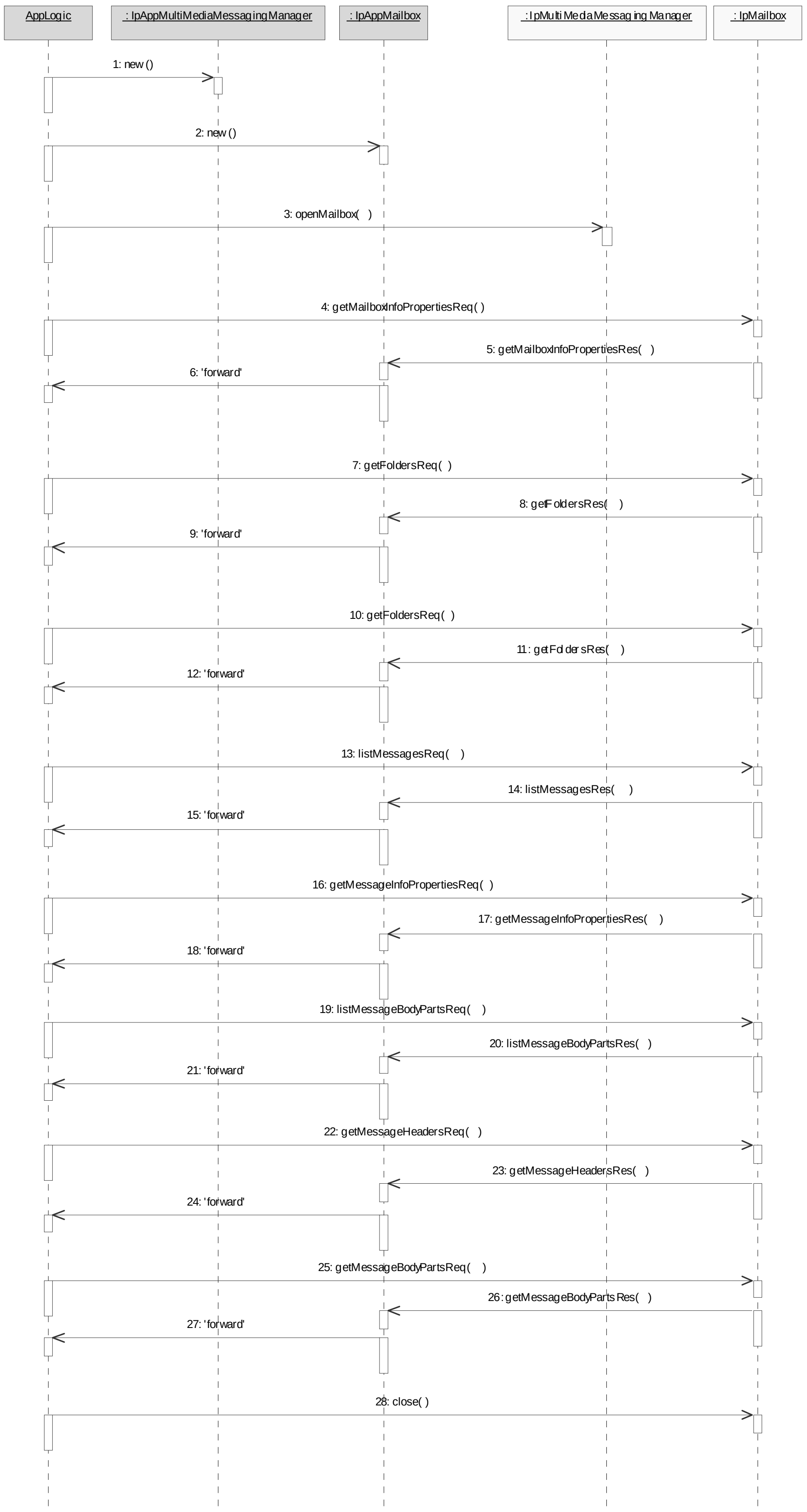
10: Using the messageID and folderID received in the reportNotification() method, the application requests to retrieve the full contents of the received message from the mailbox. The application could have chosen to retrieve individual parts of the message using getMessageBodyPartsReq(), or to retrieve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

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

13: The applicaiton closes the mailbox session.

5.2 Using Mailbox functions

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


AppLogic
: IpAppMultiMediaMessagingManager
: IpAppMailbox
: IpMultiMediaMessagingManager
: IpMailbox
1: new ()
2: new ()
3: openMailbox( )
4: getMailboxInfoPropertiesReq( )
5: getMailboxInfoPropertiesRes( )
6: 'forward'
7: getFoldersReq( )
8: getFoldersRes( )
9: 'forward'
10: getFoldersReq( )
11: getFoldersRes( )
12: 'forward'
13: listMessagesReq( )
14: listMessagesRes( )
15: 'forward'
16: getMessageInfoPropertiesReq( )
17: getMessageInfoPropertiesRes( )
18: 'forward'
19: listMessageBodyPartsReq( )
20: listMessageBodyPartsRes( )
21: 'forward'
22: getMessageHeadersReq( )
23: getMessageHeadersRes( )
24: 'forward'
25: getMessageBodyPartsReq( )
26: getMessageBodyPartsRes( )
27: 'forward'
28: close( )

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.

