Inter-Office Memorandum

To	Distribution	Date	9 August 77
From	Jim White	Location	Palo Alto
Subject	**Mail Update to Mesa FTP Specification**	Organization	SDD/SD/CS

XEROX

Filed on: <White>UFTPPackage.Ears

Introduction

This memo updates the interface specification for the mail portion of the Mesa FTP Package proposed in the author's memo, "Mesa FTP Specification", dated 15 June 77. This new interface specification corresponds to the new Mail Transfer Protocol specification contained in Ed Taft's memo, "Pup Mail Transfer Protocol (Edition 3)", dated 13 July 77.

Already Proposed Support Procedures

The mail application requires certain general purpose program management, connection management, and file access procedures already proposed for FTP. These procedures are enumerated below; the reader is referred to the complete Mesa FTP specification for a detailed description of their use:

```
FTPMakeUser: PROCEDURE RETURNS [ftpuser: FTPUser];

FTPOpenConnection: PROCEDURE [ftpuser: FTPUser, host: STRING, purpose: Purpose];

    Purpose: TYPE = {files, mail, filesandmail};

FTPSetAccessibleDirectory: PROCEDURE [ftpuser: FTPUser, directoryId: DirectoryId, directory,
  password: STRING];

    DirectoryId: TYPE = {primary, secondary};

FTPCloseConnection: PROCEDURE [ftpuser: FTPUser];

FTPDestroyUser: PROCEDURE [ftpuser: FTPUser];
```

Newly Proposed Mail Procedures

The FTP Package provides procedures for delivering or forwarding mail to and extracting mail from remote mailboxes. Rather than signalling the presence of exceptional conditions, each of these procedures returns a numeric error code as one of its results. (Note that the support procedures described above, on the other hand, signal when exceptional conditions arise.) The following error codes are currently defined:

```
MailErrorCode: TYPE = {ok, noValidRecipients, noDiskSpace, noSuchMailbox, accessDenied,
noMoreMessages, noMoreBlocks};
```

Mail Delivery

The FTP Package provides three procedures for delivering or forwarding mail to remote mailboxes. The use of these procedures is illustrated by the following slice of Mesa code:

```
mailbox1 ← [next: @mailbox2, mailboxName: "Wegbreit", mailboxHostName: "Maxc2", dmsName:
"Wegbreit.Palo Alto", errorCode: ok, errorMessage: NIL];
mailbox2 ← [next: NIL, mailboxName: "Brotz", mailboxHostName: NIL, dmsName: NIL, errorCode: ok,
errorMessage: NIL];
errorCode ← FTPBeginDeliveryOfMessage[ftpuser, @mailbox1, AllocateHeapString];
IF errorCode = ok THEN errorCode ← FTPSendBlockOfMessage[ftpuser,
  LOOPHOLE[messageHeader, POINTER]+2, messageHeader.length];
IF errorCode = ok THEN errorCode ← FTPSendBlockOfMessage[ftpuser,
  LOOPHOLE[messageBody, POINTER]+2, messageBody.length];
IF errorCode = ok THEN errorCode ← FTPEndDeliveryOfMessage[ftpuser];
mailbox ← @mailbox1;
WHILE mailbox # NIL DO
  IF mailbox.errorCode # ok THEN SIGNAL MailboxException[mailbox];
  mailbox ← mailbox.next;
  ENDLOOP;
IF errorCode # ok THEN ERROR DeliveryError[errorCode];
```

