To: Distribution

From: Steve Herbst
Gary Palter
Olin Sibert

Subject: Installing read_mail and send_mail

Date: 25 July 1978

The read_mail command has been under intensive development at MIT for several years. It has won popularity as a means of selectively reading, deleting, saving and forwarding messages. Honeywell customers have lately been asking for all of these features, and want to see a version of read_mail installed in the standard system.

The widely-used read_mail command, available at MIT as >udd>PDOI>ReadMail>read_mail, was developed by Ken Pogran, Charlie Davis and Roy Planalp. Other mail reading commands that have gone into the proposed design are John Klensin's check_mail at MIT and Jim Falksen's mail_read at Phoenix. All of these process messages sent by the installed mail command.

An important feature of each of these mail-reading programs is the ability to peruse message text. Headers have been inserted in the message to identify the subject (for quick perusal), the original sender and date-time sent (preserving them when the message is saved in another mailbox), and the various recipients (for tracing correspondence).

There is a companion program to read_mail called send_mail. It was developed at MIT by Doug Wells and is available as >udd>PDO>Palter>library>o>send_mail. This command automatically generates header fields that read_mail understands. It also allows editing of the message at any time before transmission. The read_mail command calls it to forward messages and send replies.

Together, read_mail and send_mail form a compatible and flexible mail subsystem. The useful features they provide are invisible to novice users but are easily activated.

In summary, these features are:

1. Selective printing, listing, and deleting of messages by number, author, date, subject, and substrings of the message text.

2. Selective saving of messages in specified mailboxes and ASCII segments, with the original author and date-time preserved in a header.

3. Forwarding and replying to messages.

4. Editing of messages both before transmission and after saving.

5. Sending to a list of addresses contained in one or more ASCII segments.

6. Sending a message, as an afterthought, to additional destinations not specified on the command line.

The rest of this document contains a summary description of read_mail and send_mail intended for the Introductory User's Guide, an MPM document on each command, and an appendix on ARPA Network mail.

Certain of the features described here will not appear in the initial release. These are the & and ! logical connectors in message specifiers and the majority of request functions. The only request function that will work in the initial release is execute (e).

In the future, mail addresses not otherwise recognized will be looked up in a system-wide mail table. This table will contain default projects for registered persons, default mailbox pathnames, and aliases that users establish for themselves. The mail table will be a secure database in which users can read selected fields of all records but can change only their own records.

## Introduction

The Multics mail system consists of two commands, read_mail and send_mail. Their calling sequences and internal request syntax are very similar.

## USING read_mail

A typical use of read_mail is to read your own mail. Type "read_mail" or "rdm" at command level and you will either see "You have no mail." or see "You have N messages." and be prompted for a read_mail request by the string "read_mail:".

To read mail in another user's mailbox, type "read_mail PERSON.PROJECT" or "read_mail PATHNAME". You can save messages in auxiliary mailboxes, for example, and read them by specifying pathnames to read_mail.

Several of the common read_mail requests are list, print, delete, log and quit.

The list request lists all the messages in a summary with one line per message. The summary line contains the message number, used in specifying a particular message to read_mail, the date it was sent, who it was sent by, and the subject of the message if the sender specified one.

The print, delete, and log requests all take the same set of arguments. If no arguments are specified, they operate on the current message, which is automatically set to 1 upon entry to read_mail. They accept message numbers, keywords, and match strings, for example: "print 1 2 3", "delete all", and "log /paycheck/". The print request prints messages, the delete request deletes them, and the log request saves them in a mailbox.

The mailbox used by the log request is called your log box. It has the name PERSON.sv.mbx, where PERSON is your registered Person identifier, and resides in your home directory. Since the log box is different from your mailbox, you will normally not see logged messages when reading mail.

The command line "read_mail -log" reads messages that have been saved by the log request. This command line behaves like "read_mail" but operates on your log box instead of on your default mailbox.

The quit request exits read_mail and returns to command level. Before exiting, it completes any changes to the mailbox that were made by read_mail requests, for example, deleting messages.

A typical dialogue with read_mail follows. The user's input is preceded by a ! character.

```
! read_mail
You have 4 messages.

read_mail: ! list

   #  Lines  Date     Time   From                Subject
   1.   (3)  07/17/78 13:48 Perkins.Kingdom     birds
   2.   (1)  07/17/78 20:18 Kojak.NYPD          Come clean I
   3.   (2)  07/18/78 09:23 Boop.Bedoop
   4.   (1)  07/18/78 11:51 Holmes.YARD         Don't get ex

read_mail: ! print

   1.   (3)  07/17/78 13:48 Perkins.Kingdom     birds

I found these two Left-Hinged Shrieking Cranes on my front
step this morning and don't know what to do with them.
Want some birds?
----------

read_mail: ! delete; print all

   2.   (1)  07/17/78 20:18 Kojak.NYPD

Come clean I know you're in there. Where's start_dump.pl1?
----------

   3.   (2)  07/18/78 09:23 Boop.Bedoop

Boody ooby doodoop scoop rezoolyada zoop yoodoo,
bleeby shooby oogie bloop boobyop oop!
----------

   4.   (1)  07/18/78 11:51 Holmes.YARD

Don't get excited. Kojak doesn't know what he's talking about.
----------

read_mail: ! log 3; delete all

All messages have been deleted.

read_mail: ! quit

r 1034 1.342 4.198 153
```

In this example, the user invokes read_mail, gets a summary listing the mailbox contents, prints all the messages, logs the third one, and deletes them all.

Another useful feature is the retrieve request. Since read_mail operates on a copy of the mailbox, messages that have been deleted with the delete request do not actually get deleted until the quit request is issued. It is therefore possible to retrieve a deleted message by number before quitting.

Thus, if you use the delete request and unexpectedly see "All messages have been deleted.", you can "retrieve all" to get back any messages that were deleted by mistake. Message numbers are constant within a single invocation of read_mail. That is, a message listed as number 4 remains number 4 until the quit request is issued.

Messages can be specified several different ways. If no message is specified, the current one is assumed. Message specifiers can be numbers, keywords like "all" and "next", and regular expressions as in the qedx editor. Some examples are:

```
print 1:5            Print messages 1 through 5.
delete 1 4 7         Delete messages 1, 4, and 7.
print all            Print all messages.
log previous         Log the message before the current one.
delete /Frankston/   Delete all messages containing
                     the string "Frankston".
```

Most of the requests and keywords have abbreviations to ease typing, for example:

Requests:

| | |
|---|---|
| print | pr |
| delete | dl |
| list | ls |
| log | (no abbreviation) |
| quit | q |
| retrieve | rt |

Keywords:

| | |
|---|---|
| all | a |
| previous | p |
| next | n |
| first | f |
| last | l |
| current | c |

Many other requests are described in the MPM Commands section on read_mail. Four more are the save request, which acts like the log request but saves messages in whatever mailbox is specified, the write request, which saves messages in an ASCII segment, the qedx request, which edits messages (for example, to add extra notes before saving them), and the reply request, which sends a reply to all the recipients and the senders of a particular message.

The help request prints information on the use of read_mail. Type "help requests" to see a list of requests, "help REQUEST_NAME" to describe a request, and "help *" for a list of other help arguments.