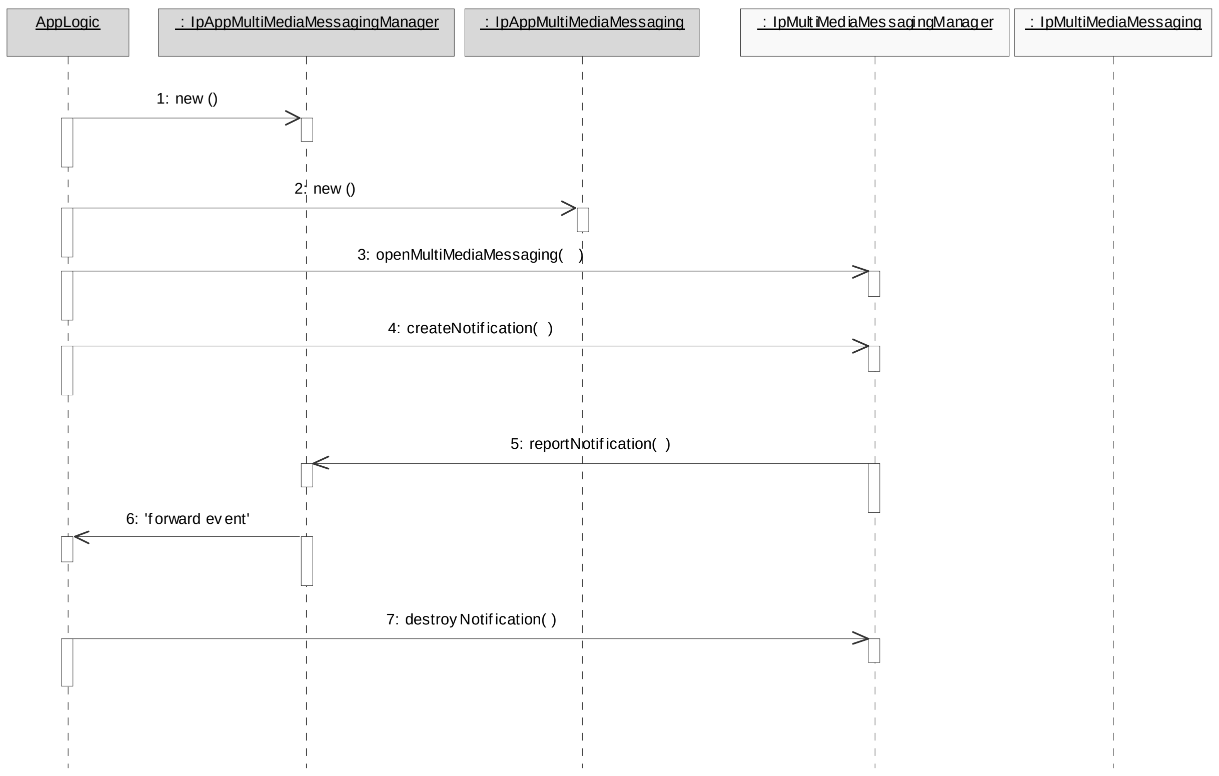
25: The application retrieves one or more parts of the message, as identified in the partIDs parameter.

26: The contents of the requested messge parts are returned.

28: The applicaiton closes the mailbox session.

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.


AppLogic
: IpAppMultiMediaMessagingManager
: IpAppMultiMediaMessaging
: IpMultiMediaMessagingManager
: IpMultiMediaMessaging
1: new ()
2: new ()
3: openMultiMediaMessaging( )
4: createNotification( )
5: reportNotification( )
6: 'forward event'
7: destroyNotification( )

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

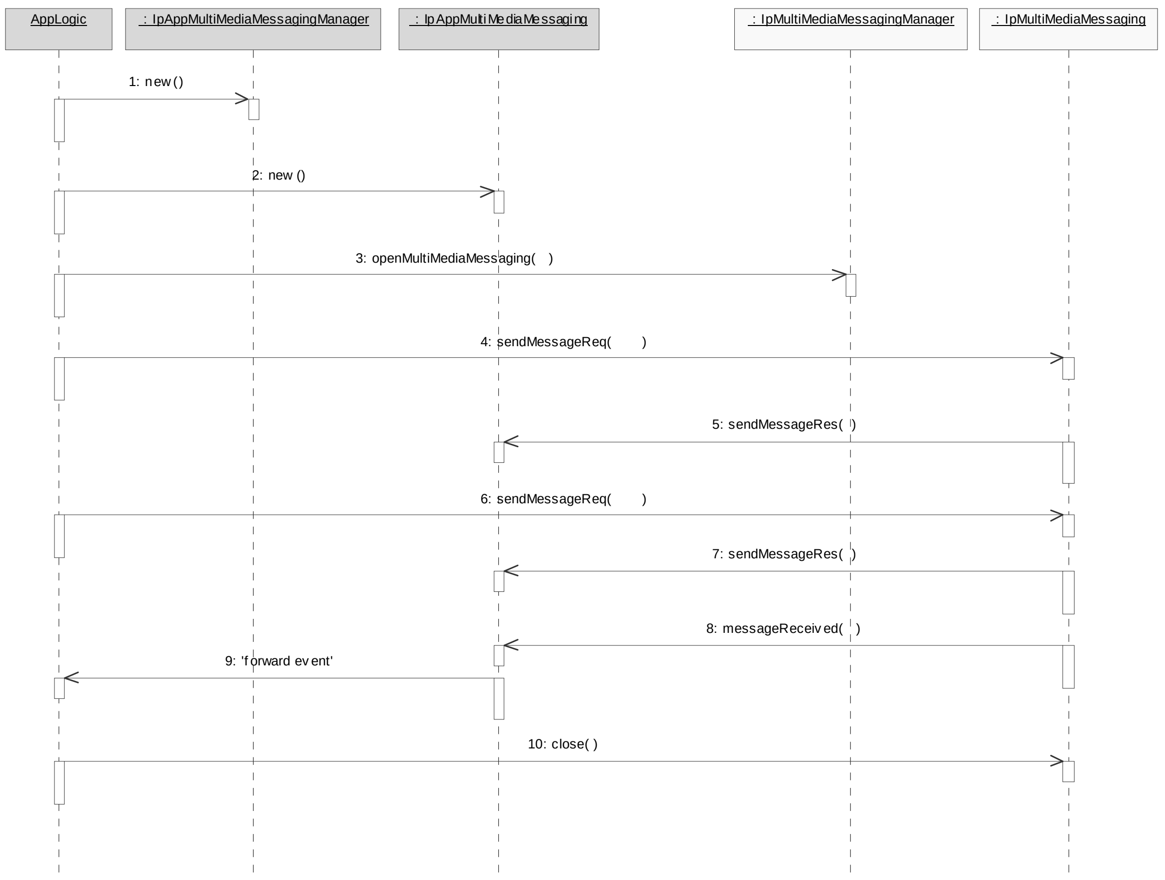
4: The application requests to be notified of any messages received for a particular destination address, using the P_EVENT_MSG_NEW_MESSAGE_ARRIVED criteria. The application may request that a MultiMedia Messaging session is created upon receipt of a message.