The first procedure, *FTPBeginDeliveryOfMessage*, initiates the delivery and/or forwarding of a message by enumerating its intended recipients via a linked list, *mailboxList*. In the simpler case, called *delivery*, in which a recipient's mailbox resides on the connected host (a case which the procedure distinguishes by finding *mailboxHostName* set to NIL), the corresponding list element need contain only a pointer, *next*, to the next element in the list (NIL signalling the end of the list) and the host-specific name, *mailboxName*, of the remote mailbox to which a copy of the message is to be appended. In the more complex case, called *forwarding*, in which a recipient's mailbox resides on a third host (a case which not all hosts will support), the corresponding list element must also contain the name, *mailboxHostName*, of the target host and (optionally) the full *dmsName* of the target mailbox (which the forwarder may be able to use to locate the recipient if he is found to have moved):

```
FTPBeginDeliveryOfMessage: PROCEDURE [ftpuser: FTPUser, mailboxList: MailboxPtr,
allocateString: PROCEDURE [INTEGER] RETURNS [STRING]] RETURNS [mailErrorCode:
MailErrorCode];

    MailboxPtr: TYPE = POINTER TO Mailbox;

    Mailbox: TYPE = RECORD [next: MailboxPtr, mailboxName, mailboxHostName, dmsName:
  STRING, errorCode: ErrorCode, errorMessage: STRING];

    ErrorCode: TYPE = {ok, noSuchMailbox, noSuchMailboxHost, noSuchDmsName,
  noForwardingProvided, unspecifiedTransientError, unspecifiedPermanentError,
  unspecifiedError};
```

Delivery of the message succeeds or fails for each of its intended recipients independently. Either *FTPBeginDeliveryOfMessage* or the *FTPEndDeliveryOfMessage* procedure described below may report the failure of an individual delivery attempt by depositing in the appropriate list element a numeric *errorCode* intended for examination by the client (*ok* signalling successful delivery, but only tentatively until *FTPEndDeliveryOfMessage* has returned) and, if *errorCode* is one of the three having the form *unspecified...Error*, a textual *errorMessage* intended for examination by a human user. Storage for any error messages that may be returned is allocated via the *allocateString* procedure provided by the client, which assumes responsibility for releasing the storage.

The second procedure, *FTPSendBlockOfMessage,* specifies a portion of the text of the message and is called repetitively once the message's recipients have been identified via *FTPBeginDeliveryOfMessage.* Successive calls specify the location in the client's address space, *source,* and the length in bytes, *byteCount,* of successive blocks of text. The text of the message must include a message header conforming to ARPANET standards, the current unofficial standard being set forth in RFC 680, "Message Transmission Protocol", dated 15 May 75. Throughout the message, end of line is indicated via a carriage return (CR):

```
FTPSendBlockOfMessage: PROCEDURE [ftpuser: FTPUser, source: POINTER, byteCount:
CARDINAL] RETURNS [mailErrorCode: MailErrorCode];
```

The third procedure, *FTPEndDeliveryOfMessage,* signals the end of the sequence of calls to *FTPSendBlockOfMessage* and, therefore, of the message's text, and effects the message's delivery and/or enqueues the message for forwarding:

```
FTPEndDeliveryOfMessage: PROCEDURE [ftpuser: FTPUser] RETURNS [mailErrorCode:
MailErrorCode];
```

Like the *FTPBeginDeliveryOfMessage* procedure already described, *FTPEndDeliveryOfMessage* reports its failure to deliver the message to one of its intended recipients by depositing in the corresponding element of the recipient list supplied to *FTPBeginDeliveryOfMessage,* a numeric *errorCode* intended for examination by the client (*ok* here signalling successful delivery with finality) and, if *errorCode* is one of the three having the form *unspecified...Error,* a textual *errorMessage* intended for examination by a human user. Storage for any error messages that may be returned is again allocated via the *allocateString* procedure provided by the client, which assumes responsibility for releasing the storage.

Mail Retrieval

The FTP Package provides four procedures for emptying a remote mailbox. The use of these procedures is illustrated by the following slice of Mesa code:

```
errorCode ← FTPBeginRetrievalOfMessages[ftpuser, "Brotz"];
IF errorCode # ok THEN ERROR RetrievalError[errorCode];
UNTIL errorCode = noMoreMessages DO
  errorCode ← FTPIdentifyNextMessage[ftpuser, @messageInfo];
  SELECT errorCode FROM
    ok =>
      BEGIN -- begin processing of new message
      UNTIL errorCode = noMoreBlocks DO
        [errorCode, block.length] ← FTPRetrieveBlockOfMessage[ftpuser,
          LOOPHOLE[block, POINTER]+ 2, block.maxlength];
        SELECT errorCode FROM
          ok =>
            BEGIN -- begin processing of new block
            END; -- complete processing of new block
          noMoreBlocks => NULL;
          ENDCASE => ERROR RetrievalError[errorCode];
        ENDLOOP;
      END; -- complete processing of new message
    noMoreMessages => NULL;
    ENDCASE => ERROR RetrievalError[errorCode];
  ENDLOOP;
errorCode ← FTPEndRetrievalOfMessages[ftpuser];
IF errorCode # ok THEN ERROR RetrievalError[errorCode];
```

