Approved: Per Berggren	No: 4/FS-CRH 109 581-1 Uen

Copyright Mobeon AB All rights reserved	Author: Andreas Dekarö Marcus Haglund Title: FS - Mailbox	Version: PA1 Date: 2007-08-27	1/16

FS - Mailbox

Content

History

Version	Date	Adjustments
PA1	2007-08-27	First version of the document. Original document

		was found in MAS. (EMAHAGL)

1 Introduction

This document specifies the function of the Mailbox component in MAS. The Mailbox component provides an interface that can be used by other components in order to retrieve a subscriber Mailbox according to the system data model described in [1].

1.1 Glossary

1.1.1 Mailbox

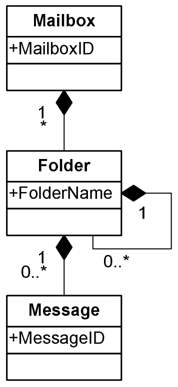

Figure 1 Mailbox

A subscriber's messages are stored in one or more Mailbox (see [1] for a detailed definition).

2 Function Requirements (Commercial)

This paragraph is intentionally left blank.

3 Function Specification (Design Related)

3.1 Introduction

The Mailbox component handles the access to subscriber mailboxes. Using the Mailbox component, clients can retrieve a subscriber's messages, print received messages and store new messages.

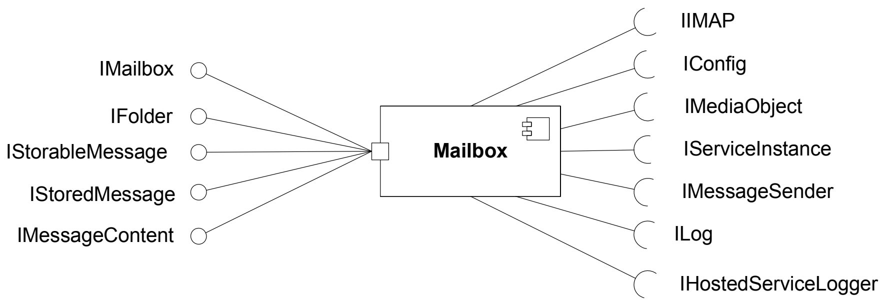

Figure 2 Mailbox component

3.2 Exported Interfaces

3.2.1 IMailbox

This interface is used to retrieve and manage a subscriber Mailbox.

3.2.1.1 Methods

IMailbox offers the following methods:

- *IMailbox IMailbox(IServiceInstance serviceInstance, String accountid, String accountPassword, String emailAddress)*
  Creates and returns the mailbox for a specific user/account. An email address is provided used for possible delivery reports.
- *IFolder getFolder(String folderName)*
  Returns the specified folder.
- *IFolder addFolder(String folderName)*
  Adds a new folder.
- *void deleteFolder(String folderName)*
  Deletes a folder.
- *int getByteUsage()*
  Returns the amount of bytes stored in the mailbox.
- *int getMessageUsage()*
  Returns the amount of messages stored in the mailbox.
- *void close()*
  Releases allocated mailbox resources. Deletes all messages having message state Deleted.

3.2.2 IFolder

This interface is used by a client to manage a Mailbox folder.

3.2.2.1 Methods

IFolder offers the following methods:

- *IStoredMessage[] getMessages(MessageFilter filter, SortFilter order)*
  Returns a list of the messages in the folder that matches the given filter.
- *IFolder getFolder(String folderName)*
  Returns the specified folder.
- *IFolder addFolder(String folderName)*
  Adds a new folder.
- *void deleteFolder(String folderName)*
  Deletes a folder.

3.2.2.2 Parameter Type Description

3.2.2.2.1 MessageFilter

A message filter is used to specify which message properties that are of interest when retrieving messages. The properties allowed in the filter are specified in chapter 3.4.8.2.

For example, to retrieve saved messages the message filter would contain information that the message state shall be set to *Saved*. If a property is left out it means that the message is retrieved whatever the message property's value, e.g. if the filter does not specify the urgent state, both urgent and non-urgent messages are retrieved.

3.2.2.2.2 SortFilter

The list is possible to sort based on the message properties, e.g. urgent and received date.