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 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.

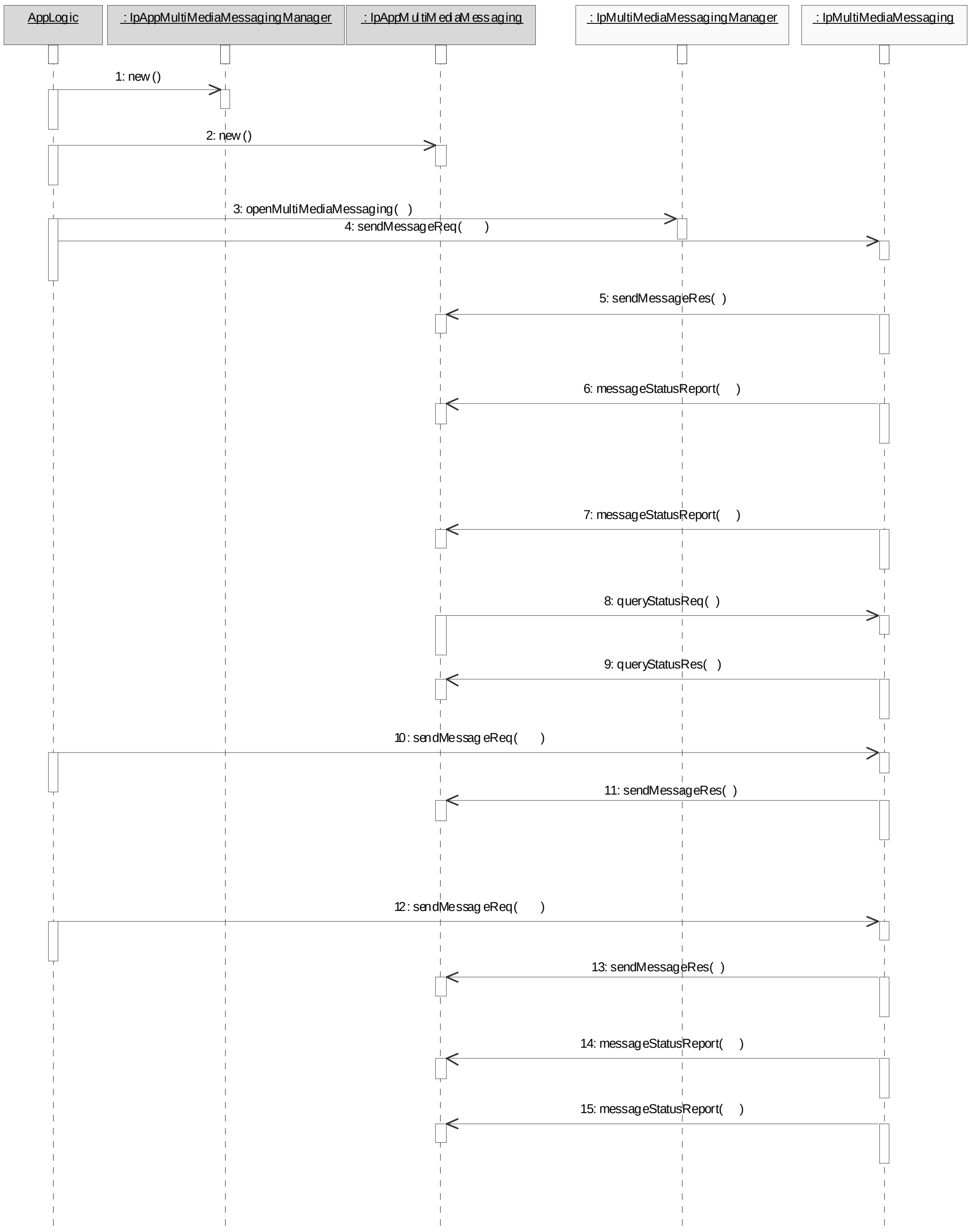
7: This method indicates successful processing of the sendMessageReq by the SCF, and that the message has been sent. It does not indicate a delivery status.

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.

5.5 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().


AppLogic
: IpAppMultiMediaMessagingManager
: IpAppMultiMediaMessaging
: IpMultiMediaMessagingManager
: IpMultiMediaMessaging
1: new()
2: new()
3: openMultiMediaMessaging( )
4: sendMessageReq( )
5: sendMessageRes( )
6: messageStatusReport( )
7: messageStatusReport( )
8: queryStatusReq( )
9: queryStatusRes( )
10: sendMessageReq( )
11: sendMessageRes( )
12: sendMessageReq( )
13: sendMessageRes( )
14: messageStatusReport( )
15: messageStatusReport( )

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().

End of Change in Clause 5

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040359	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	6.0.0
Dec 2004	CN_26	NP-040485	001	--	Removal of OSA API SCFs description in W3C WSDL	6.0.0	6.1.0
Dec 2004	--	--	--	--	Added missing code attachments	6.1.0	6.1.1

Joint-Working-Group (Parlay, ETSI Project OSA, 3GPP CT5) **C5-050231**
Meeting #31, Osaka, JAPAN, 09-13 May 2005

CR-Form-v7.1

CHANGE REQUEST

⌘	**29.198-15**	**CR**	**0003**	⌘ **rev**	**-**	⌘	Current version:	**6.1.1**	⌘

For ***HELP*** *on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Correction to TpMessageTreatment in IDL		
Source:	⌘	CT5 Ultan Mulligan, ETSI Secretariat		
Work item code:	⌘	OSA3	***Date:*** ⌘	18/04/2005
Category:	⌘	**F**	***Release:*** ⌘	Rel-6
		Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *Ph2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)* *Rel-7 (Release 7)*

Reason for change:	⌘	The IDL code by convention includes a default Dummy element in Union types, where there are more values in the discriminator than there are elements in the union. This is in order to enable the extraction of the discriminator value, in cases where the discriminator value is not encoded separately. One type in the Multi Media Messaging specification is missing this default: TpMessageTreatment
Summary of change:	⌘	Insert a default element, named Dummy, of type short, in TpMessageTreatment
Consequences if not approved:	⌘	Developers may have problems when attempting to implement MultiMedia Messaging SCF. These problems may be more or less serious, depending on which language they are using for their implementation, i.e. depending on the IDL mapping to their language.

