MULTICS EXTENDED MAIL SYSTEM
USER'S GUIDE
ADDENDUM A

SUBJECT
Additions and Changes to the Manual

SPECIAL INSTRUCTIONS

This is the first addendum to CH23-02, dated December 1983.

Insert the attached pages into the manual according to the collating instructions on the back of this cover. Throughout the manual, change bars in the margins indicate technical changes and additions; asterisks denote deletions.

Refer to the Preface for "Significant Changes."

Note:

Insert this cover after the manual cover to indicate the manual is updated with Addendum A.

SOFTWARE SUPPORTED

Multics Software Release 12.0

ORDER NUMBER
CH23-02A

November 1986

Honeywell

46438
5C986
Printed in U.S.A.
COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

REMOVE

iii, iv
4-3 through 4-6
4-13, 4-14
A-13, A-14
A-27, A-28
A-37 through A-52

INSERT

iii, blank
4-3 through 4-6
4-13, 4-14
4-14.1, blank
A-13, A-14
A-14.1, blank
A-27, A-28
A-28.1, blank
A-37 through A-50
A-51, A-51.1
A-51.2, A-52

Honeywell disclaims the implied warranties of merchantability and fitness for a particular purpose and makes no express warranties except as may be stated in its written agreement with and for its customer. In no event is Honeywell liable to anyone for any indirect, special or consequential damages.

The information and specifications in this document are subject to change without notice. Consult your Honeywell Marketing Representative for product or service availability.
Terms within angle brackets (<...>) are used to convey the kind of word that you are to provide in the indicated space. For example, <User_id> means that you are to type a User_id. Any exceptions to this usage are noted.

Technical or other unfamiliar terms are CAPITALIZED when used for the first time, and are included in the glossary (Appendix C).

In examples, an exclamation point is used to indicate a line that you type at the terminal. You do not type the exclamation point, nor does Multics type it as a way of prompting you. It is strictly a typographical convention, to distinguish between typing done by you and typing done by Multics.

All commands, and most requests and control arguments, have short names. The short names are used in most examples throughout the manual.

Mail system messages are referred to as "ordinary messages," "messages" and "mail" in this manual. However, you will also encounter other types of messages as you work on Multics. "Interactive messages" are created by users with the send_message command. Messages from the Multics operating system are generally called "system notices". "Error messages" are also sent by the operating system, although these messages often begin with the name of the particular command that has been used incorrectly. Here are examples of all three of these types of messages:

interactive message ==> From Lotte.ProjDog 08/01/80 09:03 mst Fri: Hi
system notice ==> Mail delivered to Willow.
error message ==> read_mail: Entry not found. >udd>ProjCat>Willow>print.mbx

Significant Changes in CH23-02A

The read_mail command now provides a new feature that distinguishes if the message has been seen before. The switch_on and the switch_off requests are described in Section 4. Other requests pertaining to this capability are fully described in Appendix A.
LISTING AND PRINTING

The list Request

The list (ls) request serves as a handy reference tool in many situations. It provides a one-line summary of relevant information about each of your messages; this aids in deciding what you want to do with them. Here is a sample list summary from a mailbox with four messages:

<table>
<thead>
<tr>
<th>Msg#</th>
<th>Lines</th>
<th>Date</th>
<th>Time</th>
<th>From</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1*</td>
<td>(4)</td>
<td>08/01/80</td>
<td>09:14</td>
<td>Moch</td>
<td>picnic</td>
</tr>
<tr>
<td>2</td>
<td>(2)</td>
<td>08/01/80</td>
<td>10:26</td>
<td>Brie</td>
<td>and you?</td>
</tr>
<tr>
<td>3</td>
<td>(2)</td>
<td>08/01/80</td>
<td>13:02</td>
<td>Merce</td>
<td>your talk</td>
</tr>
</tbody>
</table>
| 4    | (27)  | 08/01/80 | 16:47  | Edgar | comments y<MORE>

The Message Number column shows the position of each message in this mailbox at this time. The Lines column includes only the lines of text in a message, not the number of header lines. The date and time that the message was sent to you are recorded also, as is the person who sent it to you. If the sender has included a subject, the Subject column includes as much of the subject that will fit on the rest of the line.

There are five flag characters which can occur in the columns after a message number when you use the list request:

<table>
<thead>
<tr>
<th>Column</th>
<th>flag</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>this message is the current message</td>
</tr>
<tr>
<td>1</td>
<td>!</td>
<td>this message has been deleted</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>this message was previously printed by the print request</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>this message will be acknowledged after it is printed</td>
</tr>
<tr>
<td>4</td>
<td>&amp;</td>
<td>this message cannot be deleted due to insufficient access</td>
</tr>
</tbody>
</table>
You can use the list request to give you a summary line about a single message; simply follow the request name with a message number:

```
read_mail: 1 ls 4
```

<table>
<thead>
<tr>
<th>Msg#</th>
<th>Lines</th>
<th>Date</th>
<th>Time</th>
<th>From</th>
<th>Subject</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4*</td>
<td>(27)</td>
<td>08/01/80</td>
<td>16:47</td>
<td>Edgar</td>
<td></td>
<td>y&lt;MORE&gt;</td>
</tr>
</tbody>
</table>

At the end of the summary line, "<MORE>" indicates that the title is longer than can fit on the line. Also notice the asterisk after message #4 — listing a message makes it become the current message.

The print Request

As noted above, the print (pr, p) request prints both header and text of the message or messages you specify. With a summary of messages in front of you, you can use the print request more effectively. If you have many messages, you can choose which message to print first, or you can decide not to read certain ones at this time.

