LEVEL 68

MULTICS CARRY FACILITY

SUBJECT

Description of the Multics Carry Facility for Transferring Data from one Multics Site to Another

SPECIAL INSTRUCTIONS

This manual supersedes AN76, Revision 1, dated November 1978 and its Addendum A. Change bars in the margin indicate technical changes and additions; asterisks denote deletions.

SOFTWARE SUPPORTED

Multics Software Release 9.0

ORDER NUMBER

AN76-02

February 1981

Honeywell

Preface

This manual contains a description of the Multics Carry Facility and the practical aspects of its application. The facility is a useful tool that provides the means for transferring data (segments, multisegment files, and directory subtrees) between two Multics sites.

Reference is made in this manual to an individual designated the carry administrator. Depending on the needs and resources of a particular site, these individuals might be the system administrator or one of any number of qualified individuals to whom the system administrator has delegated this responsibility. In either case, the carry administrator requires certain privileged access to the system in order to perform administrative functions for the carry facility. The access requirements are detailed under "Carry Administrator Commands," Section 5.

Users of this manual should be familiar with some of the concepts and terminology of the Multics system. The following Multics user documentation should be consulted:

Manual Name	*Order No.*	*Text Reference*
Multics Administrator Manual, System	AK50	MAM System
Multics Programmers' Manual:		
Reference Guide	AG91	MPM Reference Guide
Commands and Active Functions	AG92	MPM Commands
Subroutines	AG93	MPM Subroutines
Subsystem Writers' Guide	AK92	MPM Subsystem Writers' Guide
Peripheral Input/Output	AX49	MPM Peripheral I/0
Communications Input/Output	CC92	MPM Communications I/O

Throughtout this manual, reference is frequently made to a number of the above manuals. For convenience, references to these manuals in text are in the form: "as described in the MPM Commands," or "see the MPM Subroutines."

File No.: 1L13 AN76-02

Changes to Multics Carry Facility, Revision 2

Section 2, "Carry Administrator," is new to this manual. The information formerly contained in Section 1, the introduction, is now divided between Section 1, which contains information pertinent to all users of the Carry Facility, and Section 2, which contains information pertinent to carry administrators.

In Sections 3 and 4, the sample exec_coms and the examples shown have been updated to denote changes in the exec_coms.

In Section 5, the administrator commands section, numerous new control arguments have been added to the carry_load command and the carry_retrieve command.

In Section 6, the user command section, numerous new control arguments have been added to the following command descriptions:

cancel_carry_request
enter_carry_request
list_carry_requests

CONTENTS

SECTION 1

INTRODUCTION

The Multics carry facility is a means for transferring files and directory subtrees from one Multics site to another. This manual describes the procedures used to "carry" information, i.e., copy (dump) the information to be transferred onto a special tape, physically send the tape to the other site, and then copy (load) the information from that tape onto the system.

A carry operation consists of three main steps:

1. A user at the sending site issues a carry request.
2. Periodically (e.g., daily) at the sending site the carry administrator dumps all user requests.
3. The carry administrator at the target site loads the tape containing the carried information.

The rest of this section describes the steps taken by a user when issuing a carry request and briefly summarizes how the request is processed. Section 2 describes the specific steps taken by the carry administrator when dumping and loading tapes.

THE CARRY QUEUE

Carry queues are message segments into which carry requests are automatically placed. Access to the queues is described by the letters adros:

a	access to add a request
d	access to delete any request
r	access to read any request
o	access to read and delete one's own requests
s	access to determine the total number of requests in the queue

Users require "ao" access to a carry queue in order to add requests and list or cancel their own requests.

ISSUING A CARRY REQUEST

Individual users submit (queue) requests for carrying their own entries using the enter_carry_request command (described in Section 6).

The carry queue is read periodically and the requested entries are dumped on a tape. This tape is sent to the target site, where it is loaded.

The process that dumps and loads carry tapes belongs to one or more carry administrators, users with r and d extended access to the queue. In most cases, there is only one carry administrator, for example Carry.Multics.

Access required to carry an entry is:

1. s to all carry administrators on the parent directory.

2. s to the user and to all carry administrators on all directories in any subtree being carried.

3. r to the user and to all carry administrators on a segment being carried, or on all segments in a subtree being carried.

If the user does not have sa access to the parent directory at the target site, the entry is not loaded and a copy is loaded instead in a directory under >daemon_dir_dir>carry_dir>copies. The requestor receives notification if this happens.

If any directories in the pathname of a target entry do not exist, they are created.

There is a destination associated with each request. Carry queues are named:

```
<destination>.carry.ms
```

and are created by the ms_create (mscr) command (described in the MAM System). The administrator can add the name carry.ms to one of the carry queues with the ms_add_name command (also described in the MAM System), making the queue the default when no destination is specified to the enter_carry_request command.