Clauses affected:	⌘	Annex A				
		Y	**N**			
Other specs affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

Change in IDL file contained in Annex A

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 2919815V610IDL.ZIP) which accompanies the present document.

(The following is the contents of the 2919815V610IDL.zip file, revision marked to show the changes)

```
//Source file: mmm.idl
//Date:  7 January 2005
//Multi Media Messaging for for 3GPP TS 29.198-15 V6.1.0


#ifndef __MMM_DEFINED
#define __MMM_DEFINED


#include "osa.idl"

module org {

  module csapi {


    module mmm {

      interface IpMailbox;
      interface IpMultiMediaMessaging;
      interface IpAppMailbox;
      interface IpAppMultiMediaMessaging;

      struct TpMailboxIdentifier {
        IpMailbox Mailbox;
        TpSessionID SessionID;
      };


      exception P_MMM_INVALID_AUTHENTICATION_INFORMATION {
        TpString ExtraInformation;
      };


      exception P_MMM_INVALID_MAILBOX {
        TpString ExtraInformation;
      };


      struct TpMultiMediaMessagingIdentifier {
        IpMultiMediaMessaging MultiMediaMessaging;
        TpSessionID SessionID;
      };
```

```
exception P_MMM_INVALID_DELIVERY_TYPE {
  TpString ExtraInformation;
};


enum TpFolderInfoPropertyName {

  P_MMM_FOLDER_UNDEFINED,
  P_MMM_FOLDER_DATE_CREATED,
  P_MMM_FOLDER_DATE_CHANGED,
  P_MMM_FOLDER_SIZE,
  P_MMM_FOLDER_NUMBER_OF_MESSAGES
};


union TpFolderInfoProperty switch(TpFolderInfoPropertyName) {
  case P_MMM_FOLDER_DATE_CREATED: TpDateAndTime FolderDateCreated;
  case P_MMM_FOLDER_DATE_CHANGED: TpDateAndTime FolderDateChanged;
  case P_MMM_FOLDER_SIZE: TpInt32 FolderSize;
  case P_MMM_FOLDER_NUMBER_OF_MESSAGES: TpInt32 FolderNumberOfMessages;
  default: short Dummy;
};


typedef sequence <TpFolderInfoProperty> TpFolderInfoPropertySet;


enum TpMailboxInfoPropertyName {

  P_MMM_MAILBOX_UNDEFINED,
  P_MMM_MAILBOX_OWNER,
  P_MMM_MAILBOX_DATE_CREATED,
  P_MMM_MAILBOX_DATE_CHANGED,
  P_MMM_MAILBOX_SIZE
};


union TpMailboxInfoProperty switch(TpMailboxInfoPropertyName) {
  case P_MMM_MAILBOX_OWNER: TpString MailboxOwner;
  case P_MMM_MAILBOX_DATE_CREATED: TpDateAndTime MailboxDateCreated;
  case P_MMM_MAILBOX_DATE_CHANGED: TpDateAndTime MailboxDateChanged;
  case P_MMM_MAILBOX_SIZE: TpInt32 MailboxSize;
  default: short Dummy;
};


enum TpMessageInfoPropertyName {

  P_MMM_MESSAGE_UNDEFINED,
  P_MMM_MESSAGE_DATE_CREATED,
  P_MMM_MESSAGE_DATE_RECEIVED,
  P_MMM_MESSAGE_DATE_CHANGED,
  P_MMM_MESSAGE_SIZE,
  P_MMM_MESSAGE_STATUS
};


enum TpMessagePriority {

  P_MMM_MESSAGE_PRIORITY_UNDEFINED,
  P_MMM_MESSAGE_PRIORITY_HIGH,
  P_MMM_MESSAGE_PRIORITY_LOW
};
```

```
typedef sequence <TpMailboxInfoProperty> TpMailboxInfoPropertySet;


struct TpListMessagesCriteria {
  TpBoolean OnlyUnreadMessages;
};


struct TpMailboxFolderStatusInformation {
  TpInt32 TotalMessageCount;
};


struct TpMessageDescription {
  TpString MessageID;
  TpAddress From;
  TpAddressSet To;
  TpString Subject;
  TpDateAndTime ReceivedDate;
  TpInt32 Size;
};


typedef sequence <TpMessageDescription> TpMessageDescriptionList;


struct TpBodyPartDescription {
  TpString ContentDescription;
  TpInt32 ContentSize;
  TpString ContentType;
  TpString ContentTransferEncoding;
  TpString ContentID;
  TpString ContentDisposition;
  TpString PartID;
  TpInt32 NestingLevel;
};


typedef sequence <TpBodyPartDescription> TpBodyPartDescriptionList;


struct TpBodyPart {
  TpBodyPartDescription BodyPartHeader;
  TpOctetSet BodyPartContent;
};


typedef sequence <TpBodyPart> TpBodyPartList;


enum TpMessageHeaderFieldType {

  P_MESSAGE_DATE_SENT,
  P_MESSAGE_SENT_FROM,
  P_MESSAGE_SENDER,
  P_MESSAGE_REPLY_TO,
  P_MESSAGE_SENT_TO,
  P_MESSAGE_CC_TO,
  P_MESSAGE_BCC_TO,
  P_MESSAGE_RFC822_MESSAGE_ID,
  P_MESSAGE_IN_REPLY_TO,
  P_MESSAGE_REFERENCES,
  P_MESSAGE_SUBJECT,
  P_MESSAGE_COMMENTS,
  P_MESSAGE_KEYWORDS,
  P_MESSAGE_TRACE_FIELD,
  P_MESSAGE_RESENT_FIELD,
  P_MESSAGE_MIME_VERSION,
  P_MESSAGE_MIME_CONTENT,
  P_MESSAGE_MIME_ENCODING,
  P_MESSAGE_MIME_ID,
  P_MESSAGE_MIME_DESCRIPTION,
  P_MESSAGE_MIME_DISPOSITION,
  P_MESSAGE_MIME_EXTENSION_FIELD,
  P_MESSAGE_EXTENSION_FIELD,
  P_MESSAGE_PRIORITY
};


enum TpMailboxMessageStatus {

  P_MMM_RECEIVED_MSG_STATUS_READ,
  P_MMM_RECEIVED_MSG_STATUS_UNREAD,
  P_MMM_RECEIVED_MSG_STATUS_FORWARDED,
  P_MMM_RECEIVED_MSG_STATUS_REPLIED_TO,
  P_MMM_DRAFT_MSG_STATUS_SAVED_OR_UNSENT,
  P_MMM_SENT_MSG_STATUS_SENT,
  P_MMM_SENT_MSG_STATUS_DELIVERED,
  P_MMM_SENT_MSG_STATUS_READ,
  P_MMM_SENT_MSG_STATUS_DELETED_UNREAD,
  P_MMM_SENT_MSG_STATUS_NOT_DELIVERABLE,
  P_MMM_SENT_MSG_STATUS_EXPIRED
};


union TpMessageInfoProperty switch(TpMessageInfoPropertyName) {
  case P_MMM_MESSAGE_DATE_CREATED: TpDateAndTime MessageDateCreated;
  case P_MMM_MESSAGE_DATE_RECEIVED: TpDateAndTime MessageDateReceived;
  case P_MMM_MESSAGE_DATE_CHANGED: TpDateAndTime MessageDateChanged;
  case P_MMM_MESSAGE_SIZE: TpInt32 MessageSize;
  case P_MMM_MESSAGE_STATUS: TpMailboxMessageStatus MessageStatus;
  default: short Dummy;
};


typedef sequence <TpMessageInfoProperty> TpMessageInfoPropertySet;


struct TpGenericHeaderField {
  TpString FieldName;
  TpString FieldValue;
};


union TpMessageHeaderField switch(TpMessageHeaderFieldType) {
  case P_MESSAGE_DATE_SENT: TpDateAndTime DateSent;
  case P_MESSAGE_SENT_FROM: TpAddressSet From;
  case P_MESSAGE_SENDER: TpAddress Sender;
  case P_MESSAGE_REPLY_TO: TpAddressSet ReplyTo;
  case P_MESSAGE_SENT_TO: TpAddressSet To;
  case P_MESSAGE_CC_TO: TpAddressSet Cc;
  case P_MESSAGE_BCC_TO: TpAddressSet Bcc;
  case P_MESSAGE_RFC822_MESSAGE_ID: TpString RFC822MessageID;