MESSAGE SPECIFIERS

In order to print your messages so far, you have issued the print request followed by a message number. A message number is one of several MESSAGE SPECIFIERS: ways of indicating which messages you want to see.

Keywords

Another kind of message specifier is the keyword. These keywords are used just like message numbers:

- current (short name c)
- next (n)
- previous (p)
- first (f)
- last (l)
- all (a)
- seen
- unseen
- new
When you type "current" directly after the print request ("pr current"), you get the message that is currently being worked on by the read_mail command. The current message is always message #1 at first, and it shifts when you issue a request that deals with some other message; for example, when you first enter read_mail, message #1 is the current message, but when you type "print 2" then message #2 becomes the current one. You can also type simply "print" to see the current message.

The "next" and "previous" keywords refer to the messages relative to the current message, so they shift as the current message shifts. The "first" and "last" keywords operate on the first and last remaining messages in the mailbox.

The "seen" keyword identifies the messages that you have previously seen by using the print request. A capital S will precede the sender name. The keyword "unseen" specifies all messages not yet printed by the print request. The "new" keyword specifies all messages received since the last one printed with the print request.

There are also several combination keywords that indicate specific seen or unseen messages. For example, if you want to print the last message that you have already seen, you would use the last_seen message specifier. The complete list is:

```
first_seen,   (fs)
first_unseen, (fu)
last_seen    
last_unseen,  (lu)
next_seen,    (ns)
next_unseen,  (nu)
previous_seen, (ps)
previous_unseen, (pu)
```

For more details refer to the read_mail command in Appendix A.

Ranges

There are also several ways to print more than one message at a time. When you know exactly which messages you want to see, you may type several message numbers separated by spaces:

```
  ! p 3 1 4
```

The messages are printed in the order you specify.

If you want to see several messages in a row, you can specify a range by typing a message specifier for the earliest message you want, then a colon, and then a message specifier (no intervening spaces) for the last message you want, like this:

```
  ! pr 2:4
```

This prints messages #2, #3, and #4 for you. The keyword "all" prints all the undeleted messages in your mailbox.
When specifying a range, you can use any combination of the above-mentioned message specifier types. For example, assuming there are four messages in your mailbox and message #1 is the current message, all of the following expressions yield the same result:

\begin{verbatim}
print f:last
pr c:4
p all
print 1:3 last
\end{verbatim}

For further information on message specifiers, see Appendix A.

**print REQUEST CONTROL ARGUMENTS**

In some cases you know that you will not want to keep a particular message after you read it. The `-delete` (`-dl`) control argument is useful then:

\begin{verbatim}
read_mail: 1 p first -dl
\end{verbatim}

This request line is equivalent to:

\begin{verbatim}
read_mail: 1 p first;d first
\end{verbatim}

After the message you specify is printed out for you, it is deleted.

If you wish to bypass printing the full header when reading a message, you can supply the print request with its `-no_header` (`-nhe`) control argument. A shortened header is then printed before the text of the message, including only essential information:

\begin{verbatim}
read_mail: 1 pr 3 -nhe
\end{verbatim}

\#3 (2 lines in body):

\begin{verbatim}
I thought your talk this morning was good. If you would like more specific comments, let me know.
---(3)---
\end{verbatim}

\begin{verbatim}
read_mail:
\end{verbatim}
The -include_deleted control argument to the list request lists all messages, including deleted ones. An exclamation point beside a message number signifies a deleted message. Note that once message #2 is deleted, the current message automatically becomes #3.

The print request also has the -idl control argument, performing the parallel operation with deleted messages. If message #2 has been deleted, then this request line:

```plaintext
read_mail:  ! p 1:3
```

prints only messages #1 and #3, but this line:

```plaintext
read_mail:  ! p 1:3 -idl
```

prints messages #1, #2, and #3.

Remember: no message is truly gone until you issue the quit request. Once you leave read_mail, though, you can no longer retrieve deleted messages.

**MANIPULATING THE SEEN SWITCH**

The read_mail command automatically marks each "seen" message with an "S" flag. (A "seen" message is one that has been printed by means of the print request.) The switch_on and switch_off requests can be used to manipulate the seen switch on a message by message basis.

In some cases, you may want to turn the seen switch of a particular message off so that the next time you type "pr unseen" you will see the message again. The switch_off request will enable you to do this. Suppose the listed messages are:

<table>
<thead>
<tr>
<th>Msg#</th>
<th>Lines</th>
<th>Date</th>
<th>Time</th>
<th>From</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1S</td>
<td>(4)</td>
<td>06/01/86</td>
<td>09:14</td>
<td>Fritz</td>
<td>Meeting</td>
</tr>
<tr>
<td>2S</td>
<td>(2)</td>
<td>06/01/86</td>
<td>10:26</td>
<td>Fox</td>
<td>Lunch</td>
</tr>
<tr>
<td>3*</td>
<td>(2)</td>
<td>06/01/86</td>
<td>13:02</td>
<td>Herbst</td>
<td>Move</td>
</tr>
</tbody>
</table>
By typing:

! swf seen 2

This will turn off the seen switch for message 2. Now if you list your messages, the S flag denoting the message has been seen will have disappeared. Your listed messages will look like this:

<table>
<thead>
<tr>
<th>Msg#</th>
<th>Lines</th>
<th>Date</th>
<th>Time</th>
<th>From</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1S</td>
<td>(4)</td>
<td>06/01/86</td>
<td>09:14</td>
<td>Fritz</td>
<td>Meeting</td>
</tr>
<tr>
<td>2</td>
<td>(2)</td>
<td>06/01/86</td>
<td>10:26</td>
<td>Fox</td>
<td>Lunch</td>
</tr>
<tr>
<td>3*</td>
<td>(2)</td>
<td>06/01/86</td>
<td>13:02</td>
<td>Herbst</td>
<td>Move</td>
</tr>
</tbody>
</table>

Alternatively, you may either know the contents or not need to read an unseen message. In this case, you would turn on its seen switch without printing it. To do this:

! swn seen N

where N is the number of the message.