3.2.3 IStoredMessage

This interface is used by a client to access a stored message's properties.

3.2.3.1 Methods

IStoredMessage offers the following methods:

- *void set<Property name>(<Type> propertyValue)*
  Sets the specified property. The available writable properties are listed in chapter 3.4.8.2.
- *<Type> get<Property name>()*
  Returns the requested property. The available properties are listed in chapter 3.4.8.2.
- *Object getAdditionalProperty (String name)*
  Returns the requested additional property. The available additional properties are configured to map into the underlaying message implementation.
- *IMessageContent[] getContent()*
  Returns the available message contents.
- *IMediaObject getSpokenNameOfSender()*
  Returns the spoken name of the message sender.

- *void print (String destination, String sender)*
  Prints the message to the specified destination and sets a given sender on the printed message.
- *IStorableMessage forward ()*
  Creates a storable message with this message content attached. The resulting IStorableMessage must be constructed so that subsequent calls to IStorableMessage.addContent(*content*) inserts content before the forward-message content.
- *void renewIssuedDate ()*
  Renews date when message was "issued" to now. Used to extend the message retention time.
- *void copy(IFolder folder)*
  Copies this message to the specified folder.
- *void saveChanges()*
  Permanently saves any changes to the message store.

3.2.3.2 Parameter Type Description

3.2.3.2.1 IMediaObject

An IMediaObject is the actual media that can be played or recorded in the system. The *IMediaObject* is an imported interface which is specified in [3].

3.2.4 IStorableMessage

This interface is used by a client to access a new message's properties.

3.2.4.1 Methods

IStorableMessage offers the following methods:

- *void set<Property name>(Object propertyValue)*
  Sets the specified property. The available properties are listed in chapter 3.4.8.1.
- *Object get<Property name>()*
  Returns the requested property. The available properties are listed in chapter 3.4.8.1.
- *void setAdditionalProperty (String name, Object value)*
  Sets the specified additional property. The available additional properties are configured to map into the underlaying message implementation.
- *Object getAdditionalProperty (String name)*
  Returns the requested additional property. The available additional properties are configured to map into the underlaying message implementation.
- *void addContent(IMediaObject mediaObject, ContentProperties props)*
  Adds content to the message. Content properties should be provided with the media content.
- *IMessageContent[] getContent()*
  Returns the available message contents.

- *void setSpokenNameOfSender(IMediaObject spokenName, String filename)*
  Sets the spoken name of the sender. Shall be used when creating a new message to store. A filename should be provided with the media content.
- *IMediaObject getSpokenNameOfSender()*
  Returns the spoken name of the message sender.
- *void store (String host)*
  Stores the message in recipients mailbox at a given host.

  void store ()
  Stores the message in recipients mailbox (preferred host will be used).

3.2.4.2 Parameter Type Description

3.2.4.2.1 IMediaObject

An IMediaObject is the actual media that can be played or recorded in the system. The *IMediaObject* is an imported interface which is specified in [3].

3.2.4.2.2 ContentProperties

ContentProperties is a set of key/value pairs to be stored with the media content. Supported keys are *description*, *language* and *filename*.

3.2.5 IMessageContent

The interface is used for representing message content. The interface adds semantics to the Media Object.

3.2.5.1 Methods

IMessageContent offers the following methods:

- *MediaProperties getMediaProperties()*
  Returns the media properties for the current content. *MediaProperties* is defined in [3].
- *IMediaObject getContent()*
  Returns the media object contained in the message content.

  ContentProperties getContentProperties()
  Returns the media content properties.

3.3 Imported Interfaces

The Mailbox Component uses the following external interfaces:

- *ILog* for logging.
- *IHostedServiceLogger* for logging hosted service circumstances.
- *IConfig* for configuration.
- *IMediaObject* for passing media objects such as a spoken name or message content.
- *IServiceInstance* for accessing service properties.

- *IMessageSender* for sending an email when storing a new message.
- *IIMAP for email store management.*

3.4 Functions

3.4.1 Get Messages

The Mailbox component can be used to retrieve subscriber messages. Figure 3 exemplifies how messages are retrieved.