For example, if a carry administrator regularly circulates tapes between a site in Boston and two other sites (e.g., New York and Phoenix), he must create two queues to contain the requests (e.g., NY.carry.ms. and Phx.carry.ms). If most requests are sent to Phoenix, the administrator can add the name carry.ms to the queue Phx.carry.ms. Then users can send requests to Phoenix by typing "enter_carry_request <paths>", whereas users wishing to send requests to New York must explicitly specify that destination by typing "enter_carry_request -destination NY <paths>".

The default directory in which queues reside is:

```
>daemon_dir_dir>carry_dir
```

This default can be overridden by specifying the -queue_dir (-qd) control argument to the enter_carry_request command.

After the user issues a request, the carry administrator performs the steps summarized below to complete the carry operation.

AT THE SENDING SITE

The carry administrator must have appropriate access to the entries to be carried and to the carry queue. (See Section 2, "Carry Administrator" for a listing of required access.)

The carry administrator makes a carry tape by issuing a command line either interactively or within an absentee process. When a tape is dumped, an exec_com (see Section 3) creates a text segment for each requestor that is also dumped on the tape. The segment contains a listing of that users' requests. When the tape is loaded at the target site, each requestor receives that segment as notification that the administrator has loaded his requests.

If, while dumping the tape, an access or other problem prevents a request from being written on the tape, the carry process prints an error message to the administrator and sends the requestor diagnostic mail. When the administrator dumps a tape, an exec_com creates another text segment named <tape_number>.input, where <tape_number> is the identifier of the tape. This segment contains a listing of all the requests to be dumped on that tape, and can be used later by the carry administrator to dump a replacement tape if the original is destroyed or lost.

AT THE TARGET SITE

The carry administrator loads the carry tape by issuing a command line (see Section 2, "Carry Administrator" for details).

The entry is loaded with the same pathname that the requestor specified at the sending site, either the original pathname or the new one specified by the -new_dir control argument to the enter_carry_request command. Therefore, the request cannot be loaded at the target site if some directory in that pathname does not exist and the carry administrator does not have sufficient access to create it. Other reasons that an entry may not be loaded are lack of quota and lack of (requestor) access.

When an entry cannot be loaded in the place specified originally, a copy of the entry is saved in the carry hierarchy, and the requestor can copy it from there. The pathname of the copy has the directory name >ddd>carry_dir>copies replacing the first two levels of the entry's pathname. For example, if the request:

```
>udd>SysProj>Niles>styles>files
```

cannot be loaded, the retrieved copy is named:

```
>ddd>carry_dir>copies>Niles>styles>files
```

Pathnames with only one or two directory levels (e.g., >exl>o) are appended as is to the pathname of the copies directory (>ddd>carry_dir>copies>exl>o).

SECTION 2

CARRY ADMINISTRATOR

ACCESS REQUIREMENTS

There can be more than one carry administrator (e.g., one per site); carry administrators are any persons having adros access to a carry queue. Since any of the carry administrators can be the one to dump the requested entries, all carry administrators must have appropriate access to the entries, as well as to the queues.

Access required at the sending site is:

- s to all the queue's carry administrators on the parent directory of the entry being carried,
- s to the requestor and to all the queue's carry administrators on all directories in any subtrees being carried, and
- r to the requestor and to all the queue's carry administrators on a segment being carried or on all segments in a subtree being carried.

Access recommended at the target site is:

- sa to the requestor on the parent directory of the entry being carried. If this access is lacking when the entry is to be loaded, a copy is loaded in the copies directory instead.

DUMPING THE TAPE

The administrator makes a carry tape by issuing the command line:

```
carry_dump tape_number {queue_path} {-control_arg}
```

or the active string:

```
[carry_dump tape_number {queue_path} {-control_arg}]
```

either interactively or within an absentee process.

If all the room on one tape is used up, the operator mounts an additional tape to complete the dump. An explanatory message is sent to the carry administrator when this happens, and the tapes are sent together to the target site. (See the carry_dump command and the -next_vol control argument to the carry_load command in Section 5.)

If a request cannot be dumped, the requestor receives diagnostic mail and the carry process prints an error message.

When a carry tape is destroyed or lost, a replacement can be written using the remake_carry_tape command. This command reads the input segment <tape_number>.input created in the queue's parent directory by the carry_dump command at the sending site.

When a tape is dumped, the directory:

```
>carry_dir>mail_to_carry
```

is also dumped on that tape containing a text segment named Person_id.Project_id for each carry requestor. When the tape is loaded at the target site, the segments under mail_to_carry are mailed to the requestors as notification that the requests have been loaded.

LOADING THE TAPE

The administrator loads a carry tape by issuing the command line:

```
carry_load tape_number {-control_args}
```

or the active string:

```
[carry_load tape_number {-control_args}]
```

The pathname of an entry as it appears on the tape is the pathname specified by the requestor to the enter_carry_request command. If some directory in this pathname does not exist at the target site, and the carry administrator's process does not have sufficient access to create it, the request is not loaded in place. Also, an entry is not loaded in place if there is insufficient quota or if the parent directory does not give sa access to both the requestor and the carry administrator. The copies are retrieved in the directory >ddd>carry_dir>copies, but the carry administrator can optionally use a different directory for copies.