QUITTING

All you need to do to leave read_mail is type quit, or just q. But even the quit request has a couple of special features.

If you have been trying out various combinations of lists, message specifiers, deleting, and retrieving, you may be confused and worried about quitting and possibly deleting messages that you want to keep. Now is the time to use the -no_delete (-ndl) control argument of the quit request:

<too many requests>

read_mail:  ! q -ndl
r 11:43 0.343 133
This discards all modifications that you have made during this session with read_mail. Next time you enter read_mail you will find your mailbox just the way you found it this time (plus any messages that have arrived since then). This control argument can be better than aspirin.

Sometimes when you issue the quit request you receive a note like this:

```
read_mail (quit): A new message has arrived. Do you still wish to quit?
```

You must answer either yes, in which case you are returned to command level, or no, which gives you another read_mail prompt. If you use the -force (-fc) request control argument with quit:

```
read_mail: ! q -fc
r 11:43 0.0703 286
```

you are returned to command level with no questions asked.

**ASSISTANCE**

The read_mail command has several means of assistance available while you are working.

**The ? Request**

When you forget the name of a request, or which letter is the short name for what request, type the ? request. It prints a multi-columnar list of all requests and their short names. Here is an abbreviated version of the ? request and response, listing only the requests discussed so far in this section:

```
read_mail: ! ?
Available read_mail requests:
help    print, pr, p   retrieve, rt   forward, fwd, for
quit, q list, ls     delete, dl, d   reply, rp
Type "list_requests" for a short description of the requests.
read_mail:
```
This page intentionally left blank.
first_seen
fs
  refers to the first undeleted "seen" message (see "seen" below).

first_unseen
fu
  refers to the first undeleted "unseen" message (see "unseen" below).

last_seen
  refers to the last undeleted "seen" message.

last_unseen
lu
  refers to the last undeleted "unseen" message.

next_seen
ns
  refers to the next undeleted "seen" message.

next_unseen
nu
  refers to the next undeleted "unseen" message.

previous_seen
ps
  refers to the previous undeleted "seen" message.

previous_unseen
pu
  refers to the previous undeleted "unseen" message.

Ranges of messages can be identified by two message numbers or keywords separated by a colon (:). For example, the following line:

3:last

identifies all messages of the appropriate type from message #3 through the last message of the appropriate type in the mailbox.

The following keywords specify collection of messages:

all
  is accepted as shorthand for "first:last"; it identifies all messages of the appropriate type in the mailbox.

unseen
  refers to all messages of the appropriate type that have not yet been printed by the print request.
seen
refers to all messages of the appropriate type that have previously been
printed by the print request.

new
refers to all messages of the appropriate type after the last "seen" message,
that is, after the latest one that was printed previously by the print request.

Message numbers can be added and subtracted using "+" and "-". For example, if
the current message is #20, the following line:

current-5:current+10

identifies all messages of the appropriate type from message #15 through #30. As this
example demonstrates, arithmetic operations are performed after any message keywords
are converted to absolute numbers.

Qedx regular expressions can be used to select all messages of the appropriate type
that contain a given string. The regular expression must be enclosed in slashes (/); for
an explanation of the syntax of regular expressions, see the Qedx Text Editor's User
Guide, Order No. CG40. If the regular expression contains spaces, horizontal tabs,
quotes ("), parentheses, or brackets, the entire expression must be enclosed in quotes to
avoid misinterpretation by the request line processor; any quotes within the regular
expression must be doubled. For example,

"/said, ""I think/"

matches any message that contains the string:

said, "I think

A regular expression can be preceeded by one of the keywords listed above to
select the first, last, etc. message containing that string. Additionally, two or more
regular expressions can be combined by connectors to express logical AND (&) and
logical OR (|). For example, the following line:

last/artificial/|intelligence/

specifies the last message of the appropriate type containing both of the strings
"artificial" and "intelligence".
Message Selection Control Arguments

The list, print, print_header, delete, and retrieve requests accept several control arguments that supply further criteria for message selection. If no message specifiers are given, all messages of the appropriate type in the mailbox are considered for selection. For example, the request line:

\texttt{! list 23:30 -from Ellery}

lists all non-deleted messages in the mailbox from message \#23 through \#30 that were sent by the user Ellery.

Selection control arguments are divided into four classes — subject selection, time selection, author selection, and recipient selection. If several control arguments from one class are provided, a message must only satisfy one of the selections in that class to be considered by the request. If control arguments from more than one class are provided, a message must satisfy one of the selections in all of these classes provided to be considered by the request. For example, the request line:

\texttt{! list -from Ellery -from Green -after 1/1/82}

lists all non-deleted messages in the mailbox that were: a) sent by either Ellery or Green, and b) sent any time from January 1982 to the present. A message sent by Ellery on 23 December 1981 would not be listed by this request.
This page intentionally left blank.
exec_com path {args}
ec path {args}

executes a program written in the exec_com language, where path is the pathname of an exec_com program. The suffix "rdmec" is added to the pathname if necessary. This program is used to pass request lines to read_mail and to pass input lines to requests that read input. Currently, any errors detected during an ec execution within read_mail will abort the request line in which the ec request was invoked. The arguments are optional arguments to the exec_com program and are substituted for parameter references in the program such as &1.