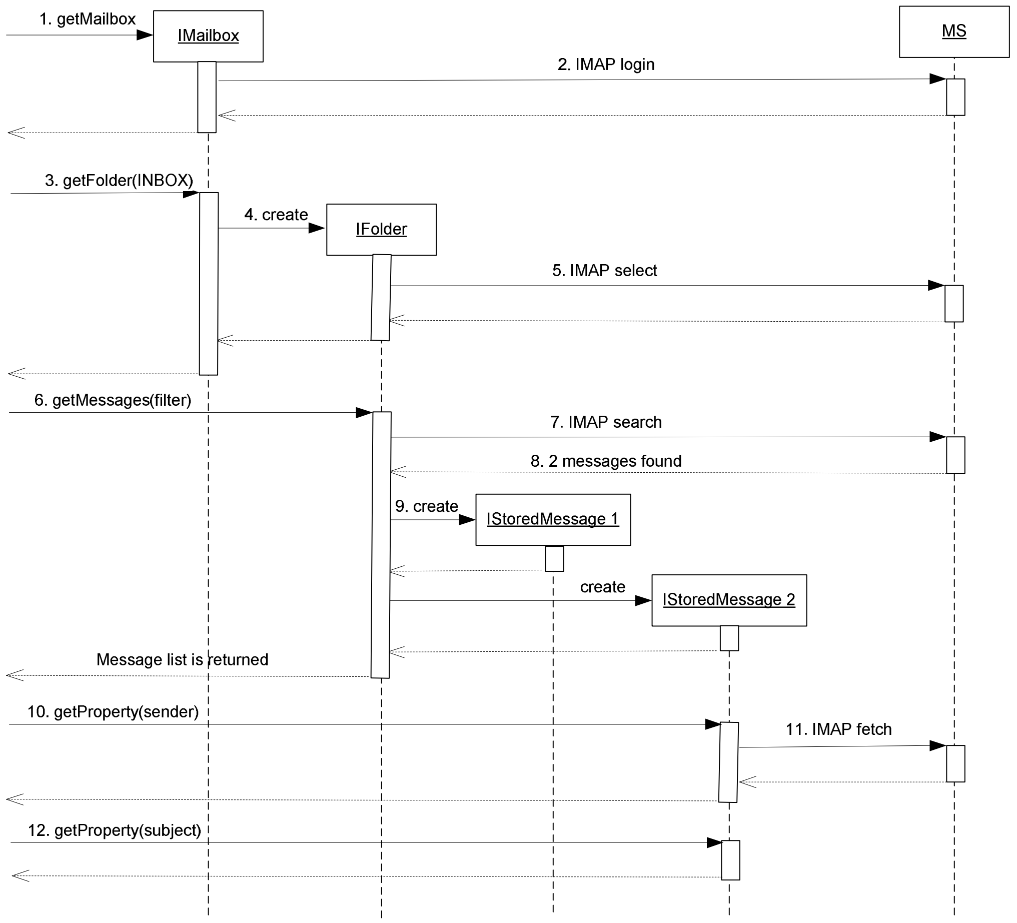

Figure 3 Example of message retrieval

1. A Mailbox instance is requested.
2. The mailbox connects to the mail host.

3. The Inbox folder is requested.

4. A Folder instance is created.

5. The folder selects the inbox folder.

6. A list of messages, filtered by *filter,* is requested.

7. The folder instance searches the message store for messages based on *filter.*

8. 2 messages are found and returned by the search.

9. 2 message instances are created. These instances will only contain information received from the search (lazy retrieval, see below).

10. The sender property of message 2 is requested.

11. All message properties are retrieved from the message store.

12. The subject property of message 2 is requested. This has been cached from the previous fetch and will return directly.

3.4.1.1 Lazy Retrieval

The Mailbox component tries to minimize the amount of IMAP calls and therefore a message is not fetched until first needed, i.e. lazy retrieval. One example of this is illustrated in Figure 3; the message properties (i.e. the message header) are not fetched over IMAP until first requested. But when requested, all header properties are fetched at once. The same applies for the message body; it is not fetched until first requested.