The administrator can list the contents of a carry tape by typing:

```
carry_map tape_number
```

and can retrieve specific entries by typing:

```
carry_retrieve tape_number paths {-control_args}
```

The -new_dir control argument to the carry_retrieve command causes the preceding path to be retrieved into a different directory. This is called cross-retrieval and is useful when quota or access restrictions prevent retrieving an entry in the originally specified place.

SECTION 3

SAMPLE CARRY FACILITY

This section describes a sample carry facility using exec_com (see the MPM Commands) and absentee segments to dump and load tapes. The two sites in this example are PHX and MIT. The exec_coms are included in Section 4 of this document. The one called to.ec (also to.absin) runs daily as a self-submitting absentee job. The one called from.ec is executed by the carry administrator whenever a carry tape from the other site arrives at the computer room.

The carry administrator is responsible for circulating the tapes between the sites, checking on the results of the absentee job to.absout at each site, running from.ec at each site, interacting with requestors, and handling any unforeseen problems that may occur while dumping or loading the tapes.

In this example, there is only one carry administrator, named Carry.Multics.

The tapes are numbered 50201 through 50210 and are allocated and freed by the manage_volume_pool (mvp) command, which is described in the MPM Commands. This command/active function frees a tape in the pool after it is loaded and allocates a free tape to dump.

To write a specified tape, the calling sequence is:

```
     ec to <destination> <tape_no>
```

To write any free tape, the calling sequence is:

```
     ec to <destination>
```

The latter prints a message of the form "Using tape 502XX".

Every time to.ec (to.absin) runs under absentee, it resubmits itself for 2:00 pm the following day. It calls the carry_dump command to write on the next free tape.

The carry administrator occasionally has to delay the absentee job to wait for important requests that must be carried that day. This is done by cancelling to.absin ("cancel_absentee_request to"), making sure the requests are in the queue and then running to.ec by hand ("ec to ...") and submitting to.absin to run the following day:

```
     enter_absentee_request to -rt -time 1400. -rsc tape_drive -ag <destination>
```

where the -rt (-restart) control argument causes the job to restart automatically in case of a crash, and the -rsc (-resource) tape_drive argument begins the job only if a tape drive is available. There is also a simple exec_com to postpone the absentee job:

```
     ec delay_to <new_time> <destination>
```

When to.ec has completed a tape, it logs what it has done by calling two exec_coms:

```
ec carry_logger <tape_no> <to_or_from> <destination> <date_time>

ec update_carry_file <tape_no> <to_or_from> <destination>
```

The first exec_com, carry_logger.ec, updates the user-readable segment XXX_tapes.info, where XXX is a destination. This info segment describes the status of all the carry tapes that go back and forth between this site and XXX. Its contents look like this:

```
-----------------------------------
|                                 |
|    07/28/81  1405.6 edt Tue     |
|    Carry tape 50203 to THERE    |
|                                 |
|    07/28/81  1408.1 mst Tue     |
|    Carry tape 50208 from THERE  |
|                                 |
|    07/27/81  1403.1 edt Mon     |
|    Carry tape 50202 to THERE    |
|                                 |
|    07/27/81  1403.7 mst Mon     |
|    Carry tape 50207 from THERE  |
|    .                            |
|    .                            |
|    .                            |
|                                 |
-----------------------------------
```

where THERE is the destination name of the other site, having the value "PHX" at MIT and "MIT" at PHX.

The second exec_com updates the user-readable segment to_THERE.info, which lists the contents of only those tapes dumped at the site where the info segment resides. The older tapes have probably been overwritten at the other site. This segment is a concatenation of tape_log segments written by carry_dump:

```
---------------------------------------------------------------------
|    07/28/81  1405.6 edt Tue:                                      |
|    Carry tape 50203 to THERE written 07/28/81  1405.6 edt Tue     |
|    Subtree >user_dir_dir>Multics>Sherman>to_xxxx  Sherman.Sample.a |
|    Segment >daemon_dir_dir>carry_dir>50203.input  Carry.Multics.a |
|                                                                   |
|    07/27/81  1403.1 edt Mon:                                      |
|    Carry tape 50201 to THERE written 07/27/81  1403.1 edt Mon     |
|    Subtree >no_backup_dir_dir>SProj>Gluck>s>secrets  Gluck.SProj.a |
|    Segment >user_dir_dir>TRLIB>Walton>thesis  Walton.TRLIB.a      |
|    Segment >user_dir_dir>Horrors>Cthulu>mask  Cthulu.Horrors.a    |
|    Segment >daemon_dir_dir>carry_dir>50202.input  Carry.Multics.a |
---------------------------------------------------------------------
```

Then to.ec calls carry_archive.ec to archive the tape_log segment created by carry_dump:

```
ec carry_archive <tape_number> <to_or_from> <destination>
```

