To: Multics Staff

From: Dennis Capps

Date: 05/14/74

Subject: The Tape Mount Package: Initial Version

This package for controlled mounting of magnetic tapes was first proposed in MSB-69. This document describes the implementation which is proposed for installation. Some features of the original proposal have not been implemented in this first version; these possible enhancements will be discussed in a separate memorandum.

The design of this Tape Mount Package is motivated primarily by security needs:

1) The system must verify that a user has access to any tape he/she tries to mount.
2) The system must verify that every tape mounted is the one requested.
3) The system must verify that the write-permit hardware is engaged when and only when a tape is mounted for writing.

We can force every tape mount to be executed by ring1 programs which will perform the above verifications by making certain entries in the hardcore tape device control module(tdcm) available only from ring1 (through admin_gate_), by making tdcm record in tape_data the validation level at the time the drive is attached and by making tdcm not honor calls from rings of higher number until the drive is explicitly "promoted".

A consequence is that, from the point of view of a user's Device Interface Module (DIM), mounting a tape is less complicated. In the course of accomplishing the above tasks, the Tape Mount Package performs some actions which are common to all mounts, including setting the density at which the drive is to operate and notifying the operator that a tape is to be mounted. Also, the wakeup which indicates that the tape requested has been mounted goes only to the process for which the tape was mounted, not, as was the case previously, to every process waiting for a drive.

Tape Registration

Volume Description Segments

In order to mount a tape properly and accurately verify that it is the correct one, the tape mount package must have at mount time certain information about the tape, such as recording density, number of tracks of information, logical format(e.g. multics, ansi), etc. This must be information that the system can trust; that is, it must be information that can be modified only by the system, although the system may make modifications in a controlled way at the user's request. This is accomplished by requiring that each tape be registered and by placing the registration information in ring1 segments, one per tape. These Volume Description Segments(VDS) (1) are managed by ring1 programs accessible through the gates trm_ and tape_admin_. Privileged and user commands are provided.

The Volume Description Segments reside in a hierarchy of system directories below >system_control_1>tape. The directories in this hierarchy are named in such a way that the pathname for a given VDS can be calculated from the volume name. The volume name is exactly six characters in length. The calculation is as follows:

```
declare
vdspn      char(168),          /* Pathname of VDS */
volid      char(6);            /* Tape name */

vdspn      = ">system_control_1>tape>" ||
               substr(volid,1,2) || ">" ||
               substr(volid,3,2) || ">" ||
               volid || ".vds";
```

Thus, for example, the VDS for tape 070090 resides at: >system_control_1>tape>07>00>070090.vds

The first design of the VDS envisioned default values for variables such that the bulk of VDS's would be of zero length. This no longer is feasible and a special feature is included to reduce the number of segments of non-zero length needed to register all the tapes in the system. The bitcount in the branch of a VDS is used to indicate whether to use the VDS itself or whether to use an archetypical VDS which serves to record the attributes of several identical VDS's. The value in the bitcount is zero if the VDS itself is to be used, an index in a table of archetypes if an archetype is to be used. The uniformity of backup tapes provides a prime example of the usefulness of such an archetype.

---

(1) Previously called TRDS and BVDS.

The use of an archetype VDS is transparent to the user. When a tape is registered it is automatically compared against entries in a table of archetypes and one is used if possible. All references to the individual VDS are made through a VDS manager which automatically locates the archetype. When a tape registration is modified the archetype is copied, the VDS changed, and a different archetype used if possible.

Registering a Tape

Only a system administrator(usually the tape librarian) can register or unregister a tape. The privileged command tape_register calls into ring1 to create a VDS, to write the registration information therein, and to set the initial tape access on the VDS. The privileged command tape_unregister expunges a registration by the straightforward method of deleting the VDS.

A user who has access to a tape can examine its registration using the command tape_registration_print or (through tssm_) the ring1 entries provided. The command tape_registration_change allows the owner(see Access Control below) of a tape to change much of its registration, under supervision. There is, however, certain administrative and privileged information in the VDS which the user is not allowed to modify.

Descriptions of these commands and examples of their uses may be found in the Appendix.