Summary:

```
read_mail (input-spec) (-control_args)
```

where:

1. input-spec is a single input specifier, either a PERSON.PROJECT identifier or the pathname of a mailbox. If this argument is not present, read the user's own mailbox. If it is -log, read the user's log box.

2. control_args can be:

| | |
|---|---|
| -list, -ls | print a summary before entering the request loop. |
| -print, -pr | print the messages before entering the request loop. |
| -total, -tt | print the number of messages and return, without accepting requests. |

Requests:

| | |
|---|---|
| delete, dl | delete specified messages. |
| help | print information about read_mail. |
| list, ls | print a summary of the messages. |
| log | save messages in the log box. |
| print, pr | print messages. |
| qedx, qx | edit messages. |
| quit, q | exit read_mail. |
| retrieve, rt | un-delete messages. |
| save, sv | save messages in save boxes. |
| write, w | write messages to ASCII files. |

USING send_mail

The companion function to read_mail is send_mail. This command facilitates the composition and transmission of a message to one or more mailboxes.

To send mail to Greenberg.Multics, you type:

```
send_mail Greenberg.Multics
```

The send_mail command prompts "Subject:" and accepts a single line, then it prompts "Message:" and accepts the message text. There are three ways to terminate the text.

A line consisting of a period (".") sends the message as is. The character sequence \f invokes the qedx editor on the text that has been typed. The character sequence \q causes send_mail to type the prompt "send_mail:" and accept requests from the terminal.

The log request saves a copy of the message to be sent in the user's log box. The save request saves a copy in a specified mailbox whose name ends in .sv.mbx. The write request saves a copy in an ASCII segment whose name ends in .mail.

The qedx request, like \f in input mode, enters the qedx editor with a copy of the message text in the buffer. Unlike the qedx invoked from command level, this one does not require you to type w (write) to reflect changes in the buffer to the actual message text. These changes are automatically made in the text to be sent.

The send request sends the message.

The quit request exits send_mail and returns to command level. If the message that has been composed has not been sent anywhere, send_mail asks "Do you wish to send the message?" and accepts a yes or no reply. If the message has been sent, you are simply returned to command level.

The following is a sample dialog with send_mail. The user's input is preceded by a ! character.

```
!  send_mail Sibert.PDO Palter.PDO -log

   Subject:  !  New Model Biplanes
   Message:
!  We got a big shipment of the new modal biplanes.  I think these
!  are a great improvement over the old traditional 707's, since
!  they are stabler and carry many more people.
!  What do you think??
!
!  -- Old Reliable
!  \f
!  1,$s/modal/model/
!  /model/
   We got a big shipment of the new model biplanes.  I think these
!  s/model/Teflon-coated/
!  .2s/stabler/more stable/
!  q

   send_mail:  !  send
   Mail delivered to your log box.
   Mail delivered to Sibert.PDO.
   Mail delivered to Palter.PDO.

   send_mail:  !  save biplanes
   Mail saved in >udd>AeroProj>SkyKing>biplanes.sv.mbx

   send_mail:  !  quit
   r 1052 .791 4.132 109
```

Two other requests are print, which prints the message, and log, which saves a copy of the message in the same log box used by read_mail. If the -log control argument is specified on the send_mail command line, the message is automatically saved in your log box when it is sent.

To send a message contained in a file, say:

```
send_mail USER.PROJECT -input_file PATH
```

To send "carbon copies" to secondary recipients while sending to one or more primary recipients, use the -cc control argument on the send_mail command line.

The remaining control arguments and requests are described under send_mail in the MPM Commands document.

Summary:

send_mail {addresses} {-control_args}

where:

1. addresses are PERSON.PROJECT identifiers and pathnames of mailboxes. If no address arguments are present, expect the "send" request to specify a destination.

2. control_args can be:

-cc ADDRESS send a carbon copy to ADDRESS.
-input_file PATH, -if PATH
send mail contained in a segment.
-log save a copy of the message in the log box.
-subject STRING, -sj STRING
specify a subject for the message.

Requests:

| | |
|---|---|
| help | print information about send_mail. |
| log | save a copy in the log box. |
| print, pr | print the message to be sent. |
| qedx, qx | edit the message to be sent. |
| quit, q | exit send_mail. |
| save, sv | save a copy of the message. |
| send | send the message. |
| subject, sj | specify a subject string. |
| write, w | save a copy in an ASCII file. |

MPM:

Name: send_mail, sdm

This command transmits a message to one or more recipients specified by Person.Project identifier or mailbox pathname. It either accepts an input file or reads text from the terminal, then either sends the message or reads requests for editing, copying and sending. The message is automatically prefixed by a header, whose standard fields give the authors, the intended recipients, and a brief summary of the contents. These fields are understood by the read_mail command.

Usage

send_mail (addresses) (-control_args)

1. addresses are described in detail under Addresses below.

2. control_args can be interspersed with the addresses and are among the following:

-acknowledge, -ack
-brief, -bf
-cc ADDRESSES
-fill
-from ADDRESSES
-header, -he
-in_reply_to STRING, -irt STRING
-input_file PATH, -if PATH
-line_length N, -ll N
-log
-long, -lg
-message_id, -mid
-no_acknowledge, -nack
-no_fill, -nfi
-no_header, -nhe
-no_log
-no_message_id, -nmid
-no_request_loop, -nrql
-no_subject, -nsj
-prompt STRING
-reply_to ADDRESSES, -rpt ADDRESSES
-request_loop, -rql
-save PATH, -sv PATH
-subject STRING, -sj STRING
-terminal_input, -ti
-to ADDRESSES

Notes

Any addresses appearing on the command line before the first -cc, -from, -reply_to, or -to control argument are considered primary recipients of the message. (See the description of the -to control argument below.)

The -cc, -from, -reply_to, and -to control arguments apply to all subsequent addresses until the next of these control arguments is given. Any other intervening control arguments do not affect this interpretation.

For example, the sequence:

```
addr1 -from addr2 addr3 -cc addr4 -to addr5
```

causes addr1 and addr5 to be processed by -to, addr2 and addr3 to be processed by -from, and addr4 to be processed by -cc.

If conflicting control arguments (for instance, -header and -no_header) are specified, the last one takes effect.

Addresses

Addresses are used by send_mail to identify both the authors and the recipients of a message. An address refers either to a mailbox, by pathname or owner, or to several mailboxes listed in an ASCII mailing list segment.

The permissible forms of address are:

```
-mailbox PATH, -mbx PATH
-user PERSON.PROJECT
-mailing_list PATH, -mls PATH
STRING
```

-mailbox PATH,
-mbx PATH

specifies a mailbox pathname. The .mbx suffix is added to PATH if it is not present.

-user PERSON.PROJECT

specifies a user as an address. The corresponding mailbox pathname is:

```
>udd>PROJECT>PERSON>PERSON.mbx
```

This control argument is useful when a segment named PERSON.PROJECT.mbx exists in the working directory.

-mailing_list PATH,
-mls PATH
finds addresses in the mailing list designated by PATH. The suffix .mls is added to PATH if it is not present. The mailing list is an ASCII segment containing one address per line. Any of the acceptable forms of address can be used. Note that a mailing list can refer to other mailing lists. (No mailing list is used twice recursively.)