After each time to.absin has run, the administrator logs in as Carry and prints the contents of the segment to.absout to see whether the tape was written successfully. A successful run looks roughly like this, with optional lines preceded by a plus sign (+):

```
       Absentee user Carry Multics logged in: 07/28/81 1400.8 est Tue
  +    From Carry.Multics 07/27/81  1406.6 edt Mon:
  +    Tape 50210 to PHX done.  (from previous day)
       Using tape 50201
  +    assign_resource: Waiting for assignment.
       Device tape_04 assigned
       Mounting tape 50201 for writing
  +    := 07/28/81  1401.3 edt Tue:
  +    Starting carry_dump to PHX, tape 50201
       Mounted tape 50201 on drive 4
       carry_dump: Normal termination.

          >ddd>carry_dir>50201.tape_log 07/28/81  1404.7 edt Tue


  +    Search failed.  (from qedx, appending to info seg)
       ID: 180340; 12 already requested.

       Absentee user Carry Multics logged out 07/28/81 1406.0 edt Tue
       CPU usage 47 sec, memory usage 717.0 units
```

If the requested tape is not available, an appropriate error message is printed, for example:

```
carry_dump: Resource not available.  50201
```

If anything goes wrong, the absentee job is automatically resubmitted for the next day. The administrator should check to make sure that the job has been queued.

Depending on the nature of any errors that have occurred, the administrator may want to remake the tape by calling the remake_carry_tape command, or by calling remake_tape.ec:

```
ec remake_tape <old_tape_number> <new_tape_number> <destination>
```

The loading exec_com is called as follows:

```
     ec from <destination> <tape_number>
```

This exec_com calls carry_load and then logs its results in THERE_carry_tapes.info and from_THERE.info. A successful run looks roughly like this:

```
-----------------------------------------------------------------------
|        ec from PHX 50206   <administrator types this line>           |
| +      assign_resource: Waiting for assignment.                      |
|        Device tape_02 assigned                                       |
|        Mounting tape 50206 for reading                               |
|        Mounted tape 50206 on drive 2                                 |
| +      build_tree: Incorrect access to directory.                    |
| +                   >udd>m>Cleary>notes                              |
| +      ACL,ring_brackets,safety_switch: Incorrect access to directory.|
| +                   >udd>FTP>Jensen>ftp_                             |
| +      Segment >udd>FTP>Jensen>ftp_ not loaded in place.             |
|        carry_load: Normal termination.                               |
| +      1 request not loaded in place.                                |
-----------------------------------------------------------------------
```

Note that >udd>m>Cleary>notes has been loaded as a copy:

```
     >ddd>carry_dir>copies>Cleary>notes
```

Any time a copy cannot be loaded for some reason, its pathname is printed at the end of the printed output. For example:

```
----------------------------------------
|    3 requests not loaded in place.    |
|    OF THESE, 2 COPIES NOT LOADED:     |
|      >udd>m>Livingston>expedition     |
|      >exl>Myths>bound_prometheus_     |
----------------------------------------
```

It is possible to quit out of from.ec if the specified tape cannot be found and to execute it again later.

When the daily dumping and loading have been completed successfully, the administrator can list the tapes written so far this week at each site by printing the segment:

```
     >ddd>carry_dir>THERE_tapes.info
```

where THERE is the name of the opposite site.

It is important to remember that before running the sample exec_coms for the first time, a value segment (for use by the value command/active function), a volumes segment (for use by the manage_volume_pool command), and two info segments (for logging results) must be created. To do this, use the following steps.

First, the segment Carry.value_seg must be created in the home directory if such a segment does not already exist:

```
cr Carry.value_seg
```

Second, the command line:

```
mvp add 502(01 02 03 04 05 06 07 08 09 10)
```

creates a volume manager segment named Carry.volumes. Any tape that is not physically at that site, and therefore not available for dumping, must be set to an allocated state:

```
mvp allocate 502XX THERE
```

where THERE is a comment naming the other site. Free tapes can be listed at any time by typing:

```
mvp list -free
```

Third, two info segments, named to_THERE.info and THERE_tapes.info (THERE names the target site) must be created in >ddd>carry_dir, each initially containing two blank lines. Both segments should give rw access to all carry administrators and r access to all users of the facility.

Fourth, a segment named >udd>m>Carry>nl must be created containing a single newline character. This segment is used for editing by the carry_logger exec_com.

To summarize the daily activities at the target site:

1. Receive incoming tape and deliver it to the computer room.

2. (After to.absin has run for the day) log in as carry administrator at each site to (a) check the contents of to.absout to make sure that the tape was written successfully, and (b) issue the command line "lar" (list_absentee_requests) at each site to make sure that to.absin is queued to dump a tape the next day.

3. (After the incoming tape has been taken to the computer room) log in as carry administrator at target site and execute from.ec.

SECTION 4

SAMPLE EXEC_COMS

The examples contained in this section illustrate the use of exec_coms to run the Multics carry facility. These exec_coms dump carry requests, remake a carry tape, load carry requests, log a tape and archive a list of its contents, and print the daily listing of carry tapes.