3.4.1.2 Sorting Messages

The interface provides abilities to sort the retrieved message list on property columns (see 3.4.8.2) - ascending or descending. If a property column is used the message header will be fetched, since IMAP not provides server-side sorting. If only *ReceivedDate* sorting is requested the implementation should be configurable to use IMAP-Message number instead of the Received-Date header to avoid the fetch of headers.

If the column is of type String rows should be sorted lexicographically.

If the column is of type Boolean then rows with column value true should be considered as “bigger” than rows with column value false. E.g. if message sorted ascending on a Boolean column false value rows comes first.

If the column is of type Date rows should be sorted in natural Date order. E.g. if message sorted ascending on a Date column older value rows comes first.

If the column is of type Enum rows should be sorted according to the Enum’s definition ordinal number order.

If the column is of type Number rows should be sorted in natural number order.

3.4.1.3 Cache

The messages retrieved from MS are cached during the lifetime of a Mailbox instance.

3.4.1.4 Copy a Message

A retrieved message can be copied to another folder using the *copy* method (see 3.2.3.1).

3.4.1.5 IMAP Encapsulation

All IMAP communication should be encapsulated and abstracted so it's possible to refine or even replace the IMAP support.

3.4.1.6 IMAP failure

When communicating with the hosted IMAP service the service might not respond to a request. If the hosted IMAP service not responds to the request the Mailbox caller is notified that the request failed. IMAP connection failures should be logged as described in ref [5] using the IHostedServiceLogger interface.

3.4.2 Store a Message

The Mailbox component can be used to store new subscriber messages. Figure 4 exemplifies how a message is stored.

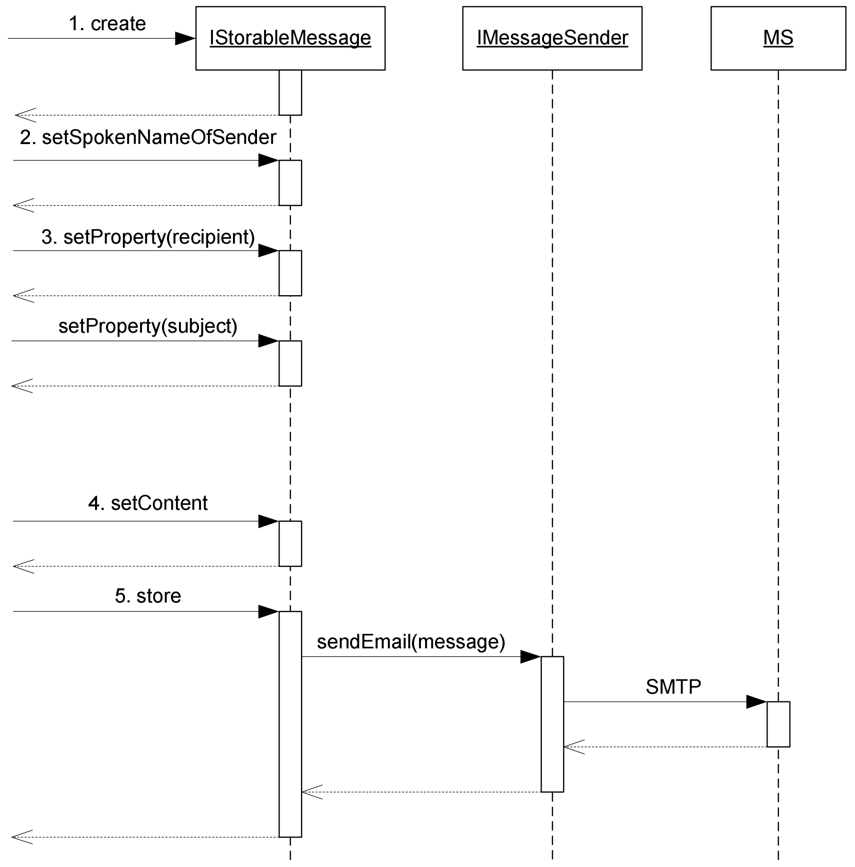

Figure 4 Example of storing a message

1. A new storable message is created.

2. The spoken name of the sender is set.

3. The properties of the message are set.

4. The content of the message is set.

5. The message is stored in the receiver mailbox.

3.4.2.1 Deferred Delivery

It is possible to store a message that shall not be sent to the recipient(s) immediately but at a specified date (i.e. future delivery or deferred delivery). This is done by specifying the message property *DeliveryDate* (see 3.4.8.1) before the message is stored. If no date is set, the message is delivered immediately.