The first procedure, *FTPBeginRetrievalOfMessages,* initiates retrieval of the contents of the remote mailbox whose host-specific name, *mailboxName,* is specified:

```
FTPBeginRetrievalOfMessages: PROCEDURE [ftpuser: FTPUser, mailboxName: STRING] RETURNS
[mailErrorCode: MailErrorCode];
```

The second procedure, *FTPIdentifyNextMessage*, retrieves information about one of the messages in the mailbox specified in the previous call to *FTPBeginRetrievalOfMessages*. This procedure is called repetitively until the error code, *noMoreMessages*, is returned. Successive calls return information about successive messages stored in the mailbox. (The client may elect to leave some or all of the mailbox's contents unretrieved, in which case whatever remains will be sent by the remote FTP Server but discarded by the local FTP User in the final call to *FTPEndRetrievalOfMessages*.) The information returned by the procedure is deposited in a record, *messageInfo*, supplied by the client, and consists of the messsage's size in bytes, *byteCount*; the date and time, *deliveryDate*, at which the message was deposited in the mailbox (the required STRING being supplied by the client); and whether or not the message has been *opened* (i.e. examined) or *deleted* while in the mailbox (a possibility only for Maxc mailboxes, which can be manipulated directly via the MSG subsystem):

```
FTPIdentifyNextMessage: PROCEDURE [ftpuser: FTPUser, messageInfo: POINTER TO MessageInfo
] RETURNS [mailErrorCode: MailErrorCode];

    MessageInfo: TYPE = RECORD [byteCount: CARDINAL, deliveryDate: STRING, opened,
    deleted: BOOLEAN];
```

The third procedure, *FTPRetrieveBlockOfMessage*, retrieves a portion of the text of the message identified by the previous call to *FTPIdentifyNext Message*. This procedure is called repetitively until the error code, *noMoreBlocks*, is returned. Successive calls return successive blocks of the message. (The client may elect to leave some or all of the message's text unretrieved, in which case whatever remains will be sent by the remote FTP Server but discarded by the local FTP User in the next call to *FTPIdentifyNextMessage*.) (Note that the client can anticipate *noMoreBlocks* on the basis of the byte count returned by *FTPIdentifyNextMessage*.) The text returned by the procedure is deposited in the buffer whose location in the client's address space, *destination*, and whose length in bytes, *maxByteCount*, are specified by the client. The procedure returns the length in bytes, *actualByteCount*, of the block of text actually retrieved (which may be shorter than the block requested). The text of the message includes a message header conforming to ARPANET standards, the current unofficial standard being set forth in RFC 680, "Message Transmission Protocol", dated 15 May 75. Throughout the message, end of line is indicated via a carriage return (CR):

```
FTPRetrieveBlockOfMessage: PROCEDURE [ftpuser: FTPUser, destination: POINTER,
maxByteCount: CARDINAL] RETURNS [mailErrorCode: MailErrorCode, actualByteCount: CARDINAL];
```

The fourth procedure, *FTPEndRetrievalOfMessages*, terminates the retrieval operation and resets the mailbox to empty. *FTPBeginRetrievalOfMessages* and *FTPEndRetrievalOfMessages* are implemented in such a way that no new messages are lost during the retrieval transaction:

```
FTPEndRetrievalOfMessages: PROCEDURE [ftpuser: FTPUser] RETURNS [mailErrorCode:
MailErrorCode];
```

Distribution:

David Boggs
Roger Needham
Ed Satterthwaite
Mike Schroeder
Dan Swinehart
Ed Taft
Ben Wegbreit

SDD/SD/CS