STRING
is any argument that does not begin with a minus sign (-). If it contains either of the characters > or <, it is interpreted as:

-mailbox STRING

If STRING does not contain > or <, it is interpreted as:

-user STRING

Addresses can be qualified by the -comment control argument. The use of -comment does not affect the destination of the message.

-comment STRING
-cmt STRING
places STRING in the header field associated with the address (including -mailbox, -user, and -mailing_list) directly preceding. If this control argument does not directly follow an address, STRING is placed by itself in the header field most recently referred to. Comments are enclosed in parentheses.

For example, the command line:

sdm Palter.PDO -comment "send_mail person"

creates the header field:

To: Palter.PDO (send_mail person)

Control Arguments

-acknowledge,
-ack causes send_mail to request that a message be sent to the user of send_mail by each recipient of the message after they have read the message. The user's name is placed in the Acknowledge-To header field. (See Headers below.)

-brief,
-bf suppresses printing of the message:

Mail delivered to ADDRESS.

when mail is sent.

-cc ADDRESSES
causes subsequent ADDRESSES to be added as secondary recipients of the message. Mail is sent to these addresses when the send request is issued with no arguments. (See Requests below.) These addresses are placed in the cc header field. (See Headers below.) There are no secondary recipients by default.

-fill causes the text of the message to be reformatted according to "fill-on" and "align-left" modes in compose, before sending, entering the editor, or entering the request loop. The line length used is 72 unless specifed by the -line_length control argument. If the -fill control argument is not specified, the message text is left unchanged.

-from ADDRESSES
causes subsequent addresses to be added as authors of the message. These addresses are placed in the From header field, overriding the user's name placed there by default.

-header,
-he causes a message header to be generated by send_mail. This is the default.

-in_reply_to STRING,
-irt STRING
causes STRING to be placed in the In-Reply-To field of the header. This field is not present by default.

-input_file PATH,
-if PATH
sends a message contained in a file. The file is sent without entering the request loop unless -request_loop (-rql) is specified. If -input_file is not specified, the user is prompted for the message text ("Message:").

-line_length N,
-ll N specifies a line length to be used when adjusting text via -fill or the fill request. The default line length is 72.

-log causes a copy of the message to be sent to the user's log box, the mailbox named PERSON.sv.mbx in the home directory. The user's name is added to the cc header field. (See Headers below.)

-long,
-lg causes the "Mail delivered to ADDRESS" message to be printed when mail is sent. This is the default.

-message_id,
-mid adds a Message-ID field to the header, containing a unique identifier for the message. This field is not present by default. (See Headers below.)

-no_acknowledge,
-nack prevents send_mail from requesting that each recipient of the message acknowledge reading the message. This is the default.

-no_fill,
-nfi causes the message text to not be adjusted as described above for -fill. This is the default.

-no_header,
-nhe causes the normal message header to not be added to the message. The only header fields added are those explicitly requested by control arguments or requests.

-no_log specifies that a copy of the message is not to be sent to the user's log box. This is the default.

-no_message_id,
-nmid specifies that a Message-ID field is not to be added to the header. This is the default.

-no_request_loop,
-nrql causes the message to be sent without entering the request loop. If an error occurs while sending the message, the request loop is entered anyway. This is the default.

-no_subject,
-nsj specifies that a Subject field is not to be added to the header.

-prompt STRING
causes the request loop to be prompted by STRING(N):, where N is the recursion level if greater than one, instead of the default prompt "send_mail(N):". If STRING is "", the user is not prompted.

-reply_to ADDRESSES,
-rt ADDRESSES
causes subsequent addresses to be added to the *Reply-To* header field. This field is not present by default. (See Headers below.)

-request_loop,
-rql causes the request loop to be entered before sending a file via -input_file (-if) or after "." is typed to terminate the input text. The default is to automatically send the message and quit.

-save PATH,
-sv PATH
causes a copy of the message to be sent to the save box PATH. The suffix .sv.mbx is added to PATH if it is not present. If the save box does not exist, the user is asked whether to create it. The user's name with a comment containing the entry name of the save box is added to the cc header field. (See Headers below.)

-subject STRING,
-sj STRING
causes STRING to be placed in the Subject field of the header. If STRING is "", no Subject field is created. If this control argument is not specified, the user is asked for a subject with the prompt "Subject:". A blank response causes the Subject field to be omitted.

-terminal_input,
-ti causes the user to be prompted for the message text ("Message:"). The user then types the message text terminated by a line consisting of a period ("."). This is the default.

-to ADDRESSES
causes subsequent ADDRESSES to be added as primary recipients of the message. Addresses not preceded by any of the above control arguments are also primary recipients. All of these addresses are placed in the To header field (see Headers below). Mail is sent to them when the send request is issued with no arguments (see Requests below). There are no primary recipients by default.

Headers

Messages created by send_mail begin with a message header. This header contains information used by the read_mail and send_mail commands to facilitate listing, saving, forwarding, and replying to messages.

The message header is separated from the text by one or more blank lines. It consists of one more more fields. Each field consists of an identifier, a colon, and one or more entries. Multiple entries are separated by commas. If a field is too long to fit on one line, it is continued on successive lines. (See the following example.)

A sample header is:

```
Date:     25 May 1978 14:54-EDT
From:     Palter.PDO
Subject:  headers in send_mail
To:       Sibert.PDO, Herbst.Multics,
          (mbx >udd>PDO>Palter>mlsys.sv)
cc:       Greenberg.Multics
```

ADDRESSES IN HEADERS

Several header fields contain lists of addresses. There are three distinct formats used for an address in a header field:

PERSON.PROJECT
: specifies a user identifier. It is generated by the -user form of address or a STRING address that is not a mailbox pathname.

(mbx PATH)
: specifies a mailbox pathname. The .mbx suffix is not included in PATH.

(list PATH)
: specifies a mailing list pathname. The .mls suffix is not included in PATH.

In addition, any of the above can be followed by a comment in parentheses. This comment is supplied by the -comment control argument. For example:

```
(list >udd>PDO>Palter>mlsys) (Mail System Developers)
```

HEADER FIELDS

The standard header fields are listed below in the order they appear. The first two are always present. The others are optional.

Date contains the date and time when the message was first transmitted, for example:

5 June 1978 23:45-EDT

From contains a list of addresses identifying the author(s) of the message. If -from is not specified and the from request is not issued, this field names the user who invoked send_mail.

Sender contains the address of the user who actually sent the message. This field is created only if the -from control argument or the from request is used.

Reply-To contains a list of addresses to which a reply should be sent. When present, read_mail uses the addresses given in this field instead of the addresses in the From field as recipients of a reply request. This field is created only if the -reply_to control argument or the reply_to request is used.

Subject contains a brief description of the contents of the message. This field is present only if a non-null string is given to the -subject control argument, a non-blank line is given to the "Subject:" prompt, or the subject request is issued with one or more arguments.

In-Reply-To
contains an ASCII string describing the message to which this message is a reply. This field is present only if the -in_reply_to control argument or the in_reply_to request is used.