3.4.2.2 Send failure

If Message Sender signals that it failed to send the email, the storable message should notify the caller of the store method about the failure without trying to resend the message.

3.4.3 Forward a Message

A retrieved message (see 3.4.1) can be forwarded to another recipient. This is done using the method *forward* (see 3.2.3.1).

3.4.4 Print a Message

The Mailbox component can be used to print subscriber messages.

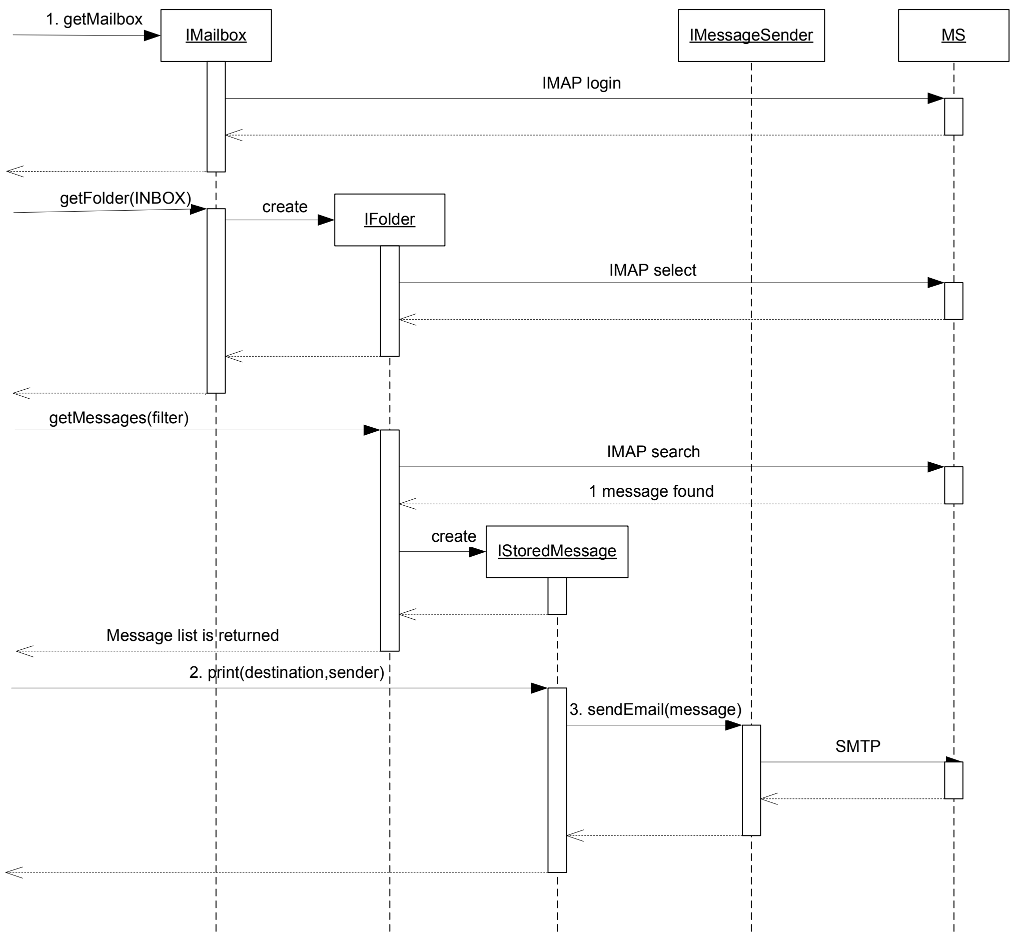

Figure 5 Example of printing a message

1. Messages are retrieved as described in 3.4.1.
2. The message is printed.

3. The message is printed by sending it as an email to MS which is responsible to print the message. The external interface *IMessageSender* is used for this purpose.