If the pathname does not contain a "<" or">
" character, read_mail searches for the exec_com program using the mail_system search list. The default content of this search list is:

-working_dir
>udd>[user project]>[user name]>[user name].mlsys

When evaluating a read_mail exec_com program, subsystem active requests are used rather than Multics active functions when evaluating the &[...] construct and the active string in an &if statement. The read_mail execute active request may be used to evaluate Multics active strings within the exec_com.

[exec_com path {args}]  
[ec path {args}]

executes a program written in the exec_com language that specifies a return value of the exec_com request by use of the &return statement. The arguments are the same as for the exec_com request.

execute STR

e STR

executes the supplied line as a Multics command line, where STR is the Multics command line to be executed or the Multics active string to be evaluated. It need not be enclosed in quotes.

The recommended method to execute a Multics command line from within read_mail is the ".." escape sequence. The execute request is intended as a means of passing information from read_mail to the Multics command processor.

All O, [], and "'s in the given line are processed by the read_mail request processor, not the Multics command processor. Thus, the values of subsystem active requests may be passed to Multics commands when using the execute request. For example, the following request line lists the ACL of the mailbox being read by the current invocation of read_mail.

l embla [mailbox]
read_mail (rdm)

[execute STR]
[e STR]
evaluates a Multics active string from within read_mail. For example, the following read_mail request line:

! write all [e strip_entry [mailbox]]

writes the ASCII representation of all messages in the mailbox into a segment in the working directory whose entry name is the same as that of the mailbox, with the "mbx" suffix changed to "mail".

first {--ca}
f {--ca}
prints the number of the first message of the specified type. The control argument may be one of the following:

--include_deleted
--id
prints the number "1" (i.e., the number of the first message, whether or not it has been deleted.)

--only_deleted
--odl
prints the message number of the first deleted message.

--only_non_deleted
--ondl
prints the message number of the first non-deleted message. This is the default.

[first {--ca}]
[f {--ca}]
returns the number of the first message of the specified type. If there are no messages of the specified type, it returns the value zero. This active request takes the same control arguments as the first request.

first_seen
fs
prints the message number of the first message printed with the print request.

[first_seen]
[fs]
returns the message number of the first message printed with the print request.
first_unseen
fu
   prints the message number of the first message NOT yet printed with the print request.

[first_unseen]
[fu]
   returns the message number of the first message NOT yet printed with the print request.

forward {spec} addresses {~ca}
fwd {spec} addresses {~ca}
for {spec} addresses {~ca}
   forwards the specified message(s) to the stated recipients. Forwarding addresses may be given in any of the forms described under "Addresses" in the send_mail command description (later in this appendix).

The forward request will acknowledge any message(s) requiring acknowledgement, unless --no_acknowledge is specified on the read_mail command line.
This page intentionally left blank.
read_mail (rdm)

skip {~scn} {~seen}
  s {~scn} {~seen}
  skips to the next paragraph. If ~section or ~scn is given, help skips all
  paragraphs of the current section. If ~seen is given, help skips to the next
  paragraph that the user has not seen. Only one control argument is allowed in
  each skip response.

title {~top}
  lists titles and line counts of the sections that follow; if ~top or ~t is given,
  help lists all section titles. The previous question is repeated after titles are
  printed.

yes
  y
  prints the next paragraph of information on this topic.

if [EXPR] ~then LINE1 {~else LINE2}
  conditionally executes one of two request lines depending on the value of an
  active string. The arguments are:

  EXPR
  is the active string that must evaluate to either "true" or "false". The active
  string is constructed from read_mail active requests and Multics active strings
  (using read_mail's execute active request).

  LINE1
  is the read_mail request line to execute if EXPR evaluates to "true". If the
  request line contains any request processor characters, it must be enclosed in
  quotes.

  LINE2
  is the read_mail request line to execute if EXPR evaluates to "false". If
  omitted and EXPR is "false", no additional request line is executed. If the
  request line contains any request processor characters, it must be enclosed in
  quotes.

  [if [EXPR] ~then STR1 {~else STR2}]
  returns one of two character strings to the read_mail request processor, depending
  on the value of an active string. The arguments are:

  EXPR
  is the active string that must evaluate to either "true" or "false". The active
  string is constructed from read_mail active requests and Multics active strings
  (using read_mail's execute active request).

  STR1
  is returned as the value of the if active request if the EXPR evaluates to
  "true".
read_mail (rdm)

STR2

is returned as the value of the if active request if the EXPR evaluates to "false". If omitted and the EXPR is "false", a null string is returned.

last {-ca}
1 {-ca}

prints the number of the last message of the specified type. The control argument may be one of the following:

-include_deleted
-idl

prints the number of the last message, whether or not it has been deleted.

-only_deleted
-odl

prints the number of the last deleted message.

-only_non_deleted
-ondl

prints the message number of the last non-deleted message. This is the default.

[last {-ca}]
[1 {-ca}]

returns the number of the last message of the specified type. If there is no message of the specified type, it returns the value zero. This active request takes the same control arguments as the last request.

last_seen

prints the message number of the last message printed with the print request.

[last_seen]

returns the message number of the last message printed with the print request.

last_unseen
lu

prints the message number of the last message NOT yet printed with the print request.