Mounting Unregistered Tapes

Provision has been made for mounting an unregistered tape(one with no VDS). When mounting such a tape the Tape Mount Package allows the user to supply information which ordinarily would come from the VDS. In doing so, however, the Tape Mount Package relies on the human operator to locate the correct tape. This is clearly a weak point in the system, and an installation can disable this feature by editing an include file and recompiling one program. (1)

A user indicates that a request is for an unregistered tape and supplies the necessary description by means of a special syntax. Rather than specifying just a volume identifier (tape name or slot number), the user provides a volume description, a character string which contains the volume identifier as its first element followed by qualifiers, separated by commas, which describe the tape. This same syntax is used by the registration commands and the qualifiers are listed in the writeup in the

---

(1) There is obvious room for improvement here.

Appendix. The description defaults when mounting are the same as when registering a tape. For a registered tape, of course, only the volume identifier is necessary since the description is taken from the VDS.

Information Protection

Two levels of information protection must be supplied for tape data: logical protection which defines which users may use the data and how the data may be used; and physical protection, the need for which arises from the detachable nature of tapes and which insures that the physical reel being used is indeed the one to which the logical protection applies.

Access Control

Logical protection for a tape and its Volume Description Segment is represented by bits in the extended access control list for the VDS. Access is manipulated by ring1 programs, also available through trm_ and tape_admin_, and user commands are provided which are similar to other access control commands. The extended access is checked before any tape is mounted and before any modification to the VDS or to its ACL to ensure that the requested action is consistent with the user's access.

The initial access on a tape registered to User.Project:

Reg ACL	Ex ACL	Ring Br	Acc Name
rw	rwaco	1,1,1	User.Project.*
rw	null	1,1,1	*.SysDaemon.*
null	null	1,1,1	*.*.*

The meanings of the tape access modes:

read(r)
: User may read the contents of the volume.

write(w)
: User may write anywhere on the volume.

append(a)
: User may write on the tape beyond the end of any data which is on the tape at mount time, but may not erase or overwrite such previous data. (This mode of operation is not supported in any other way.)

control(c)
: User may give to or remove from another user read, write, or append access if User has it, but may not give or remove control access.

owner(o)
: User may give or remove any access except owner. User may modify the registration of the tape. There may be only one owner per tape and this access is set by the privileged tape_register command at the time the tape is registered. (This is an accounting rather than a technical restriction; accounting procedures charge tape rental to the owner. If distribution of charges among users or simply charging the first owner found can be handled, then multiple owners can be allowed.)

Verification of Mounted Tape

The identity of a tape may be verified at mount time in one or both of two ways. When the tape is mounted the operator types a command to the Initializer to indicate that the tape is ready. One argument to this command is a three-character authentication code found on a sticker on the reel. This code is an obscure function of the volume id and may be calculated by the Tape Mount Package and matched to what the operator types.

The other way to verify a tape is to read its label record(s) and examine the information contained therein. One means or the other must be used; both may be used for greater safety.

Reading, Writing, and Examining Volume Labels

When a standard format(Multics, ANSI) tape is mounted for reading, its label is read and examined. When mounted for writing, the label is similarly examined unless it is a new tape(registered as not yet labeled). When mounted for writing, a volume label is not written unless the tape is a Multics standard tape or is not yet labeled.

The actual reading and verification or writing of the label is performed by an entry (DIM_labeler_ in Diagram 2)in the DIM's Device Strategy Module(DSM). This must be code that can run in ring1 and which cannot be replaced by the user, but need not be restricted to running in ring1. Because the Tape Mount Package cannot verify who called it, the labeler is chosen according to the information in the VDS. Thus, a tape must be registered either as conforming to an established and supported standard, or as having no label. In the latter case, no label verification can be performed and authentication is required.

A labeler must have an entry for reading and verifying labels and an entry for writing labels. The read entry is responsible for reading the label record(s) and for performing whatever computation is necessary to ensure that it is a valid label of the standard format. The labeler also extracts the volume name from the label and returns it to the caller.

The write entry is responsible for writing a correct label containing the volume name supplied by the caller.