3.4.4.1 Send failure

If Message Sender signals that it failed to send the email, the stored message should notify the caller of the print method about the failure without trying to resend the message.

3.4.5 Delete a Message

To delete a Message the state is set to deleted and when mailbox is closed a command will be sent for storage deletion. See Figure 6.

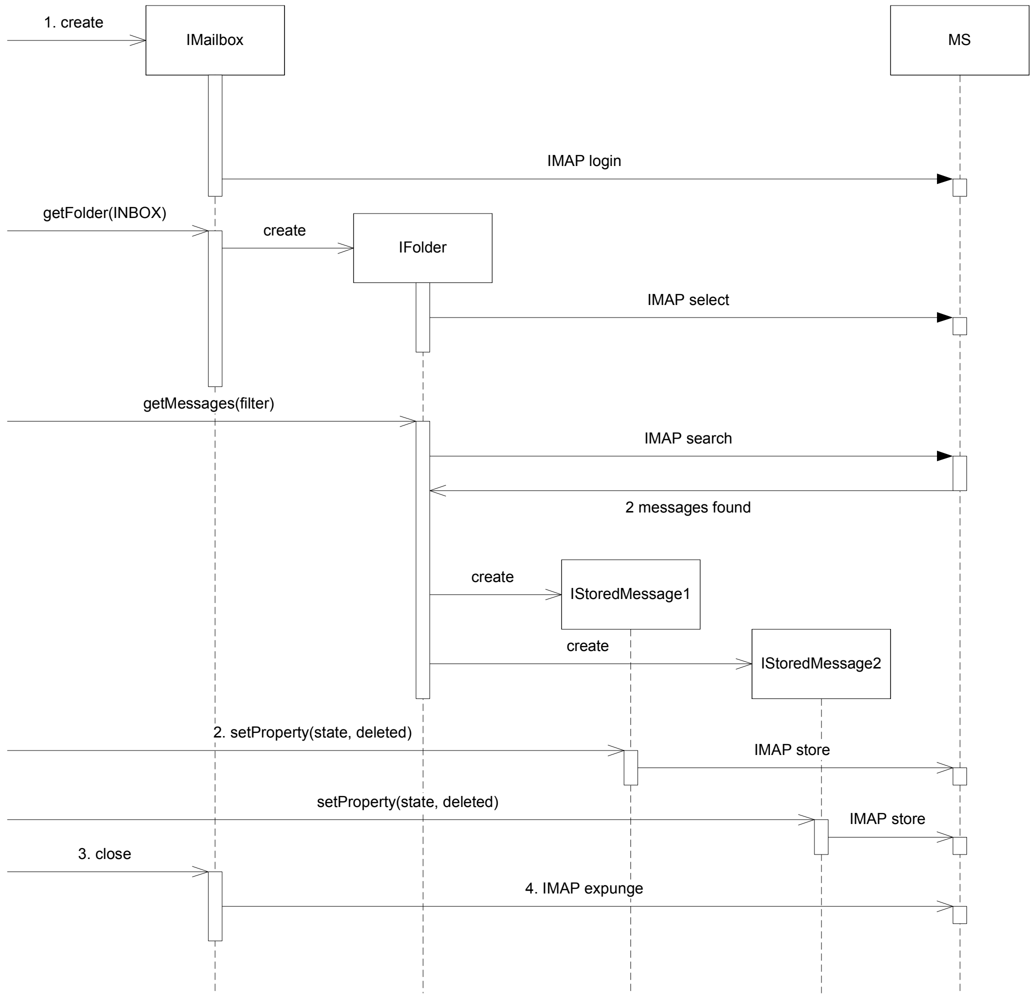

Figure 6 Example of deleting messages

1. Messages are retrieved as described in 3.4.1.

2. Messages to be deleted are set to state deleted.

3. The mailbox is closed.

4. The expunge command is sent to the storage.

3.4.6 Get Mailbox Usage

Two methods are available for retrieving information about mailbox usage:

- *getByteUsage*
- *getMessageUsage*.