The carry administrator must make sure that all of the following names are on the exec_com segment:

```
to.ec
to.absin
remake_tape.ec
delay_to.ec
from.ec
carry_logger.ec
update_carry_file.ec
carry_archive.ec
```

```
&command_line off
&input_line off
&goto &ec_name
&
&
&   to.ec: Dump carry requests.
&
&
&label to
&
&  Check calling sequence
&
&if [exists argument &1] &then &goto arg1_there
&print Usage:  ec to destination {tape_number}
&if [not [user absentee]] &then &quit
sm [user name].[user project] No destination for to.absin
sm [user name].[user project] Abs job not run or resubmitted.
&quit
&
&label arg1_there
value$set to_&1_retrying false
&
&  Skip absentee tests for to.ec
&
&if [not [user absentee]] &then &goto ec_to
&
&  Remake a tape whose creation was previously interrupted
&
&if [equal [value to_&1_status] not_completed] &then &goto ec_to
value$set to_&1_status not_completed
&
&  Remake Friday's tape on Monday not Saturday or Sunday
&
&if [equal [day_name] Saturday] &then &goto to_weekend
&if [not [equal [day_name] Sunday]] &then &goto to_nextday
&
&label to_weekend
ear to -rt -time "Monday 1400" -rsc tape_drive -ag &1
&print Attempt to run to.absin on weekend.
&print Resubmitted for Monday at 1400.
&quit
&
&label to_nextday
&if [equal [day_name] Friday]
&then value$set next_to_&1_time "Monday 1400."
&else value$set next_to_&1_time 1400.
&
&  Resubmit absentee
&
ear to -rt -time [date_time [value next_to_&1_time]] -rsc tape_drive -ag &1
&
&  Both to.ec and to.absin come here
&
&label ec_to
value$set to_&1_date [date]
&if [not [exists argument &2]] &then &goto use_default
&
&  Tape specified; see if it's free
&
```

```
&if [mvp test &2] &then &goto use_tape_specified
value$set to_&1_tape [mvp allocate * &1]
string Tape &2 not available, using [value to_&1_tape]
&goto make_tape
&
&label use_tape_specified
value$set to_&1_tape &2
mvp allocate &2 &1
&goto make_tape
&
&label use_default
value$set to_&1_tape [mvp allocate * &1]
string Using tape [value to_&1_tape]
&
&label make_tape
&if [user absentee] &then sm [user name].[user project]
     Starting carry_dump to &1, tape [value to_&1_tape]
&
&  Call carry_dump
&
&if [carry_dump [value to_&1_tape] >ddd>carry_dir>&1.carry.ms
     -force]
&then &goto log_to
&mvp free [value to_&1_tape]
&if [user absentee] &then &goto retry
string Unable to dump tape [value to_&1_tape]
&quit
&
&label retry
mvp allocate [value to_&1_tape] ERR
sm [user name].[user project] Unable to dump
     tape [value to_&1_tape] to &1 at [date_time]
&if [equal [value to_&1_retrying] true] &then &quit
value$set to_&1_retrying true
&goto use_default
&
&label log_to
ec carry_logger [value to_&1_tape] to &1 [date_time]
ec update_carry_file [value to_&1_tape] to &1 [date_time]
ec carry_archive [value to_&1_tape] to &1
&if [not [user absentee]] &then &quit
value$set to_&1_status completed
value$set to_&1_retrying false
sm [user name].[user project] Tape [value to_&1_tape] to &1 done.
&quit
&
&  remake_tape.ec : Remake a carry tape
&
&label remake_tape
&if [exists argument &3] &then &goto remake_test
&print Usage:  ec remake_tape old_number new_number destination
&quit
&
&label remake_test1
&if [mvp test &2] &then &goto remake
&print Tape &2 not available
&quit
&
&label remake
```

```
&if [remake_carry_tape &1 &2] &then &goto finish_remake
&
&print Unable to make tape &2
&quit
&
&label finish_remake
mvp allocate &2 &3
value$set to_&3_date [date]
ec carry_logger &2 to &3 [date_time]
ec update_carry_file &2 to &3 [date_time]
ec carry_archive &2 to &3
&quit
&
&  delay_to.ec : Postpone to.absin to run at a later time
&
&if [exists argument &2] &then &goto delay
&print Usage:  ec delay_to new_time destination {tape_number}
&quit
&
&label delay
car to
ear to -rt -time "&1" -rsc tape_drive  -ag &f2
&quit
&
&  from.ec : Load carry requests
&
&label from
&if [exists argument &2] &then &goto load_tape
&print Usage:  ec from destination tape_number
&quit
&
&label load_tape
&
&if [not [carry_load &f2 -ds MIT]] &then &quit
&
mvp free &2
&
value$set from_&1_date [date]
ec carry_logger &2 from &1 [date_time]
ec update_carry_file &2 from &1 [date_time]
ec carry_archive &2 from &1
&print
&quit
&
&   carry_logger.ec, update_carry_file.ec, and carry_archive.ec :
&
&   Log tape in THERE_tapes.info,
&   log its contents in from_THERE.info
&   or to_THERE.info and archive its tape_log segment in
&   from_THERE.archive or to_THERE.archive
&
&
&label carry_logger
&attach
&if [not [exists segment >ddd>carry_dir>&3_tapes.info]]
&then create >ddd>carry_dir>&3_tapes.info
&if [not [exists segment >ddd>carry_dir>&2_&3.info]]
&then create >ddd>carry_dir>&2_&3.info
qx
r >ddd>carry_dir>&3_tapes.info
0i
&4:
Carry tape &1 &2 &3
\f
.r >udd>m>Carry>nl
31,$d
w >ddd>carry_dir>&3_tapes.info
q
```

```
&quit
&
&
&label update_carry_file
&attach
abc >ddd>carry_dir>&1.tape_log -ch
qx
r >ddd>carry_dir>&2_&3.info
/Carry tape &1/,$d
0r >ddd>carry_dir>&1.tape_log
.r >udd>m>Carry>nl
.r >udd>m>Carry>nl
1i
&4:
/f
w >ddd>carry_dir>&2_&3.info
q
&quit
&
&
&label carry_archive
an >ddd>carry_dir>&1.tape_log [value &2_&3_date].&1
archive a &2_&3 >ddd>carry_dir>[value &2_&3_date].&1
dn >ddd>carry_dir>[value &2_&3_date].&1
&quit
```