  case P_MESSAGE_IN_REPLY_TO: TpStringSet InReplyTo;
```

```
          case P_MESSAGE_REFERENCES: TpStringSet References;
          case P_MESSAGE_SUBJECT: TpString Subject;
          case P_MESSAGE_COMMENTS: TpString Comments;
          case P_MESSAGE_KEYWORDS: TpStringSet Keywords;
          case P_MESSAGE_TRACE_FIELD: TpGenericHeaderField TraceField;
          case P_MESSAGE_RESENT_FIELD: TpGenericHeaderField ResentField;
          case P_MESSAGE_MIME_VERSION: TpString MimeVersion;
          case P_MESSAGE_MIME_CONTENT: TpString MimeContent;
          case P_MESSAGE_MIME_ENCODING: TpString MimeEncoding;
          case P_MESSAGE_MIME_ID: TpString MimeID;
          case P_MESSAGE_MIME_DESCRIPTION: TpString MimeDescription;
          case P_MESSAGE_MIME_DISPOSITION: TpString MimeDisposition;
          case P_MESSAGE_MIME_EXTENSION_FIELD: TpGenericHeaderField
MimeExtensionField;
          case P_MESSAGE_EXTENSION_FIELD: TpGenericHeaderField ExtensionField;
          case P_MESSAGE_PRIORITY: TpMessagePriority Priority;
        };


        typedef sequence <TpMessageHeaderField> TpMessageHeaderFieldSet;


        enum TpSetPropertyError {

          P_MMM_PROPERTY_NOT_SET,
          P_MMM_PROPERTY_READONLY,
          P_MMM_PROPERTY_INSUFFICIENT_PRIVILEGE,
          P_MMM_PROPERTY_NAME_UNKNOWN
        };


        struct TpMessageInfoPropertyError {
          TpMessageInfoPropertyName MessagePropertyName;
          TpSetPropertyError Error;
        };


        typedef sequence <TpMessageInfoPropertyError>
TpMessageInfoPropertyErrorSet;


        enum TpMessagingEventName {

          P_EVENT_MSG_NAME_UNDEFINED,
          P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED,
          P_EVENT_MSG_NEW_MESSAGE_ARRIVED
        };


        struct TpNewMailboxMessageArrivedCriteria {
          TpString MailboxID;
          TpString AuthenticationInfo;
        };


        struct TpNewMailboxMessageArrivedInfo {
          TpString MailboxID;
          TpString FolderID;
          TpMessageDescriptionList MessageDescription;
          TpMessageHeaderFieldSet ExtendedHeaderInformation;
        };
```

```
       typedef TpInt32 TpMessageDeliveryReportType;


       const TpMessageDeliveryReportType P_MESSAGE_REPORT_DELIVERY_UNDEFINED =
0;

       const TpMessageDeliveryReportType P_MESSAGE_REPORT_DELIVERED = 1;

       const TpMessageDeliveryReportType P_MESSAGE_REPORT_READ = 2;

       const TpMessageDeliveryReportType P_MESSAGE_REPORT_DELETED_UNREAD = 4;

       const TpMessageDeliveryReportType P_MESSAGE_REPORT_NOT_DELIVERABLE = 8;

       const TpMessageDeliveryReportType P_MESSAGE_REPORT_EXPIRED = 16;

       struct TpQueryStatusReport {
         TpAddress DestinationAddress;
         TpMessageDeliveryReportType ReportedStatus;
       };


       typedef sequence <TpQueryStatusReport> TpQueryStatusReportSet;


       enum TpDeliveryTimeType {

         P_MMM_SEND_IMMEDIATE,
         P_MMM_DELIVERY_TIME
       };


       union TpDeliveryTime switch(TpDeliveryTimeType) {
         case P_MMM_DELIVERY_TIME: TpDateAndTime DeliveryTime;
         default: short Dummy;
       };


       typedef TpString TpMessageDeliveryType;


       const TpMessageDeliveryType P_MMM_SMS = "P_MMM_SMS";

       const TpMessageDeliveryType P_MMM_SMS_BINARY = "P_MMM_SMS_BINARY";

       const TpMessageDeliveryType P_MMM_MMS = "P_MMM_MMS";

       const TpMessageDeliveryType P_MMM_WAP_PUSH = "P_MMM_WAP_PUSH";

       const TpMessageDeliveryType P_MMM_EMAIL = "P_MMM_EMAIL";

       enum TpMessageTreatmentType {

         P_MMM_TREATMENT_UNDEFINED,
         P_MMM_TREATMENT_REPORT_REQUESTED,
         P_MMM_TREATMENT_BILLING_ID,
         P_MMM_TREATMENT_DELIVERY_TIME,
         P_MMM_TREATMENT_VALIDITY_TIME
       };


       union TpMessageTreatment switch(TpMessageTreatmentType) {
         case P_MMM_TREATMENT_REPORT_REQUESTED: TpMessageDeliveryReportType 
DeliveryReport;
         case P_MMM_TREATMENT_BILLING_ID: TpString BillingID;
         case P_MMM_TREATMENT_DELIVERY_TIME: TpDeliveryTime DeliveryTime;
         case P_MMM_TREATMENT_VALIDITY_TIME: TpDateAndTime ValidityTime;
         default: short Dummy;
       };


       typedef sequence <TpMessageTreatment> TpMessageTreatmentSet;


       struct TpTerminatingAddressList {
         TpAddressSet ToAddressList;
         TpAddressSet CcAddressList;
         TpAddressSet BccAddressList;
       };


       exception P_MMM_MAX_MESSAGE_SIZE_EXCEEDED {
         TpString ExtraInformation;
       };


       exception P_MMM_INVALID_FOLDER_ID {
         TpString ExtraInformation;
       };


       exception P_MMM_INVALID_MESSAGE_ID {
         TpString ExtraInformation;
       };


       exception P_MMM_INVALID_PART_ID {
         TpString ExtraInformation;
       };


       exception P_MMM_DELIVERY_TYPE_ADDRESS_TYPE_MISMATCH {
         TpString ExtraInformation;
       };


       exception P_MMM_DELIVERY_TYPE_MESSAGE_TYPE_MISMATCH {
         TpString ExtraInformation;
       };


       exception P_MMM_INVALID_PROPERTY {
         TpString ExtraInformation;
       };


       enum TpMessagingError {

         P_MMM_ERROR_UNDEFINED,
         P_MMM_ERROR_INVALID_AUTHENTICATION_INFORMATION,
         P_MMM_ERROR_INVALID_MAILBOX,
         P_MMM_ERROR_INVALID_DELIVERY_TYPE,
         P_MMM_ERROR_MAX_MESSAGE_SIZE_EXCEEDED,
         P_MMM_ERROR_INVALID_FOLDER_ID,
         P_MMM_ERROR_INVALID_MESSAGE_ID,
         P_MMM_ERROR_INVALID_PART_ID,
         P_MMM_ERROR_DELIVERY_TYPE_ADDRESS_TYPE_MISMATCH,
         P_MMM_ERROR_DELIVERY_TYPE_MESSAGE_TYPE_MISMATCH,
         P_MMM_ERROR_INVALID_DELIVERY_TIME,
         P_MMM_ERROR_INVALID_VALIDITY_TIME,