3.4.7 Managing Folders

Nested folders are supported by the Mailbox component. Folders can be added and deleted using methods *addFolder* and *deleteFolder* in the *IMailbox* and *IFolder* interfaces.

3.4.8 Managing Messages

The messages follow the M3 message format specified in [2].

3.4.8.1 Available IStorableMessage Properties

Table 1 lists the properties available to use when setting or retrieving a storable message property.

Table 1 StorableMessage Properties

Property Name	Type	Valid values	Description
Sender	String	Any string conforming to [2]	The sender of the message.
Recipients	String[]	A list of text strings	The recipients of the message.
SecondaryRecipients	String[]	A list of text strings	The secondary recipients of the message.
Subject	String	Any text string	The subject of the message.
ReplyToAddr	String	Any text string	The address a reply of the message shall be sent to.
Type	Enum	"voice", "video", "fax" or "email"	The message type.
Language	String	Any valid language according to RFC2798	The language of the message.
DeliveryDate	Date	Any date conforming to [4].	The date and time when the receiver shall receive or has received the message.
Urgent	Boolean	True or false	Indicates if message is urgent.
Confidential	Boolean	True or false	Indicates if message is confidential.

3.4.8.2 *Available IStoredMessage Properties*

Table 2 lists the properties available to use in the MessageFilter or when setting or retrieving a stored message property.

Table 2 IStoredMessage Properties

Property Name	Type	Valid values	Description	R/W	Sortable
Sender	String	Any string conforming to [2]	The sender of the message.	R	Y
Recipients	String[]	A list of text strings	The recipients of the message.	R	N
SecondaryRecipients	String[]	A list of text strings	The secondary recipients of the message.	R	N
Subject	String	Any text string	The subject of the message.	R	Y
ReplyToAddr	String	Any text string	The address a reply of the message shall be sent to.	R	N
Type	String	"voice", "video", "fax" or "email"	The message type.	R	Y
State	Enum	"new", "read", "saved", "deleted"	The message state.	RW	Y
Language	Enum	Any valid language according to RFC2798	The language of the message.	R	N
ReceivedDate	Date	Any date conforming to [4]	The date and time when the receiver shall receive or has received the message.	R	Y
Urgent	Boolean	True or false	Indicates if message is urgent.	R	Y
Confidential	Boolean	True or false	Indicates if message is confidential.	R	Y
Forwarded	Boolean	True or false	Indicates if message is forwarded.	R	N
DeliveryReport	Boolean	True or false	Indicates if message is a delivery report.	R	N
DeliveryStatus	Enum	”store-failed”, ”print-failed”	If message is a delivery report this property indicates if message was undeliverable due store or print, otherwise N/A.	R	N

3.4.8.3 Spoken Name of Sender

It is possible to access the spoken name of the message sender using the method *getSpokenNameOfSender*. For a new message that shall be stored, the spoken name can be set using corresponding method: *setSpokenNameOfSender*.

3.4.8.4 Additional Message Properties

I addition to the supported message properties in 3.4.8 new message properties may be introduced without modifying the Mailbox implementation.

3.4.9 Thread Safe

Shared data within the Mailbox component is protected for use among several instances of for example *IMailbox*. But the Mailbox component is not thread safe in other aspects; one Mailbox instance is intended to be used by one thread only. The reason for this is that the behavior would be much unexpected if several threads could add and delete messages and folders in parallel in the same mailbox.

4 External Operation Conditions

4.1 Configuration

The following are configurable in Mailbox:

- Timeout of IMAP commands

4.2 Logging

- Hosted service circumstances should be logged as described in ref [5] using the IHostedServiceLogger interface.
- All calls to interface methods are logged at info level. Method name and parameter values are logged. At return is returned value logged.

5 Capabilities

This paragraph is intentionally left blank.

6 References

[1] TR - FS M3 Data Model
6/0363-PRO049 Uen

[2] IWD End User Message Format
3/155 19-HDB 101 02 Uen

[3] FS Media Object
8/FS-CRH 109 581-1 Uen

[4] RFC 1123
http://www.ietf.org/rfc/rfc1123.txt

[5] FS Log Manager
2/FS-CRH 109 581-1 Uen

7 Terminology

MS	Message Store