SECTION 5

CARRY ADMINISTRATOR COMMANDS

This section describes commands whose use is restricted to the Carry Administrator. The commands are presented in alphabetical order and follow the syntax used throughout Multics manuals. (For details about this syntax, see the MPM Commands.)

Name: carry_dump

This command dumps the carry requests in a specified queue onto a specified tape.

Usage

```
     carry_dump  tape_number {queue_path}  {-control_arg}
```

where:

1. tape_number
   is the identifier of the tape to be written.

2. queue_path
   is the pathname of a carry queue. The default is >daemon_dir_dir>carry_dir>carry.ms.

3. control_arg
   can be -force or -fc to write a tape even if there are no requests in the queue. If there are no requests, the tape contains only a segment named <tape_number>.input, consisting of a header line followed by the line "No requests submitted." If this control argument is not specified, a tape is only written if one or more requests exist.

Notes

The carry administrator must have r and d extended access to the queue.

Mail is sent to those carry requestors whose entries cannot be dumped.

The following entries are created in the queue's parent directory and are dumped on the tape:

1. An input segment named <tape_number>.input listing all the carry requests.
2. A tape log segment named <tape_number>.tape_log listing all the requests that were actually written onto the tape.
3. A directory named mail_to_carry containing a segment named Person_id.Project_id for each carry requestor. These segments are mailed at the target site by carry_load to inform users that their requests have been loaded.

If the space on the tape runs out and there is more information to be dumped, carry_dump notifies the operator to mount another tape. The operator mounts another tape and the dumping continues on the new tape. (This requires no action by the carry administrator.)

Example

If an additional tape is mounted to continue dumping of one carry_dump request, appropriate notification appears on the carry administrator's terminal:

```
                .
                .
                .

     Mounting tape (another) for writing
     Mounted tape (another) on drive 4
     carry_dump:  Normal termination.

                .
                .
                .
```

The second tape is requested by the string "(another)" so that the choice of a tape is left up to the operator.

Name: carry_load

This command loads a carry tape at its target site.

Usage

```
     carry_load tape_number {-control_args}
```

where:

1. tape_number
   is the identifier of the tape to be read.

2. control_args
   can be chosen from the following:

   -comment STR, -com STR
      prints the message STR at the operator's console. This comment can be used following a tape_number argument or the -next_vol control argument, when an identifier for the same tape varies between sites (see "Notes" and "Examples" below).

   -copy_dir path, -cpd path
      loads copies of entries under the pathname "path" when they cannot be loaded in place because of access or quota restrictions. The copies are loaded under "path" rather than under the default directory >ddd>carry_dir>copies (see "Notes" below).

   -force, -fc
      loads the tape even if it is more than five days old. By default, the tape is not loaded if it is older than five days, and an error message is printed.

   -next_vol tape_number, -nxv tape_number
      sequentially loads more than one tape (when the dumping has run onto additional tapes). Multiple occurrences of this control argument are allowed on a line.

   -queue_dir path, -qd path
      specifies the pathname of the carry queue's parent directory at the sending site, if different from >daemon_dir_dir>carry_dir.

Notes

When a request cannot be loaded because of insufficient access or quota, a copy is loaded in the directory >daemon_dir_dir>carry_dir>copies. This directory name replaces the first two levels of the entry's pathname. For example, if the entry:

```
     >udd>Demo>JRSmith>tx.archive
```

cannot be reloaded, the retrieved copy is named:

```
     >ddd>carry_dir>copies>JRSmith>tx.archive
```

If the original pathname has only one or two directory levels (e.g., >exl>o), the entire pathname is appended to the pathname of the default directory (e.g., >ddd>carry_dir>copies>exl>o).

A tape may be labelled with two different identifying numbers when each site has its own method of identification. The -comment control argument can be used by the carry administrator to specify a tape identifier at the target site when it is different from the identifier of the same tape at the sending site (see "Examples" below).