[last_unseen]
[lu]

returns the message number of the last message NOT yet printed with the print request.
read_mail (rdm)

list {spec} {–selca} {–ca}
ls {spec} {–selca} {–ca}
prints a summary line for each of the specified messages, or for all undeleted
messages if no specifiers are given. Control arguments may be chosen from the
following:

–delete
–dl
  deletes the messages after listing them.

–header
–he
  prints a header line before the list of messages. This is the default.

–include_deleted
–idl
  prints the list of messages, including deleted ones.

–line_length N
–ll N
  prints the list of messages, using the supplied line length N to determine
  where and if to truncate the message subject. (The default length is the
  terminal’s line length.)

–no_delete
–ndl
  does not delete the messages after listing them. This is the default.

–no_header
–nhe
  omits the header line preceding the list of messages.

–no_line_length
–nll
  does not truncate the message subject unless the subject is more than one line
  long.

–no_reverse
–nrv
  lists the messages in ascending numeric order. This is the default.

–only_deleted
–odl
  lists only deleted messages.

–only_non_deleted
–ondl
  lists only non-deleted messages. This is the default.
-reverse
-rev

prints the list of messages in reverse order.

One or two lines are printed for each message. The format of the first line is:

NFFFF (L) MM/DD/YY HH:MM AUTHOR SUBJECT

The N is the message number and L is the number of lines in the body of the message (excluding the header). The F's are optional flags and are described below. MM/DD/YY HH:MM specifies the date/time when the message was originally transmitted. AUTHOR specifies the original author(s) of the message and is normally the part of the author's name which will fit in the space provided; if the first author does not have a name, the printed representation of his address will be used. The authors of messages are listed in the From Field. SUBJECT defines the contents on the remainder of the line. If the message is an interactive message, and does not contain a subject, the actual text of the message that will fit on the remainder of the line will be used.

There are five flag characters which can occur in the columns after a message number when you use the list request:

<table>
<thead>
<tr>
<th>Column</th>
<th>flag</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>this message is the current message</td>
</tr>
<tr>
<td>1</td>
<td>!</td>
<td>this message has been deleted</td>
</tr>
<tr>
<td>2</td>
<td>$</td>
<td>this message was previously printed by the print request</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>this message will be acknowledged after it is printed</td>
</tr>
<tr>
<td>4</td>
<td>&amp;</td>
<td>this message cannot be deleted due to insufficient access</td>
</tr>
</tbody>
</table>

If the message has been forwarded, a second line is included in the listing. This line has the format:

(*N) Forwarded (Nth time) at MM/DD/YY HH:MM by STR

where N indicates the number of times that this message has been forwarded. (N is omitted if the message has only been forwarded once.) MM/DD/YY HH:MM specifies the date/time that the message was last forwarded, and is derived from the most recent Redistributed-Date field. STR specifies the person who last forwarded the message, and is the contents of the most recent Redistributed-From field in the message.
read_mail (rdm)

[list {spec} {selca} {ca}]
[ls {spec} {selca} {ca}]
 returns a list of the numbers of the specified messages separated by spaces. This
active request takes the same selection arguments and control arguments as the list
request.

list_help {topics}
lh {topics}
displays the name of all read_mail information segments on given topics. If no
topics are given, all read_mail information segments are listed.

When matching topics with info segment names, an info segment name is
considered to match a topic only if that topic is at the beginning or end of a
word within the segment name. Words in info segment names are bounded by the
beginning and end of the segment name and by the characters period (.), hyphen
(-), underscore (_), and dollar sign ($). The ".info" suffix is not considered when
matching topics.

list_requests {STR} {ca}
lr {STR} {ca}
prints a brief description of selected read_mail requests, where STR specifies the
request(s) to be described. Any request with a name containing one of these
strings is listed unless -exact is used, in which case the request name must exactly
match one of these strings. When matching STRs with request names, a request
name is considered to match a STR only if that STR is at the beginning or end
of a word within the request name. Words in request names are bounded by the
beginning and end of the request name and by the characters period (.), hyphen
(-), underscore (_), and dollar sign ($).

Control arguments are:

-all
-a
 includes undocumented and unimplemented requests in the list of requests
 eligible for matching the STR arguments.

-exact
 lists only those requests one of whose names exactly match one of the STR
 arguments.

log {spec} {ca}
saves the specified messages in the user's logbox. The user's logbox has the
pathname >udd>Project_id>Person_id>Person_id.sv.mbx. It is created automatically
if it does not already exist, and the user is informed of its creation. Date and
From header fields are added as required to logged messages. Any messages
requiring acknowledgement are acknowledged unless -no_acknowledge is specified
on the read_mail command line. Control arguments for this request are the same
as for the append request.
mailbox
mbx
prints the absolute pathname of the mailbox currently being read.

[mailbox]
[mbx]
returns the absolute pathname of the mailbox currently being read.

new
prints the message numbers of all messages received since the last printed with the print request.

[new]
returns the message numbers of all messages received since the last one printed with the print request.

next {-ca}
prints the number of the next message of the specified type. The control argument may be one of the following:

--include_deleted
--idl
prints the number of the next message in the mailbox, whether or not it has been deleted.

--only_deleted
--odl
prints the number of the next deleted message.

--only_non_deleted
--ondl
prints the number of the next non-deleted message. This is the default.

[next {-ca}]
returns the number of the next message number of the specified type. If there are no messages of the specified type, the value zero is returned. This active request takes the same control arguments as the next request.

next_seen {msg_spec}

ns {msg_spec}
prints the message number of the next "seen" message following the specified message. (A "seen" message is one previously printed by the print request.)