         P_MMM_ERROR_MAX_SUBJECT_SIZE_EXCEEDED,
         P_MMM_ERROR_INVALID_ID,
         P_MMM_ERROR_INVALID_NESTING_LEVEL,
         P_MMM_ERROR_INVALID_CRITERIA,
         P_MMM_ERROR_INFORMATION_NOT_AVAILABLE,
         P_MMM_ERROR_CANNOT_CANCEL,
         P_MMM_ERROR_INVALID_HEADER,
         P_MMM_INVALID_NETWORK_STATE,
         P_MMM_ERROR_RESOURCE_UNAVAILABLE,
         P_MMM_ERROR_RESOURCE_TIMEOUT
       };


       exception P_MMM_INVALID_DELIVERY_TIME {
         TpString ExtraInformation;
       };


       exception P_MMM_INVALID_VALIDITY_TIME {
         TpString ExtraInformation;
       };


       exception P_MMM_MAX_SUBJECT_SIZE_EXCEEDED {
         TpString ExtraInformation;
       };


       exception P_MMM_INFORMATION_NOT_AVAILABLE {
         TpString ExtraInformation;
       };


       exception P_MMM_CANNOT_CANCEL {
         TpString ExtraInformation;
       };


       exception P_MMM_INVALID_HEADER {
         TpString ExtraInformation;
       };


       struct TpNewMessageArrivedCriteria {
         TpAddressRange SourceAddress;
         TpAddressRange DestinationAddress;
         TpBoolean CreateMultiMediaMessagingSession;
       };


       union TpMessagingEventCriteria switch(TpMessagingEventName) {
         case P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED:
TpNewMailboxMessageArrivedCriteria EventNewMailboxMessageArrived;
         case P_EVENT_MSG_NEW_MESSAGE_ARRIVED: TpNewMessageArrivedCriteria
EventNewMessageArrived;
         default: short Dummy;
       };
       typedef sequence <TpMessagingEventCriteria> TpMessagingEventCriteriaSet;


       struct TpMessagingNotificationRequested {
         TpMessagingEventCriteriaSet EventCriteria;
         TpInt32 AssignmentID;
       };


       typedef sequence <TpMessagingNotificationRequested>
TpMessagingNotificationRequestedSet;


       struct TpMessagingNotificationRequestedSetEntry {
         TpMessagingNotificationRequestedSet MessagingNotificationRequestedSet;
         TpBoolean Final;
       };


       struct TpNewMessageArrivedInfo {
         TpAddress SourceAddress;
         TpAddressSet DestinationAddressSet;
         TpOctetSet Message;
         TpMessageHeaderFieldSet Headers;
         TpMultiMediaMessagingIdentifier MultiMediaMessagingIdentifier;
       };


       union TpMessagingEventInfo switch(TpMessagingEventName) {
         case P_EVENT_MSG_NAME_UNDEFINED: TpString EventNameUndefined;
         case P_EVENT_MSG_NEW_MAILBOX_MESSAGE_ARRIVED:
TpNewMailboxMessageArrivedInfo EventNewMailboxMessageArrived;
         case P_EVENT_MSG_NEW_MESSAGE_ARRIVED: TpNewMessageArrivedInfo
EventNewMessageArrived;
       };


       typedef sequence <TpMessagingEventInfo> TpMessagingEventInfoSet;


       typedef sequence <TpMailboxIdentifier> TpMailboxIdentifierSet;


       typedef sequence <TpMultiMediaMessagingIdentifier>
TpMultiMediaMessagingIdentifierSet;


       interface IpAppMultiMediaMessagingManager : IpInterface {
         void mailboxTerminated (
           in TpMailboxIdentifier mailboxIdentifier
           );

         IpAppMultiMediaMessaging reportNotification (
           in TpAssignmentID assignmentID,
           in TpMessagingEventInfoSet eventInfo
           );

         void notificationsInterrupted ();

         void notificationsResumed ();

         void multiMediaMessagingTerminated (
           in TpMultiMediaMessagingIdentifier multimediaMessagingIdentifier
           );

         void terminateMultipleMailboxes (
           in TpMailboxIdentifierSet mailboxSet
           );

         void terminateMultipleMultiMediaMessagingSessions (
           in TpMultiMediaMessagingIdentifierSet multiMediaMessagingSet
           );

       };


       interface IpMultiMediaMessagingManager : IpService {
         TpMailboxIdentifier openMailbox (
           in TpString mailboxID,
           in TpString authenticationInfo,
           in IpAppMailbox appMailbox
           )
           raises
(TpCommonExceptions,P_MMM_INVALID_MAILBOX,P_MMM_INVALID_AUTHENTICATION_INFORMATI
ON,P_INVALID_INTERFACE_TYPE);

         TpMultiMediaMessagingIdentifier openMultiMediaMessaging (
           in TpTerminatingAddressList defaultDestinationAddressList,
           in TpAddress defaultSourceAddress,
           in IpAppMultiMediaMessaging appMultiMediaMessaging
           )
           raises
(TpCommonExceptions,P_INVALID_INTERFACE_TYPE,P_INVALID_ADDRESS);

         TpAssignmentID createNotification (
           in IpAppMultiMediaMessagingManager appMultiMediaMessagingManager,
           in TpMessagingEventCriteriaSet eventCriteria
           )
           raises
(TpCommonExceptions,P_INVALID_CRITERIA,P_INVALID_INTERFACE_TYPE);

         void destroyNotification (
           in TpAssignmentID assignmentID
           )
           raises (TpCommonExceptions,P_INVALID_ASSIGNMENT_ID);

         void changeNotification (
           in TpAssignmentID assignmentID,
           in TpMessagingEventCriteriaSet eventCriteria
           )
           raises
(TpCommonExceptions,P_INVALID_ASSIGNMENT_ID,P_INVALID_CRITERIA);

         TpMessagingNotificationRequestedSetEntry getNextNotification (
           in TpBoolean reset
           )
           raises (TpCommonExceptions);

         TpAssignmentID enableNotifications (
           in IpAppMultiMediaMessagingManager appMultiMediaMessagingManager
           )
           raises (TpCommonExceptions,P_INVALID_INTERFACE_TYPE);

         void disableNotifications ()
           raises (TpCommonExceptions);
```

```
};

interface IpAppMailbox : IpInterface {
  void createFolderRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString folderID