Examples

To load the three-volume carry tape consisting of 50204 continued on 50011 and then on 50012 (all dumped by a single invocation of the carry_dump command), use the -next_vol (-nxv) control argument:

```
carry_load 50204 -nxv 50011 -nxv 50012
```

The above command line loads sequentially the three designated tapes in the order shown.

To load the two-volume carry consisting of 50207 continued on 50201, where 50207 is registered at the target site as TX653 and 50201 as TX647, use the -comment control argument along with the -next_vol control argument:

```
carry_load 50207 -com TX653 -nxv 50201 -com TX647
```

Name: carry_map

This command reads a carry tape and lists its contents.

Usage

```
carry_map tape_number {-control_arg}
```

where:

1. tape_number
   is the identifier of the tape.

2. control_arg
   can be of the form:

   -next_vol tape_number, -nxv tape_number
   sequentially loads more than one tape (when the dumping has run onto additional tapes). Multiple occurrences of this control argument are allowed on a line.

Note

The table of contents is contained in the text segment:

```
>daemon_dir_dir>carry_dir>TAPE_NUMBER.archive
```

which is retrieved from the tape and printed. An example is:

```
Carry tape 50207 to PHX written 07/13/81  1401.1 edt Tue
Segment >daemon_dir_dir>carry_dir>50207.input  Carry.Multics
Subtree >user_dir_dir>DEV>Franks>carry_dir  Franks.DEV
Segment >user_dir_dir>Multics>Smedley>MTB662  Smedley.Multics -move
           Move to >udd>m>Smedley>bk1
Segment >user_dir_dir>MC>Aiken>formulas.pln  Aiken.MC
```

Name: carry_retrieve

This command loads specified entries from a carry tape and optionally moves them to different pathnames.

Usage

```
carry_retrieve tape_number paths {-control_args}
```

where:

1. tape_number
   is the identifier of the tape to be read.

2. paths
   are the pathnames of storage system entries to be loaded.

3. control_args
   can be chosen from the following:

   -new_dir dir_path, -nd dir_path
   loads the entry of the preceding path argument under the directory specified by dir_path instead of under its original parent (at the target site).

   -next_vol tape_number, -nxv tape_volume
   sequentially loads more than one tape (when the dumping has run onto additional tapes). Multiple occurrences of this control argument are allowed on a line.

   -select
   prints a line-numbered table of contents for the tape, then asks a single query of the form "Request numbers:", and accepts line numbers of requests to be retrieved (see "Examples" below).

Notes

The requestor requires sa access to the (target) parent directory of each entry to be retrieved. If the requestor lacks m access to the parent, an error message states that the entry's ACL, ring brackets, and safety switch cannot be loaded.

Examples

The command line:

```
carry_retrieve 50206 one two -new_dir nd three -new_dir nd four
```

retrieves the entries named one and four in the working directory and retrieves the entries named two and three in the directory named nd, under the working directory. It is assumed that all the entries were in the current working directory when they were dumped.

The command line:

```
carry_retrieve 50206 -select
```

reads tape 50206, prints a list of the requests contained on the tape, and then prints a query of the form "Request numbers:". The carry administrator answers this query with a single line of input, consisting of the line numbers as shown on the list just printed. The line numbers must be separated by spaces and can optionally be interspersed with "-new_dir path" control arguments to cross-retrieve entries to different directories. For example:

```
carry_retrieve 50206 -select <administrator requests retrieval>
Mounting tape 50206 for reading
Mounted tape 050206 on drive tape_02

Carry tape 50206 to MIT written 02/03/81  1302.5 mst Tue

  1  Segment >daemon_dir_dir>carry_dir>50206.input  Carry.Multics
  2  Subtree >udd>SysProj>New>LIB107-B  Smith.SysProj
  3  Subtree >udd>SysProj>New>comp_dir  Jones.SysProj
  4  Segment >udd>m>Disturbed>reginald  Frank.Multics
  5  Segment >udd>Hometeam>Manager>scores  Ryan.Hometeam

Request numbers:   3 4 -nd >udd>Flames>Old 5
```

which retrieves lines 3 and 5, and cross retrieves line 4 into the directory >udd>Flames>Old.

Name: carry_total

This active function returns the number of queued carry requests.

Usage

```
   [carry_total {-control_args}]
```

where control_args can be chosen from the following:

-admin
: returns the total number of requests. The default is the number of the user's own requests.

-destination STR, -ds STR
: specifies the queue STR.carry.ms instead of the default queue carry.ms.

-queue_dir path, -qd path
: specifies the pathname of a directory in which carry queues reside. By default, this directory is >daemon_dir_dir>carry_dir.

Note

The carry administrator must have status permission on the queue in order to use the -admin control argument.

Name: remake_carry_tape

This command rewrites a carry tape using the input segment <tape_number>.input from a previous carry_dump.

Usage

```
remake_carry_tape tape_number {new_number} {-control_arg}
```

where:

1. tape_number
   is the identifier of a previously written tape for which the input segment <tape_number>.input exists in the directory >ddd>carry_dir.

2. new_number
   is the identifier of the tape to be written, if different from tape_number.