[next_seen {msg_spec}]
[ns {msg_spec}]
returns the message number of the next "seen" message following the specified message. (A "seen" message is one previously printed by the print request.)
read_mail (rdm)

next_unseen {msg_spec}
nu {msg_spec}
prints the message number of the next "unseen" message following the specified message. (An "unseen" message is one not previously printed by the print request.)

[next_unseen {msg_spec}]
[nu {msg_spec}]
returns the message number of the next "unseen" message following the specified message. (An "unseen" message is one not previously printed by the print request.)

preface {spec} path {--ca}
prf {spec} path {--ca}
same as the append request, but inserts messages at the beginning of the ASCII segment specified by path, rather than at the end.

previous {--ca}
prints the number of the previous message of the specified type. The control argument may be one of the following:

--include_deleted
--idl
prints the number of the previous message, whether or not it has been deleted.

--only_deleted
--odl
prints the number of the previous deleted message.

--only_non_deleted
--ondl
prints the number of the previous non-deleted message. This is the default.

[previous {--ca}]
returns the number of the previous message of the specified type. If there is no message of the specified type, the value zero is returned. This active request takes the same control arguments as the previous request.

previous_seen {msg_spec}
ps {msg_spec}
prints the message number of the "seen" message immediately preceding the specified message. (A "seen" message is one printed by the print request.)

[previous_seen {msg_spec}]
[ps {msg_spec}]
returns the message number of the "seen" message immediately preceding the specified message. (A "seen" message in one printed by the print request.)
read_mail (rdm)

previous_unseen {msg_spec}
pu {msg_spec}
  prints the message number of the "unseen" message immediately preceding
  the specified message. (A "unseen" message is one NOT yet printed by the print
  request.)

[previous_unseen {msg_spec}] 
[pu {msg_spec}]
  returns the message number of the "unseen" message immediately preceding
  the specified message. (A "unseen" message is one NOT yet printed by the print
  requests.)

print {spec} {-selca} {-ca}
pr {spec} {-selca} {-ca}
p {spec} {-selca} {-ca}
  prints the specified messages. This request causes the specified messages to be
  acknowledged, if requested by the sender, unless -no_acknowledge is specified on
  the read_mail command line.

If you use this request while in the video system (documented in the
Programmer's Reference Manual, Order No. AG91), the reset_more control order
is issued after each message is printed. This allows users of the video system to
easily abort the printing of a single message, when printing several messages.

Control arguments may be chosen from the following:

- brief_header
- bfh
  specifies that the minimal amount of information from the message header
  should be displayed. The date and authors are always displayed; the subject is
  displayed if it isn't blank; the number of recipients is displayed either if there
  is more than one recipient or if the user is not the sole recipient of the
  message. If the message was ever forwarded with comments, these comments
  are also displayed.

- delete
- dl
  deletes the specified messages upon exiting read_mail, if all the specified
  messages are successfully printed.

- header
- he
  specifies that all information from the message header should be displayed,
  including user-defined fields but excluding the message trace and redundant
  information. This is the default.

- include_deleted
- idl
  prints the messages, whether or not they have been deleted.
read_mail (rdm)

-long_header
-lghe
specifies that all information from the message header including network tracing information should be displayed, even if some of the information is redundant. (In other words, if the From, Sender and Delivery-By fields are all equal, this option will force the print request to display all three fields.)

-no_delete
-ndl
does not delete the specified messages upon exiting read_mail. This is the default.

-no_header
-nhe
specifies that absolutely no information from the message header should be displayed. Only the message number, message body line count, and message body will be displayed.

-no_reverse
-nrv
prints the messages in ascending numeric order. This is the default.

-only_deleted
-oal
prints only the deleted messages.

-only_non_deleted
-onal
prints the non-deleted messages. This is the default.

-reverse
-rev
prints messages in reverse order.

print_header {spec} {--selca} {--ca}
prhe {spec} {--selca} {--ca}
prints only the header of the specified message. This request causes the specified messages to be acknowledged if requested by the sender, unless --no_acknowledge is specified on the read_mail command line. Control arguments may be chosen from the following:

-brief
-bf
specifies that the minimal amount of information from the message header should be displayed. The date and authors are always displayed; the subject is displayed if it isn't blank; the number of recipients is displayed eithe: if there is more than one recipient or the user is not the sole recipient of the message. If the message was ever forwarded with comments, these comments are also displayed.
read_mail (rdm)

-default
-dft
specifies that all information from the message header should be displayed, including user-defined fields but excluding the message trace and redundant information. This is the default.

-delete
-dl
deletes the specified messages upon exiting read_mail, if all the specified messages are successfully printed.

-include_deleted
-idl
prints the messages, whether or not they have been deleted.

-long
-lg
specifies that all information from the message header including network tracing information should be displayed, even if some of the information is redundant. (In other words, if the From, Sender and Delivery-By fields are all equal, this option will force the print_header request to display all three fields.)

-no_delete
-nl
does not delete the specified messages upon exiting read_mail. This is the default.

-no_reverse
-nrv
prints the messages in ascending numeric order. This is the default.

-only_deleted
-odl
prints only the deleted messages.

-only_non_deleted
-ondl
prints the non-deleted messages. This is the default.

-reverse
-rev
prints messages in reverse order.

quit {-ca}
q {-ca}
exits the read_mail command; any requested deletions are actually performed at this point. Control arguments may be chosen from the following:
read_mail (rdm)

-delete
-dl
  deletes the specified messages upon exiting read_mail. This is the default.

-force
-fc
  does not check for newly arrived messages before returning to command level.