To contains a list of addresses naming the primary recipients of the message. The send request when invoked with no arguments sends the message to all of these addresses. This field is created only if primary recipients are specified on the command line or if the to request is used.

cc contains a list of addresses naming the secondary recipients of the message. The send request when invoked with no arguments sends the message to these addresses. This field is created only if the -cc control argument or the cc request is used, or if either -save or -log is specified, in which case a cc field entry is created for the person creating the message, with a comment containing the name of the mailbox being saved into, as in:

Sibert.PDO (mail_system.sv)

Acknowledge-To
This field contains an address to which an acknowledgement message should be sent after someone receiving this piece of mail reads it. This field is generated only when the -acknowledge control argument to send_mail is used.

Message-ID
contains a unique character string identifier for the message. This string is generated by send_mail, and is useful when reading messages to detect multiple copies of the same message.

Redistributed-Date
Redistributed-By
Redistributed-To
specify information about the forwarding of the message. These fields are added only by read_mail's forward request.

Terminal Input

Unless -input_file (-if) is specified, send_mail prompts with "Message:" and reads text from the terminal. Three character sequences are recognized in terminal input:

A line consisting of a period (".") causes send_mail to send the message and quit. If the message cannot be sent, send_mail enters its request loop.

A line containing the characters \f causes send_mail to enter the qedx editor. The remainder of the line following the \f is treated as editor requests. When qedx is exited, the request loop is entered. See the description of the qedx request below.

A line consisting of the characters \q causes read_mail to enter its request loop.

Request Loop

Requests are prompted by the default string "send_mail:" or by a user-settable prompt string (see the -prompt control argument above). If the user has invoked send_mail recursively, the prompt also contains a recursion number, for example "send_mail(3):". This example indicates that there are two interrupted invocations of send_mail in the process as well as the current one.

The quit request terminates the request loop and exits send_mail. Requests are available to edit (qedx or qx), reformat (fill or fi), and transmit (send) the message. Requests are also available to modify the header fields.

A request line beginning with ".." is treated as a special escape used to pass commands directly to the Multics command processor. Except for removing the leading "..", send_mail does not process this line in any way before passing it on.

The request line consisting of a single period (".") causes send_mail to print "send_mail (N) re: SUBJECT", where N is the current recursion level if greater than one and SUBJECT is the subject field if one was specified.

The request line consisting of "?" prints a summary of the available send_mail requests and request functions.

Request lines have identical syntax to Multics command lines. Arguments containing spaces or other command language characters must be quoted. Iteration is specified by means of parentheses. Semicolon (;) is used to separate multiple requests on a line.

Brackets invoke send_mail request functions, which act like Multics active functions but are chosen from an internal set. These return character strings that are useful within the context of send_mail. The execute (e) request function can be used to invoke Multics active functions from within send_mail.

When an error occurs executing a request, the request is aborted and any other requests on that line are discarded.

The available send_mail requests are:

```
?
apply (-CONTROL_ARG) STRINGS,
   ap (-CONTROL_ARG) STRINGS
cc (ADDRESSES)
execute STRINGS, e STRINGS
fill (-CONTROL_ARG), fi (-CONTROL_ARG)
from (ADDRESSES)
help (STRING)
in_reply_to (STRINGS), irt (STRINGS)
list, ls
log
message_id, mid
print (-CONTROL_ARG), pr (-CONTROL_ARG)
qedx (-CONTROL_ARG), qx (-CONTROL_ARG)
quit (-CONTROL_ARG), q (-CONTROL_ARG)
remove (ADDRESSES) (-CONTROL_ARGS)
reply_to (ADDRESSES), rpt (ADDRESSES)
save PATH, sv PATH
send (ADDRESSES) (-CONTROL_ARGS)
subject (STRINGS), sj (STRINGS)
to (ADDRESSES)
write PATH
```

The available request functions are:

```
cc
execute STRINGS, e STRINGS
from
in_reply_to, irt
message_id, mid
reply_to, rpt
subject, sj
to
```

REQUEST DESCRIPTIONS

? Print a summary of the available send_mail requests.

apply (-CONTROL_ARG) STRINGS,
ap (-CONTROL_ARG) STRINGS

Place the message in a temporary segment in the process directory. Concatenate STRINGS with intervening spaces and append the pathname of the temporary segment. Pass this concatenated command line to the Multics command processor. When the command line has completed, replace the message in send_mail with the contents of the temporary segment.

If the -header (-he) control argument is specified, operate on the header and the text. If -no_header (-nhe) or no control argument is specified, operate on the text only. Control arguments must precede the STRINGS.

If the message header is changed by the command line, send_mail parses it and updates the lists of primary and secondary recipients, authors, reply addresses, etc.

This request can be used to edit the message with an arbitrary editor, for example:

```
apply teco
```

cc (ADDRESSES)

If any addresses are specified, add them to the list of secondary recipients of the message. Mail is sent to these addresses when a subsequent send request is issued with no arguments. The addresses are added to the cc field, which is created if necessary.

If no addresses are specified, list the secondary recipients of the message.

The cc request function returns the list of secondary recipients, which are separated by commas and spaces. It returns "" if there are no secondary recipients.

execute STRINGS,
e STRINGS

Pass the concatenation of STRINGS with intervening spaces to the Multics command processor. This request is different from ".." because it is first parsed as a send_mail request line. The send_mail request interpreter expands send_mail request functions, strips quotes, and performs iteration before the line is passed on to the command processor. Therefore, the request line:

```
e ioa_ [subject]
```

prints the contents of the subject field, whereas:

```
..ioa_ [subject]
```

produces the error message: "Segment subject not found.". The ".." escape should normally be used to execute Multics command lines from within send_mail.

The execute request function can be used to invoke a Multics active function from within send_mail. The request line:

```
save [execute date]
```

saves the message in a save box whose name is the current date.

fill (-CONTROL_ARG),
fi (-CONTROL_ARG)

Reformat the message text according to "fill-on" and "align-left" modes in compose. If the -line_length N (-ll N) control argument is specified, use N as the line length. Otherwise, use the value specified to the -line_length control argument on the send_mail command line, or 72 if -line_length was not specified.

from (ADDRESSES)

If any addresses are specified, add them to the list of authors of the message. The addresses are added to the From field of the header.

If no addresses are specified, list the authors of the message.

The from request function returns the list of authors, which are separated by commas and spaces.

help {STRING}
: Print information about the send_mail command. If specified, STRING is the name of a send_mail request or one of the topics "requests", "control_args", and "changes". If STRING is "*", print the list of available send_mail topics. If STRING is not specified, print introductory information on the use of send_mail followed by a list of topics.

in_reply_to {STRINGS},
irt {STRINGS}
: If STRINGS are supplied, replace the In-Reply-To field of the message (if any) with the concatenation of the STRINGS with intervening spaces. Otherwise, print the current In-Reply-To field of the message.

  The in_reply_to (irt) request function returns the In-Reply-To field of the message, or "" if there is none.

list,
ls
: Print a summary of the message in the format produced by read_mail. (See the read_mail command in this document.)

log
: Save a copy of the message in the user's log box (PERSON.sv.mbx). This request creates the log box if it does not already exist.

message_id,
mid
: Print the Message-ID field of this message, creating the field if necessary.