3. control_arg
   can be of the form:

   -queue_dir path, -qd path
   specifies the directory containing the input segment. By default, this directory is >daemon_dir_dir>carry_dir.

SECTION 6

USER COMMANDS AND ACTIVE FUNCTIONS

This section describes the commands and active functions pertaining to the Multics carry facility that are available to general users of the facility as well as to administrators. Commands whose use is restricted to the Carry Administrator can be found in Section 4. The descriptions are presented in alphabetical order and follow the syntax used throughout Multics manuals. (For details about this syntax, see the MPM Commands.)

Name: cancel_carry_request, ccr

This command cancels requests queued by the enter_carry_request command.

Usage

```
     ccr {paths} {-control_args}
```

where:

1. paths
   are the pathnames of segments and directories. The star convention is allowed.

2. control_args
   can be chosen from the following:

   -admin, -am
   allows any user's requests to be cancelled. This control argument requires r and d extended access to the queue. By default, only the user's own requests can be cancelled.

   -destination STR, -ds STR
   specifies a destination site, where STR is up to 23 characters long. The carry queue searched is named STR.carry.ms. If no destination is specified, the queue is named carry.ms, the name added to the queue for the default destination.

   -entry STR, -et STR
   specifies the entryname (STR) of the request to be cancelled, instead of its entire pathname. (See "Notes" below.) The star convention is allowed.

   -queue_dir path, -qd path
   specifies the queue's parent directory. The default is >daemon_dir_dir>carry_dir.

Notes

At least one path or -entry argument must be specified.

If the -entry control argument is used and the STR specified is not a starname, there must be only one request with that entryname in the queue (e.g., you should not have a request with the same entryname in a different directory).

See also the enter_carry_request and list_carry_requests commands.

Example

To cancel a carry request for the segment >udd>m>Charles>work>system.ec, type:

```
ccr -entry system.ec
```

To cancel all of your own carry requests, type:

```
ccr -entry **
```

Name: enter_carry_request, ecr

This command queues a segment or directory subtree to be carried to another site.

Usage

```
     ecr paths {-control_args}
```

where:

1. paths
   are pathnames of segments or directories. In the case of a directory, the entire subtree is carried. The star convention is allowed. Starnames queue separate requests for each matching segment, multisegment file, and directory.

2. control_args
   can be chosen from the following:

   -destination STR, -ds STR
      specifies a destination site, where STR is up to 23 characters long. The carry queue used is named STR.carry.ms. If no destination is specified, the request is placed in carry.ms, the name added to the queue for the default destination.

   -new_dir dir_path, -nd dir_path
      loads the entry of the preceding path argument under the directory specified by dir_path instead of under its original parent (at the target site).

   -notify, -nt
      sends notification to the requestor when the request is dumped.

   -no_notify, -nnt
      suppresses sending of notification when the request is dumped. This is the default.

   -no_trim
      suppresses the deletion (when requests are loaded at the target site) of target subtree entries that do not appear in the corresponding subtrees at the sending site. This is the default.

   -queue_dir path, -qd path
      specifies the queue's parent directory. The default is >daemon_dir_dir>carry_dir.

   -trim
      deletes entries in subtrees at the target site that do not exist in the carried subtrees. The default is -no_trim.

-user User_id
: specifies the owner of the carried entries at the target site, if different from the requestor. User_id is of the form Person_id.Project_id. This control argument sets access to the carried entries at the target site for User_id rather than for the requestor, and sends notification to User_id rather than to the requestor. This control argument is needed if the requestor is registered on a different project or with a different name at each of the two sites.

Notes

See also the list_carry_requests and cancel_carry_request commands. *

Name: list_carry_requests, lcr

This command lists requests queued by the enter_carry_request command.

Usage

```
lcr {-control_args}
```

where control_args can be chosen from the following:

-admin, -am
: lists all the requests. This control argument requires r extended access to the queue. By default, only the user's own requests are listed.

-all, -a
: lists requests in all carry queues (in the default directory queue or the one specified by the -queue_dir control argument) to which the user has r or o extended access.

-destination STR, -ds STR
: lists only requests specified for destination site STR, where STR is up to 23 characters long. The carry queue listed is named STR.carry.ms. If no destination is specified, the queue is named carry.ms, the name added to the queue for the default destination.

-queue_dir path, -qd path
: lists only requests specified by the queue's parent directory. The default is >daemon_dir_dir>carry_dir.

Note

See also the enter_carry_request and cancel_carry_request commands.

INDEX

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	LEVEL 68 MULTICS CARRY FACILITY
ORDER NO.	AN76-02
DATED	FEBRUARY 1981

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. ☐

FROM: NAME ______________________ DATE ____________

TITLE ______________________

COMPANY ______________________

ADDRESS ______________________

____

CUT ALONG LINE

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

FOLD ALONG LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

FOLD ALONG LINE

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In the U.K.: Great West Road, Brentford, Middlesex TW8 9DH
In Australia: 124 Walker Street, North Sydney, N.S.W. 2060
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

30649, 5C281, Printed in U.S.A.

AN76-02