-no_delete
-ndl
  does not delete the specified messages upon exiting read_mail.

-no_force
-nfc
  queries the user for permission to exit read_mail if there are newly arrived
  messages. This is the default.

ready
rdy
  prints a Multics ready message. The Multics general_ready command may be used
to change the format of the ready message printed by this request, and also after
execution of request lines if the ready_on request is used. The default ready
message gives the time of day, the amount of CPU time, and page faults used
since the last ready message was typed.

ready_off
rdf
  does not generate a ready message after the execution of each request line. This
  is the default.

ready_on
rdn
  causes a ready message to be printed after the execution of each request line.

reply {spec} {-ca} {-to addresses} {-ca more_addresses}
rp {spec} {-ca} {-to addresses} {-ca more_addresses}
  allows the user to reply to the specified messages. By default, the reply is sent
  only to the authors of the original messages. The reply is created in send_mail;
  the user is returned to read_mail after the message is sent. (The In-Reply-To
  field is initialized with the appropriate set of references before send_mail is
  invoked.) This request acknowledges any messages requiring acknowledgement unless
  -no_acknowledge is specified on the read_mail command line.

Control arguments for the reply request are:

-bcc addresses
  specifies the "blind" recipients of the reply.
-cc addresses
sends a copy of the reply to the specified addresses. The given addresses
become the only secondary recipients of the reply unless the -include_recipients
control argument is also included.

-delete
-deletes the messages after replying to them. However, if you exit send_mail
without sending the reply, this control argument is ignored.

-include_authors
-iat
includes the author(s) of the original message as primary recipient(s) of the
reply. This is the default, unless -to is also specified, in which case this
argument must be explicitly specified if the author(s) are to receive the reply.

-include_deleted
-idl
includes all messages in the mailbox, whether or not they have been deleted,
when processing the message_specifiers to determine which messages will be
answered.

-include_original
-10
includes the text and the Date, From, and Subject fields of the messages being
replied to as part of the text of the reply. This text is indented four spaces
if no indentation is explicitly specified.

-include_recipients
-irc
includes all recipients of the original message as secondary recipients of the
reply.

-include_self
-is
allows a copy of the reply to be sent to the author of the reply if it is
determined that such a copy should be sent from the use of the
-include_authors or -include_recipients control arguments.

-indent N
-ind N
indents the text of the original message by N spaces in the reply when
-include_original is specified. The default is 4 spaces.

-notify
-nt
specifies that the mail system should send a "You have mail." notification to
each recipient of the reply message. This is the default.
-no_delete
-no

does not delete the messages. This is the default.

-no_include_authors
-ns

does not include the author(s) of the original message as primary recipients of the reply.

-no_include_original
-nio

does not include the original messages as part of the text of the reply. This is the default.

-no_include_recipients
-nirc

does not include the recipients of the original message as secondary recipients of the reply. This is the default.

-no_include_self
-nis

specifies that a copy of the reply is sent to the author of the reply only if this is explicitly requested by use of the -to or -cc control arguments. This is the default. This default allows the user to create a reply abbreviation that automatically logs the reply without receiving an extra copy whenever -include_recipients is specified.

-no_notify
-nnt

specifies that the mail system should not send notification messages to the recipients of the reply message.

-no_refill
-nrfi

does not reformat the original text. This is the default.

-only_deleted
-odl

includes only deleted messages when processing the message_specifiers to determine which messages will be answered.

-only_non_deleted
-ondl

includes only non-deleted messages when processing the message_specifiers to determine which messages will be answered. This is the default.

-refill
-rfi

reformats the original text to fit within the line length of the reply.
-to addresses

sends a copy of the reply to the specified addresses. The -to control argument overrides the -include_authors default, so the given addresses become the only primary recipients of the reply unless the -include_authors control argument is also included.

The following send_mail control arguments can also be used on the reply request line:

-abbrev, -ab
-abort
-acknowledge, -ack
-brief, -bf
-fill, -fi
-from addresses
-input_file path, -if path
-line_length N, -ll N
-log
-long, -lg
-message_id, -mid
-no_abbrev, -nab
-no_abort
-no_acknowledge, -nack
-no_fill, -nfi
-no_log
-no_message_id, -nmid
-no_prompt
-no_request_loop, -nrql
-no_subject, -nsj
-profile_path, -pf path
-prompt STR
-reply_to addr, -rpt addr
-request STR, -rq STR
-request_loop, -rql
-save path, -sv path
-subject STR, -sj STR
-terminal_input, -ti

(For the -reply_to control argument in the above list, "addr" means "addresses").

Notes on recipients:
By default, the reply is sent only to the authors of the original messages or to those recipients specified by the authors to receive replies in place of the authors.
In the following text, the term "authors of the original messages" means either the authors or their designated agents.

The -to and -include_authors control arguments specify the primary recipients for the reply. If the -to control argument is used and -include_authors does not appear on the request line, only those addresses specified after -to are used as the primary recipients of the reply. If both -to and -include_authors are used on the request line, the primary recipients of the message are the authors of the original messages and the addresses specified after the -to control argument. Use of -include_authors on the read_mail command line does not affect this interaction of -to and -include_authors on the reply request line.

The -cc and -include_recipients control arguments specify the secondary recipients for the reply. If -include_recipients is specified either on the reply request line or the read_mail command line, all recipients of the original messages are included as secondary recipients of the reply. If -cc is used on the request line, the addresses following the -cc control argument are added to the list of secondary recipients of the reply. For example, the command line:

! read_mail -include_recipients

in conjunction with the request line
I reply -to Smith -cc Riley

composes a reply for the current message that is sent to Smith as the sole primary recipient and to all the recipients of the current message plus Riley as the secondary recipients.