    );

  void createFolderErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getFoldersRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString folderID,
    in TpStringList folderNames
    );

  void getFoldersErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString folderID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void deleteFolderRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID
    );

  void deleteFolderErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void copyFolderRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID
    );

  void copyFolderErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void moveFolderRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID
    );

  void moveFolderErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

void putMessageRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString messageID
    );

void putMessageErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

void copyMessageRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID
    );

void copyMessageErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

void moveMessageRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID
    );

void moveMessageErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

void deleteMessageRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID
    );

void deleteMessageErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

void listMessagesRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessageDescriptionList messageList,
    in TpMailboxFolderStatusInformation mailboxStatusInfo,
    in TpBoolean final
    );

void listMessagesErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void listMessageBodyPartsRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpBodyPartDescriptionList partsList
    );

  void listMessageBodyPartsErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getMessageBodyPartsRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpBodyPartList bodyParts
    );

  void getMessageBodyPartsErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getMessageHeadersRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessageHeaderFieldSet headers
    );

  void getMessageHeadersErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getMessageContentRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString contentType,
    in TpString contentTransferEncoding,
    in TpOctetSet content
    );

  void getMessageContentErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getFullMessageRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpOctetSet message
    );

  void getFullMessageErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getMailboxInfoPropertiesRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMailboxInfoPropertySet returnedProperties
    );

  void getFolderInfoPropertiesRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString folderID,
    in TpFolderInfoPropertySet returnedProperties
    );

  void getMessageInfoPropertiesRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString messageID,
    in TpMessageInfoPropertySet returnedProperties
    );

  void setMessageInfoPropertiesRes (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString messageID,
    in TpMessageInfoPropertySet propertiesUpdated
    );

  void setMessageInfoPropertiesErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpString messageID,
    in TpMessageInfoPropertyErrorSet propertiesNotUpdated
    );

  void getMailboxInfoPropertiesErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getFolderInfoPropertiesErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
    );

  void getMessageInfoPropertiesErr (
    in TpSessionID mailboxSessionID,
    in TpAssignmentID requestID,
    in TpMessagingError error,
    in TpString errorDetails
```

```
          );

      };


      interface IpMailbox : IpService {
        void close (
          in TpSessionID mailboxSessionID
          )
          raises (TpCommonExceptions,P_INVALID_SESSION_ID);

        TpAssignmentID createFolderReq (
          in TpSessionID mailboxSessionID,
          in TpString folderID
          )
          raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID);

        TpAssignmentID getFoldersReq (
          in TpSessionID mailboxSessionID,
          in TpString folderID
          )
          raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID);

        TpAssignmentID deleteFolderReq (
          in TpSessionID mailboxSessionID,
          in TpString folderID
          )
          raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID);

        TpAssignmentID copyFolderReq (
          in TpSessionID mailboxSessionID,
          in TpString sourceFolderID,
          in TpString destinationFolderID
          )
          raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID);

        TpAssignmentID moveFolderReq (
          in TpSessionID mailboxSessionID,
          in TpString sourceFolderID,
          in TpString destinationFolderID
          )
          raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID);

        TpAssignmentID putMessageReq (
          in TpSessionID mailboxSessionID,
          in TpString folderID,
          in TpOctetSet message
          )
          raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_MAX_MESSA
GE_SIZE_EXCEEDED);

        TpAssignmentID copyMessageReq (
          in TpSessionID mailboxSessionID,
          in TpString fromFolderID,
          in TpString toFolderID,
          in TpString messageID
          )
```

```
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID moveMessageReq (