  The message_id (mid) request function returns the Message-ID field, creating it if necessary.

print {-CONTROL_ARG},
pr {-CONTROL_ARG}
: Print the message. If the -header (-he) control argument is specified, print the header and the text. If -no_header (-nhe) or neither is specified, print only the text.

qedx {-CONTROL_ARG},
qx {-CONTROL_ARG}
: Invoke the qedx editor to modify the message. If the -header (-he) control argument is specified, edit both the header and the text. If -no_header (-nhe) or neither is specified, edit only the text.

The qedx w (write) request is not necessary to reflect changes in the message to send_mail. An additional request, Q (quit-force), is added to qedx to return to send_mail without reflecting any changes made during editing.

If the message header is changed during editing, send_mail parses it when qedx returns and updates the lists of primary and secondary recipients, authors, reply addresses, etc. Requests to send_mail (subject, reply_to, etc.) are recommended over qedx requests for changing header fields.

quit (-CONTROL_ARG),
q (-CONTROL_ARG)

Exit the send_mail command. If the message has not been sent, or if it has been modified by qedx since it was last sent, and if the -force (-fc) control argument is not specified, query the user before exiting.

remove (ADDRESSES) (-CONTROL_ARGS)

At least one ADDRESS or one control argument must be specified.

Delete from the list of primary and/or secondary recipients any ADDRESSES appearing before the first -cc, -from, -reply_to, or -to control argument.

Delete ADDRESSES appearing after -cc, -from, -reply_to, or -to from the specified field. Delete the entire field if the control argument is followed by -all.

If the -in_reply_to (-irt), -message_id (-mid), or -subject (-sj) control argument is specified, delete the appropriate field entirely. The presence of this class of control argument does not affect the interaction of ADDRESSES and the other control arguments.

For example, the request line:

```
remove -subject Palter.PDO -from CRDavis.CSR
     Sibert.PDO -to Herbst.Multics
```

deletes the Subject field from the message, removes Palter.PDO from both the primary and secondary recipient lists, removes Herbst.Multics from the primary recipient list only, and removes CRDavis.CSR and Sibert.PDO from the list of authors of the message.

reply_to (ADDRESSES),
rpt (ADDRESSES)

If any addresses are specified, add them to the list of addresses to use when sending a reply to this message. These addresses are also appended to the Reply-To field of the header, which is created if necessary.

If no addresses are specified, list the addresses to receive replies.

The reply_to (rpt) request function returns the list of reply addresses, which are separated by commas and spaces. It returns "" if there are no reply addresses.

save PATH,
sv PATH

Save a copy of the message in the indicated save box. The suffix .sv.mbx is added to PATH if not already present. If the save box does not exist, the user is asked whether to create it.

send (ADDRESSES) (-CONTROL_ARGS)

If no arguments are specified, transmit the message to the primary and secondary recipients.

If any ADDRESSES are specified, transmit the message to these ADDRESSES without adding them to the header.

The control arguments -log and -save PATH (-sv PATH) cause a copy of the message to be placed in the log box and specified save box, respectively.

subject (STRINGS),
sj (STRINGS)

If any STRINGS are specified, replace the Subject field of the message (if any) with the concatenation of the STRINGS with intervening spaces. Otherwise, print the Subject field of the message.

to (ADDRESSES)

If any ADDRESSES are specified, add them to the list of primary recipients of the message. Mail is sent to these addresses when a subsequent send request is issued with no arguments. The addresses are added to the To field of the header, which is created if necessary.

If no ADDRESSES are specified, list the primary recipients of the message.

The to request function returns the list of primary recipients, which are separated by commas and spaces. It returns "" if there are no primary recipients.

write PATH (-CONTROL_ARG)

Append the message complete with header to an ASCII file, in a format acceptable to the -input_file option of read_mail. The suffix .mail is added to PATH if it is not present. The -extend and -truncate (-tc) control arguments accepted by the file_output command can be specified.

Name: read_mail, rdm

This command provides a facility for examining and manipulating the contents of a mailbox. By default, it operates only on mail and send_mail messages, ignoring interactive messages from the send_message command. The *read_mail* command enters a loop reading requests from the terminal. These requests enable the user to selectively list, print, delete, save, forward, and reply to messages.

Usage

read_mail (input-spec) (-control_args)

where:

1. input-spec Is a single input-spec of one of these forms:

STRING
-log
-save PATH
-sv PATH
-mailbox PATH
-mbx PATH
-input_file PATH
-if PATH
-user Person.Project

2. control_args are selected from among the following:

-brief, -bf
-header, -he
-interactive_messages, -im
-list
-long, -lg
-no_header, -nhe
-own
-print, -pr
-prompt STRING
-quit
-total, -tt

Input-specs

An input-spec tells read_mail where to read from. It specifies either a mailbox or an ASCII file produced by the read_mail write (w) request. If no input-spec appears in the command line, the user's default mailbox (>udd>PROJECT>PERSON>PERSON.mbx) is read. Since only one source can be read at a time, it is an error to include more than one input-spec in the read_mail command line.

-input_file PATH,
-if PATH
causes read_mail to read from the ASCII file named PATH instead of from a mailbox. The .mail suffix is added to PATH if it is not present. This file is assumed to have been produced by the read_mail write (w) request. If read_mail is unable to parse the file, it prints an error message and exits.

-log
causes read_mail to read from the user's log box instead of from the user's mailbox. The log box is the mailbox PERSON.sv.mbx in the home directory.

-mailbox PATH,
-mbx PATH
causes read_mail to read from the mailbox specified by PATH instead of from the user's default mailbox. The .mbx suffix is added to PATH if it is not present.

-save PATH,
-sv PATH
causes read_mail to read from the mailbox PATH instead of from the user's default mailbox. The suffix .sv.mbx is added to PATH if it is not present. This control argument is equivalent to "-mailbox PATH.sv".

-user NAME.PROJECT
causes read_mail to read from the specified user's mailbox. This control argument is useful if a segment by the name PERSON.PROJECT exists in the working directory.

STRING
is any argument that does not begin with a minus sign (-). If it contains either of the characters > or <, it is the pathname of a mailbox. The .mbx suffix is added if it is not present. If STRING does not contain > or <, it is interpreted as "-mailbox STRING", and if no such mailbox exists, it is interpreted as "-user STRING".

Control Arguments

-brief, -bf causes read_mail to be terser when printing informative messages.

-header, -he causes message headers to be printed by the print (pr) request to read_mail. (See the section on Headers in the description of the send_mail command.) By default, these headers are omitted and a brief header line is printed, along with the subject line (if one was present), as in:

```
3. (24)   07/22/78 12:20 Herbst.Multics          subject
```

-interactive_messages,
-im causes read_mail to operate on interactive messages as well as mail messages. The default is to ignore interactive messages.

-list, -ls prints a summary of the messages in the mailbox before entering the request loop.

-long, -lg causes read_mail to print the full text of its informative messages, as opposed to -brief. This is the default.

-no_header,
-nhe causes headers to be omitted when printing messages via the print (pr) request. Instead, a brief header line is printed giving the message number, the sender, and the date-time sent. This is the default.

-own causes read_mail to operate only on the user's own messages instead of on all the messages. This control argument can be useful when examining someone else's mailbox.

-print, -pr prints the messages in the mailbox before the entering the request loop.

-prompt STRING
changes the prompt for read_mail request lines to STRING(N):, where N is the recursion level if greater than one. The default prompt is "read_mail(N):". If STRING is "", the user is not prompted.