Notes:
Unless overridden by use of the -abbrev, -no_abbrev, or -profile control arguments, the send_mail invocation created by this request has the same state of request line abbreviation expansion and uses the same profile as the current read_mail invocation.

Unless overridden by use of the -subject or -no_subject control arguments, this request constructs a subject for the reply message by combining the subjects of all the original messages. Additionally, the subject is prefixed by the string "Re: ".

This request constructs an In-Reply-To field for the reply message identifying the original messages being answered by this reply.

retrieve {spec} {selca}
rt {spec} {selca}
causes the specified messages, if deleted, to be undeleted. This action is allowed until the user quits and returns to command level. When the user exits read_mail, all messages deleted by the delete (dl) request are actually deleted from the mailbox and can no longer be retrieved.

save {spec} path {ca}
sv {spec} path {ca}
saves the specified messages in the mailbox designated by path. The .sv.mbx suffix is added to path if it is not present. If the savebox 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. If no messages are specified, the current one is saved. This request causes the specified messages to be acknowledged if requested by the senders, unless -no_acknowledge is specified on the read_mail command line. Control arguments are the same as for the append request.

seen
prints the message numbers of all the messages that have previously been printed by the print request.

[seen]
returns the message numbers of all the messages that have previously been printed by the print request.
subsystem_name
   prints the name of the current subsystem.

[subsystem_name]
   returns the name of the current subsystem. This active request is useful as part of
   an abbrev that is shared by multiple subsystems.

subsystem_version
   prints the version of the current subsystem.

[subsystem_version]
   returns the version of the current subsystem. This active request may be used in
   an abbrev that is shared by multiple subsystems.

switch_on switch_name {msg_spec}
swn switch_name {msg_spec}
   turns on a specified switch for each selected message.

switch_name
   currently must be the keyword "seen". If the "seen" switch is "on," the
   message will be identified by the system as a "seen" message when it is
   printed by the print request. (An "S" flag will appear in the one-line
   summary of information provided by the list request.)

message_specifiers
   identifies messages for which the switch is to be turned on. If no
   message_specifiers are used, the current message is assumed.

switch_off switch_name {msg_spec}
swf switch_name {msg_spec}
   turns off a specified switch for each selected message.

switch_name
   currently must be the keyword "seen". If the "seen" switch is "off," the
   system will not identify a message printed by the print command as a "seen"
   message.

message_specifiers
   identifies messages for which the switch is to be turned off. If no
   message_specifiers are used, the current message is assumed.

unseen {msg_spec}
   prints the message numbers of all messages that have not been printed with the
   print request.

[unseen {msg_spec}]  
   returns the message numbers of all messages that have not been printed with the
   print request.
read_mail (rdm)

write {spec} path {--ca}
w {spec} path {--ca}
    appends the specified messages to the ASCII segment designated by path. The .mail suffix is added to path if it is not present. If no messages are specified, the current one is written. Date and From fields are added to any messages that do not have them. This request causes the specified messages to be acknowledged if requested by the senders unless --no_acknowledge is specified on the read_mail command line. Control arguments may be chosen from the following:

--delete
--dl
    deletes the messages after writing them, if all the write operations are successful.

--extend
    writes the messages at the end of the segment. This is the default.

--include_deleted
--idl
    writes the messages, whether or not they have been deleted.

--no_delete
--ndl
    does not delete the messages after writing them. This is the default.

--no_reverse
--nr
    writes the messages in ascending numeric order. This is the default.

--only_deleted
--odl
    writes only the deleted messages.

--only_non_deleted
--ondl
    writes the non-deleted messages. This is the default.

--reverse
--rev
    writes the messages in reverse order.

--truncate
--tc
    truncates the segment before writing the messages to it.
send_mail (sdm)

Name: send_mail (sdm)

SYNTAX AS A COMMAND

sdm {addresses} {-ca}

FUNCTION

The send_mail command transmits a message to one or more recipients. The message is automatically prefixed by a header whose standard fields give the author(s), the intended recipients, and a brief summary of the contents.

ARGUMENTS

addresses

specifies the primary recipients of the message. By default, the message has no primary recipients. Addresses can be specified in one or more of the following forms:

-log

specifies the user's logbox and is equivalent to:

-mailbox >udd>Project_id>Person_id>Person_id.sv.mbx

This address is included as a "blind" recipient of the message.

-mailbox PATH

-mbx PATH

specifies the pathname of a mailbox. The .mbx suffix is assumed if it is not present.

-mailing_list PATH

-mls PATH

specifies the pathname of a mailing list. The .mls suffix is assumed if it is not present. The archive component pathname convention is accepted.

-meeting PATH

-mtg PATH

specifies the pathname of a forum meeting. The .control suffix is assumed if it is not present. If the pathname given is just an entryname (i.e., no "<" or ">" characters appear in the pathname), the user's forum search paths are used to find the meeting.

-save PATH

-sv PATH

specifies the pathname of a savebox. The .sv.mbx suffix is assumed. This address is included as a "blind" recipient of the message.
HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE
MULTICS EXTENDED MAIL SYSTEM
USER'S GUIDE
ADDENDUM A

ORDER NO. CH23-02A
DATED NOVEMBER 1986

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

PLEASE FILL IN COMPLETE ADDRESS BELOW.

FROM: NAME ___________________________ DATE ____________

TITLE ___________________________

COMPANY ___________________________

ADDRESS ___________________________

______________________________

______________________________