            in TpSessionID mailboxSessionID,
            in TpString fromFolderID,
            in TpString toFolderID,
            in TpString messageID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID deleteMessageReq (
            in TpSessionID mailboxSessionID,
            in TpString fromFolderID,
            in TpString messageID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID listMessagesReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID,
            in TpListMessagesCriteria criteria,
            in TpBoolean reset
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_INVALID_CRITE
RIA);

          TpAssignmentID listMessageBodyPartsReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID,
            in TpString messageID,
            in TpInt32 maxNestingLevel
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID getMessageBodyPartsReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID,
            in TpString messageID,
            in TpStringList partIDs
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID,P_MMM_INVALID_PART_ID);

          TpAssignmentID getMessageHeadersReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID,
            in TpString messageID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID getMessageContentReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID,
            in TpString messageID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID getFullMessageReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID,
            in TpString messageID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID,P_MMM_INVALID_M
ESSAGE_ID);

          TpAssignmentID getMailboxInfoPropertiesReq (
            in TpSessionID mailboxSessionID
            )
            raises (TpCommonExceptions,P_INVALID_SESSION_ID);

          TpAssignmentID getFolderInfoPropertiesReq (
            in TpSessionID mailboxSessionID,
            in TpString folderID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_FOLDER_ID);

          TpAssignmentID getMessageInfoPropertiesReq (
            in TpSessionID mailboxSessionID,
            in TpString messageID
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_MESSAGE_ID);

          TpAssignmentID setMessageInfoPropertiesReq (
            in TpSessionID mailboxSessionID,
            in TpString messageID,
            in TpMessageInfoPropertySet properties
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_MMM_INVALID_MESSAGE_ID,P_MMM_INVALID_
PROPERTY);

        };


        interface IpAppMultiMediaMessaging : IpInterface {
          void sendMessageRes (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID
            );

          void sendMessageErr (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID,
            in TpMessagingError error,
            in TpString errorDetails
            );

          void cancelMessageRes (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID
            );

          void cancelMessageErr (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID,
            in TpMessagingError error,
            in TpString errorDetails
            );

          void queryStatusRes (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID,
            in TpQueryStatusReportSet result
            );

          void queryStatusErr (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID,
            in TpMessagingError error,
            in TpString errorDetails
            );

          void messageStatusReport (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID,
            in TpAddress destinationAddress,
            in TpMessageDeliveryReportType deliveryReportType,
            in TpString deliveryReportInfo
            );

          void messageReceived (
            in TpSessionID sessionID,
            in TpOctetSet message,
            in TpMessageHeaderFieldSet headers
            );

        };


        interface IpMultiMediaMessaging : IpService {
          TpAssignmentID sendMessageReq (
            in TpSessionID sessionID,
            in TpAddress sourceAddress,
            in TpTerminatingAddressList destinationAddressList,
            in TpMessageDeliveryType deliveryType,
            in TpMessageTreatmentSet messageTreatment,
            in TpOctetSet message,
            in TpMessageHeaderFieldSet additionalHeaders
            )
            raises
(TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_ADDRESS,P_MMM_INVALID_DELIVER
Y_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_INV
ALID_VALIDITY_TIME,P_MMM_MAX_SUBJECT_SIZE_EXCEEDED,P_MMM_INVALID_HEADER);

          void cancelMessageReq (
            in TpSessionID sessionID,
            in TpAssignmentID assignmentID
            )
            raises (TpCommonExceptions, P_INVALID_SESSION_ID,
P_INVALID_ASSIGNMENT_ID,P_INVALID_NETWORK_STATE,P_MMM_CANNOT_CANCEL);
```

```
         void queryStatusReq (
           in TpSessionID sessionID,
           in TpAssignmentID assignmentID
           )
           raises (TpCommonExceptions, P_INVALID_SESSION_ID,
P_INVALID_ASSIGNMENT_ID,P_MMM_INFORMATION_NOT_AVAILABLE);

         void close (
           in TpSessionID sessionID
           )
           raises (TpCommonExceptions, P_INVALID_SESSION_ID);

       };

     };

   };

};

#endif
```

End of Change in Annex A

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2004	CN_25	NP-040359	--	--	Draft v100 submitted to TSG CN#25 for Approval.	1.0.0	6.0.0
Dec 2004	CN_26	NP-040485	001	--	Removal of OSA API SCFs description in W3C WSDL	6.0.0	6.1.0
Dec 2004	--	--	--	--	Added missing code attachments	6.1.0	6.1.1