-quit causes read_mail to exit after performing any operations specified by control arguments. The default is to enter the request loop.

-total, -tt causes read_mail to print the total number of messages in the mailbox (including interactive ones if -interactive_messages is specified) and quit. The request loop is not entered.

Requests

When invoked, the read_mail command goes into a loop accepting requests. Requests are available to list, print, edit, save, forward, and reply to messages in the mailbox. The quit request terminates the loop and exits read_mail.

A line beginning with ".." is treated as a special escape used to pass commands directly to the Multics command processor. Except for removing the leading "..", read_mail does not process the line in any way before passing it on.

A line consisting of a single period (".") causes read_mail to print "read_mail (N): PATH #M", where N is the current recursion level if greater than one, PATH is the pathname of the mailbox being read, and M is the number of the current message or "[EMPTY]" if all messages have been deleted.

A line consisting of "?" prints a list of the available read_mail requests.

Request lines have identical syntax to Multics command lines. Arguments containing spaces or other command language characters must be quoted. Iteration is specified by means of parentheses. Semicolon (;) is used to separate multiple requests on a line.

Brackets invoke read_mail request functions, which act like Multics active functions but are chosen from an internal set. These return character strings that are useful inside read_mail. The execute (e) request function can be used to invoke Multics active functions from within read_mail.

When an error occurs executing a request, the request is aborted and any other requests that were present on that line are discarded.

MESSAGE SPECIFIERS

Message specifiers refer to messages in the mailbox. They are composed of message numbers, keywords, the arithmetic operators + and -, and qedx-type regular expressions for string matching. Ranges are composed of two expressions separated by colon (:), for example:

```
6:last-3
```

Message numbers are integers. These are assigned by read_mail when listing or printing the messages in the mailbox. Messages, including deleted messages, keep their numbers during an invocation of read_mail.

The available keywords and their short forms are:

```
first, f
last, l
previous, p
next, n
current, c
all, a
```

The first four are used like message numbers, for example:

```
last-1
```

The "current" message is initialized to 1 and changed by various requests. The "all" keyword denotes the range of all messages, and is equivalent to first:last.

Simple regular expressions are character strings enclosed in slashes (/). Slashes inside the match string must be preceded by \c. More complicated expressions can be built from simple ones, the connector & for logical AND, and the connector ¦ for logical OR. Any of these expressions can be preceded by a keyword, for example:

```
last/artificial/&/intelligence/
```

which specifies the last message containing both of the strings "artificial" and "intelligence".

Because of the syntax of the request language, regular expressions containing special characters such as quote, space, and parenthesis must be enclosed in quotes.

Keywords can be used as prefixes for regular expressions. As prefixes, they mean the first, last, previous, or next message matching the given regular expression. The message specifier current/STRING/ is undefined if the current message does not match STRING. If a regular expression is not prefixed with a keyword, the default keyword is "all".

Normally, message specifiers refer to messages that have not been deleted. The "all" keyword refers to all undeleted messages, and "first" refers to the first undeleted message. If the -all control argument is specified to a request, however, deleted messages are included in the ranges.

If a range is specified and -all is not specified, there must be at least one message within the range that has not been deleted. For instance, if the mailbox originally had 20 messages in it, and 10 and 12 are the only ones left, it is perfectly valid to say "print 4:11". This request prints only message number 10.

The values used for "last", "first", "next", and "previous" also change depending on whether -all is specified. If -all is not specified, they refer only to existing messages. Therefore, in the example above, the request line "print last-4:1" is the same as "print 10 12" and the request line "print last-4:1 -all" is the same as "print 16:20 -all". The last two request lines print deleted messages.

Some examples of message specifiers are:

```
1           message number 1
1:3         messages 1 through 3
/foo/       all messages containing the string "foo"
last-3      the third from last message
1-3:1       the last four messages
next+4      the message five after the current one
p-2         three messages previous
c:c+4       the current message and the next four
c+1:1       the next through last messages
/a/|/b/     all messages containing either "a" or "b"
"1/ it /"   the last message containing " it "
```

The available read_mail requests are listed below. The string SPECS stands for message specifiers and ADDRESS stands for a mail address.

```
?
apply (-CONTROL_ARGS) STRINGS,
   ap (-CONTROL_ARGS) STRINGS
cc (ADDRESSES) (-CONTROL_ARGS)
copy (SPECS) PATH, cp (SPECS) PATH
delete (SPECS), dl (SPECS)
execute STRINGS, e STRINGS
fill (SPECS) (-CONTROL_ARG), fi (SPECS) (-CONTROL_ARG)
forward SPEC ADDRESSES, fwd SPEC ADDRESSES
from (ADDRESSES) (-CONTROL_ARG)
help (STRING)
in_reply_to (STRINGS) (-CONTROL_ARG),
   irt (STRINGS) (-CONTROL_ARG)
list (SPECS), ls (SPECS)
log (-CONTROL_ARG) (SPECS)
message_id (SPECS), mid (SPECS)
print (-CONTROL_ARG) (SPECS), pr (-CONTROL_ARG) (SPECS)
qedx (-CONTROL_ARG), qx (-CONTROL_ARG)
quit, q
remove (ADDRESSES) (-CONTROL_ARGS)
reply (SPECS), rp (SPECS)
reply_to (ADDRESSES) (-CONTROL_ARG),
   rpt (ADDRESSES) (-CONTROL_ARG)
retrieve (SPECS), rt (SPECS)
save (-CONTROL_ARG) (SPECS) PATH,
   sv (-CONTROL_ARG) (SPECS) PATH
subject (STRINGS) (-CONTROL_ARGS),
   sj (STRINGS) (-CONTROL_ARGS)
to (ADDRESSES) (-CONTROL_ARGS)
write (SPECS) PATH, w (SPECS) PATH
```

The available read_mail request functions are:

```
cc (SPEC)
current (-CONTROL_ARG), c (-CONTROL_ARG)
date (SPEC), dt (SPEC)
execute, e
first (-CONTROL_ARG), f (-CONTROL_ARG)
from (SPEC)
in_reply_to (SPEC), irt (SPEC)
last (-CONTROL_ARG), l (-CONTROL_ARG)
message_id (SPEC), mid (SPEC)
next (-CONTROL_ARG), n (-CONTROL_ARG)
previous (-CONTROL_ARG), p (-CONTROL_ARG)
reply_to (SPEC), rpt (SPEC)
subject (SPEC), sj (SPEC)
to (SPEC)
```

REQUEST DESCRIPTIONS

The various read_mail requests are described below. For a description of headers and header fields, see the section entitled Headers under the send_mail command in this document.

? Print a summary of the available read_mail requests.

apply (-CONTROL_ARGS) STRINGS,
ap (-CONTROL_ARGS) STRINGS

Place the current message or the single one specified by -message SPEC (-msg SPEC) in a temporary segment in the process directory. Concatenate STRINGS with intervening spaces, and append the pathname of the temporary segment. Pass this concatenated command line to the Multics command processor. When the command line has completed, replace the message with the contents of the temporary segment.

If the -header (-he) control argument is specified, operate on the header and the text. If -no_header (-nhe) or neither is specified, operate on the text only. Control arguments, including -message SPEC (-msg SPEC), must precede the STRINGS.

cc (ADDRESSES) (-CONTROL_ARGS)

Add the specified addresses to the cc header field. If no addresses are specified, print the contents of the cc field. The control argument -message SPEC (-msg SPEC) causes this request to operate on specified messages instead of on the current message. The remove request can be used to delete addresses from the header.

The cc request function must be invoked on a single message and returns the cc header field, or "" if there is none.

copy (SPECS) PATH,
cp (SPECS) PATH

Copy the specified messages verbatim into the mailbox designated by PATH. The .mbx suffix is added to PATH if it is not present. Unlike the save and log requests, this request does not add Date and From header fields if they are missing.

current (-CONTROL_ARG),
c (-CONTROL_ARG)

This request function returns the number of the current message, or 0 if the current message has been deleted. If -all is specified, it returns the number of the current message whether or not that message has been deleted, or 0 if there are no messages.

date (SPEC),
dt (SPEC)

This request function must be invoked on a single message and returns the Date field.

delete (SPECS)
dl (SPECS)

Delete the specified messages. If no messages are specified, delete the current one. Deleted messages can be retrieved before exiting read_mail by using the retrieve (rt) request.

execute STRINGS,
e STRINGS

Pass the concatenation of STRINGS with intervening spaces to the Multics command processor. This request is different from ".." because it is first parsed as a read_mail request line. The read_mail request interpreter expands read_mail request functions, strips quotes, and performs iteration before the line is passed on to the command processor. Therefore, the request:

```
e ioa_ [subject]
```

prints the contents of the subject field, whereas:

```
..ioa_ [subject]
```

produces the error message: "Segment subject not found.". The ".." escape should normally be used to execute Multics command lines from within read_mail.

The execute request function can be used to invoke a Multics active function from within read_mail. For example:

```
save [execute date]
```

saves the current message in a save box whose name is the current date.

fill (SPECS) (-CONTROL_ARG)
fi (SPECS) (-CONTROL_ARG)

Reformat the text of the specified messages according to "fill-on" and "align-left" modes in compose. If no messages are specified, reformat the current one. If -CONTROL_ARG is -line_length N (or -ll N), use N as the line length. Otherwise, use the default line length 72.

first (-CONTROL_ARG),
f (-CONTROL_ARG)
This request function returns the number of the first message that has not been deleted, or 0 if all messages have been deleted. If -all is specified, it returns the number of the first message whether or not that message has been deleted, or 0 if there are no messages.

forward SPEC ADDRESSES,
fwd SPEC ADDRESSES
Forward the messages indicated by the single message specifier to ADDRESSES. This request adds three fields to the header to record where the message came from: Redistributed-Date, Redistributed-By, and Redistributed-To. Iteration is needed to give more than one message specifier.

from (ADDRESSES) (-CONTROL_ARG)
Add the specified addresses to the From header field. If no addresses are specified, print the contents of the From field. The -message SPEC (-msg SPEC) control argument causes this request to operate on specified messages instead of on the current one.

The from request function must be invoked on a single message and returns the From header field, or "" if there is none.

help (STRING)
Print information about the read_mail command. If specified, STRING is the name of a read_mail request or one of the topics "requests", "control_args", and "changes". If STRING is "*", print the list of available read_mail topics. If STRING is not specified, print introductory information on the use of read_mail followed by a list of topics.

in_reply_to (STRINGS) (-CONTROL_ARG),
irt (STRINGS) (-CONTROL_ARG)
If STRINGS are specified, replace the In-Reply-To field (if any) with the concatenation of STRINGS with intervening spaces. If no STRINGS are specified, print the contents of the In-Reply-To field. The -message SPEC (-msg SPEC) control argument causes this request to operate on specified messages instead of on the current message.

The in_reply_to (irt) request function must be invoked on a single message and returns the In-Reply-To field, or "" if there is none.

last (-CONTROL_ARG),
l (-CONTROL_ARG)
This request function returns the number of the last message that has not been deleted, or 0 if all messages have been deleted. If -all is specified, it returns the number of the last message whether or not that message has been deleted, or 0 if there are no messages.

list (SPECS),
ls (SPECS)
Print a summary line for each of the specified messages, for example:

```
     3. (52)   07/10/78 14:20  Sibert.PDO                subject
```

log (-CONTROL_ARG) (SPECS)
Save the specified messages in the user's log box, the mailbox named PERSON.sv.mbx in the home directory. Add Date and From header fields to those messages that do not have them. If the control argument -delete (-dl) is specified, delete the messages after logging them. If no messages are specified, operate on the current one.

message_id (SPECS)
mid (SPECS)
Print the Message-ID field of the specified messages, creating this field if necessary. If no messages are specified, use the current one.

The message_id (mid) request function must be invoked on a single message and returns the Message-ID field, or "" if there is none.

next (-CONTROL_ARG),
n (-CONTROL_ARG)
This request function returns the number of the next message that has not been deleted, or 0 if all messages have been deleted. If -all is specified, it returns the number of the next message whether or not that message has been deleted, or 0 if there is no next message.

previous (-CONTROL_ARG),
p (-CONTROL_ARG)
This request function returns the number of the last previous message that has not been deleted, or 0 if all messages have been deleted. If -all is specified, it returns the number of the previous message whether or not that message has been deleted, or 0 if there is no previous message.

print (-CONTROL_ARG) (SPECS)
pr (-CONTROL_ARG) (SPECS)
Print the specified messages. If the -header (-he) control argument is specified, print the header and the text. If -no_header (-nhe) or neither is specified, print only the text, preceded by a summary line as printed by the list request.

qedx (-CONTROL_ARG),
qx (-CONTROL_ARG)
Invoke the qedx editor on the current message. If -CONTROL_ARG is -header (-he), edit both the header and the text. If -CONTROL_ARG is -no_header (-nhe) or missing, edit only the message text. The w (write) request to qedx is not necessary to reflect changes to the actual message. An additional request, Q, is added to qedx to return to read_mail without reflecting any changes made during editing.

The -message SPEC (-msg SPEC) control argument can be followed by a message specifier indicating a single message, in which case the specified message is edited instead of the current one.

quit,
q
Exit the request loop and the read_mail command.

remove (ADDRESSES) (-CONTROL_ARGS)
At least one ADDRESS or one control argument must be specified.

Delete ADDRESSES appearing before the first -cc, -from, -reply_to or -to control argument from both the To and cc header fields.

Delete ADDRESSES following -cc, -from, -reply_to, or -to from the specified field. Delete the entire field if the control argument is followed by -all.

If any control argument is -in_reply_to (-irt), -message_id (-mid), or -subject (-sj), delete the appropriate field entirely.

The -message SPEC (-msg SPEC) control argument causes this request to operate on specified messages instead of on the current one.

The reply_to (rpt) request function must be invoked on a single message and returns the Reply-To field, or "" if there is none.

reply (SPECS) (-CONTROL_ARGS),
rp (SPECS) (-CONTROL_ARGS)

Allow the user to reply to the specified messages. A header is constructed for each reply, naming the sender of each message as the primary recipient of the reply and the recipients of each message as the secondary recipients of the reply. The text of all the replies is the same. Prompt for "Message:", accept a message text ending with ".", and then enter the send_mail request loop. When the quit request is issued to send_mail, the user is returned to read_mail.

A copy of the reply is automatically sent to the user's log box, unless -no_log is specified or -save PATH causes it to be saved elsewhere.

reply_to (ADDRESSES) (-CONTROL_ARG),
rpt (ADDRESSES) (-CONTROL_ARG)

Add the specified ADDRESSES to the Reply-To header field. If no ADDRESSES are specified, print the contents of the Reply-To field. The -message SPEC (-msg SPEC) control argument causes this request to operate on specified messages instead of on the current one.

The reply_to (rpt) request function must be invoked on a single message and returns the Reply-To field, or "" if there is none.

retrieve (SPECS)
rt (SPECS)

Cause the specified messages, if deleted, to be un-deleted. This action is allowed until the request loop is exited. When the user exits read_mail, all messages deleted by the delete (dl) request are actually deleted from the mailbox and cannot be retrieved.

save (-CONTROL_ARG) (SPECS) PATH,
sv (-CONTROL_ARG) (SPECS) PATH

Save the specified messages in the mailbox designated by PATH. The suffix .sv.mbx is added to PATH if it is not present. If the save box does not exist, the user is asked whether to create it. Date and From fields are automatically added to any messages that do not have them. These fields preserve information about the origin of the message. If the control argument -delete (-dl) is specified, delete the messages after saving them. If no messages are specified, operate on the current one.

subject (STRINGS) (-CONTROL_ARGS),
sj (STRINGS) (-CONTROL_ARGS)
If STRINGS are supplied, replace the Subject field (if any) with the concatenation of STRINGS with intervening spaces. If no STRINGS are specified, print the contents of the Subject field. The -message SPEC (-msg SPEC) control argument causes this request to operate on specified messages instead of on the current one.

The subject (sj) request function must be invoked on a single message and returns the Subject field, or "" if there is none.

to (ADDRESSES) (-CONTROL_ARGS)
Add the specified addresses to the To field. If no ADDRESSES are specified, print the contents of the To field. The -message SPEC (-msg SPEC) control argument causes this request to operate on specified messages instead of on the current one.

The to request function must be invoked on a single message and returns the To field, or "" if there is none.

write (SPECS) PATH,
w (SPECS) PATH
Append the specified messages to the ASCII segment designated by PATH. The suffix .mail is added to PATH if it is not present. If no messages are specified, write the current one. Add Date and From fields to any messages that do not have them. The -extend and -truncate (-tc) control arguments accepted by the file_output command can be specified.

# Appendix A

## ARPA Network Usage

This appendix describes the changes and extensions made to the read_mail and send_mail commands to accommodate messages sent over the ARPA Network.

These changes and extensions take effect on any system where the ARPA Network host number recorded in installation_parms is not equal to -1. They fall into three categories:

address format
header format
message transmission

Each category is discussed below.

### Addresses

Addresses are extended to include a designation of the ARPA Network host where the mailbox is located. An address can be either local or foreign. The -mailbox and -mailing_list forms of address are valid only for local addresses, although foreign addresses can be included in mailing lists. Foreign addresses must be of either the -user or the STRING form.

An extension is made to the -user and STRING forms of local address. If the mailbox >udd>PROJECT>PERSON>PERSON.mbx cannot be found, or if the STRING supplied as an address does not contain a period, the address is checked against the ARPA Network Mailer's address table. This table is simply a directory of links to mailboxes. Only if this check fails is an error message printed:

No mailbox for ADDRESS.

Three new address qualifiers (control arguments) are added to distinguish between local and foreign addresses. Two of these qualifiers are global in that they apply to all subsequent addresses in the command or request line. The third is local (like -comment) and is used to temporarily override the global qualifier presently in force.

-local This global qualifier specifies that all subsequent addresses on the command or request line are local addresses. The local system is the machine on which the user has invoked send_mail. This is the default global qualifier.

-host HOSTID
This global qualifier specifies that all subsequent addresses on the command or request line are foreign addresses on the ARPA Network system identified by HOSTID. HOSTID is any string acceptable to the host_id_ subroutine and can be an ASCII host name, a decimal number, or a hexadecimal number preceded by an & character. If HOSTID specifies the local system, this qualifier specifies that subsequent addresses are local addresses.

-at HOSTID
This local qualifier specifies that the immediately preceding address is a foreign address on the ARPA Network host identified by HOSTID. However, if HOSTID specifies the local system, the immediately preceding address is a local address. This qualifier can appear either before or after the -comment qualifier (if any) associated with this address.

For example, assuming that the local system is MIT-Multics, the command/request line fragment:

```
Palter.PDO -host MIT-MC GMP CRD Sibert.PDO
     -at MIT-Multics MAIL-USERS
```

specifies that Palter.PDO and Sibert.PDO are local addresses and GMP, CRD, and MAIL-USERS are foreign users on the MIT-MC system.

Header Format

The message header format chosen for use by the Multics mail system is very similar to the standard format for message headers described in the ARPA Network Protocol Handbook. As a result, only minor changes need be made to support the ARPA Network.

The message header parsing algorithm used by read_mail must be extended slightly to handle several differences in the syntax. These differences are not clearly defined and are not discussed here.

The only change to the format of message headers produced by send_mail is the format of the address entries in the From, Sender, Reply-To, To, cc, Redistributed-By, and Redistributed-To fields.

The three forms of address become:

```
STRING at HOSTID
"(mbx PATH)" at HOSTID
"(list PATH)" at HOSTID
```

where HOSTID is the official ARPA Network host name of the system on which the specified mailbox(es) reside.

All message headers produced by send_mail when using the ARPA Network contain the "at" phrase. To avoid unnecessary confusion, the various read_mail and send_mail requests that print the header or sections of the header do not print the "at" phrase if HOSTID names the local system.

For example, the sample header shown in the Headers section of the send_mail document is changed slightly to accommodate the ARPA Network:

```
Date:     25 May 1978 14:54-EDT
From:     Palter.PDO at MIT-Multics
Subject:  headers and other features of send_mail
To:       Sibert.PDO at MIT-Multics,
          Herbst.Multics at MIT-Multics,
          "(mbx >udd>PDO>Palter>mlsys.sv)"
               at MIT-Multics,
cc:       Greenberg.Multics at MIT-Multics
```

However, the to request of send_mail or read_mail still prints:

```
To: Sibert.PDO, Herbst.Multics,
    (mbx >udd>PDO>Palter>mlsys.sv)
```

## Message Transmission

The transmission of messages to local addresses is not changed by the addition of ARPA Network support. The user's process adds the message to the specified mailboxes immediately upon execution of the send request in send_mail or the forward request in read_mail.

Due to access and cost considerations, messages being transmitted across the ARPA Network may not be delivered by the user's process. If a message cannot be sent right away, the message complete with header is placed into a segment in a pool directory (>udd>Daemon>send_mail_pool) and this segment is queued for delivery by the ARPA Network Mailer. The user is informed of this fact by the message:

Mail queued for delivery to ADDRESS.

rather than the normal "Mail delivered..." message. When the ARPA Network Mailer succeeds or fails in delivering the message, it sends a message to the user.

Finally, the ARPA Network Mailer periodically examines its internal queues and deletes segments that are no longer of use from the pool.