Both entries may have other responsibilities to the DSM itself and fulfilling these responsibilities may require communication with the DSM. This communication is accomplished by use of the dimdatap argument(see Appendix). This is a pointer to arbitrary data as agreed upon by the labeler and the DSM. The Mount Package does not presume to know anything about this data or its format; the value of dimdatap with which the Mount

Package is called is passed to the labeler, the value returned from the labeler is returned to the DSM.

Organization of Mounting Tasks

Mounting magnetic tapes is done in five steps, two of which are performed by procedures operating in the administrative ring of the user's process, two of which are performed by a tape system control function operating in a system process, and one of which is performed by the human operator.

1) Initiate Request: verify access, set up databases, get drive (user process)

2) Notify Operator: write mount message (system process)

3) Mount: put reel on drive, push appropriate buttons (human operator)

4) Process Operator Reply: authenticate, start accounting (system process)

5) Final Verification: set density, check write-protect, read/write label, promote drive (user process)

The cooperation between the user's process and the tape system control function is accomplished using interprocess communciation and a common data base, the Tape Communication Segment. The tape scf informs the operator by typing messages; the operator is provided with a command for responding to the tape scf.

Tape Communication Segment

The Tape Communication Segment(TCS) consists of a header containing global information followed by an array of tape status blocks(tsb) for individual mount requests. The header contains a lock; the process id of the tape system control function(Initializer process) and the id of the interprocess event channel which invokes the tape system control function; and information to keep track of the array of individual mounts such as the number of requests pending, the number of tapes mounted, the number of elements in the array, and the head of the free storage list.

The tape status block is assigned and filled in by the user-process mounting procedures. It contains an integer code indicating the state of the mount request (requested, pending, mounted, promoted), the user's process id and the event channel over which the user is to be notified when the tape is mounted, a description of the tape and how to handle it, and information to aid in telling the operator what to do.

The description of the tape is derived from that in the VDS and/or from any information which the user may be allowed to specify when making the mount request. In this version the description is taken entirely from the VDS unless an unregistered tape is being mounted. In the latter case the description is taken from the qualifiers supplied by the user at mount time(see writeup of trm_$mount in Appendix) with the same defaults for unspecified attributes as apply when registering a tape. Once filled in, the tsb serves as the sole internal description for the duration of the mount.

The index of the tsb entry within the TCS identifies the request in communication with the operator and with the user DSM. This index is returned to the DSM by trm_$mount and must be supplied on subsequent calls to trm_$mount_check and trm_$dismount. Similarly, the index is included in the mount message written on the tape operations console and must be included in the operator's typed response.

User Process Portion of the Tape Mount Package

The attach entry of a tape DSM calls tmm_$mount to initiate a mount request where previously it would have called hcs_$tdcm_attach. This entry acquires a drive, sets up communications with the tape system control function, notifies the tape scf that a tape is to be mounted, and returns to its caller. The DSM then blocks or otherwise waits for an interprocess wakeup from the tape scf. The tape system control function informs the operator of the mount request and when the tape has been mounted, a wakeup is sent to the user process.

When informed that the tape is ready, the DSM must call tmm_$mount_check to verify that the correct tape has been mounted. This entry will also check for a write ring and set density.

The call to tmm_$mount_check cannot be omitted. The drive is attached in ring1 and continues to have a validation level of 1 until ring1(or ring0) "promotes" it. tdcm will not honor any call from a ring of number higher than the validation level of the drive.

The event message associated with the wakeup that the DIM receives from the tape system control function contains the character-string status key that the operator types as part of the "tape" command. This response may be positive("ok", "redun") or negative("notape", "nodrive", "shutdown"). These negative responses may be used to give the user some idea of what went wrong. In one sense, the information in the event message is redundant and may be ignored; in the negative cases, if tmm_$mount_check is called anyway, it will discover from the state indicator in the tsb that it cannot go on and will return a non-zero error code. However, this error code will often be less precise about what happened than the event message.

Diagram 1 presents a comparison of code a DIM needs to mount a tape without and with the Tape Mount Package. MPM-like writeups of the tmm_ entries are appended to the end of this memorandum.

The DSM does not need to provide the Tape Mount Package with a tseg. However, a tseg(or any other data) may be passed to the DIM's labeler module by means of the dimdatap argument.

The flags returned by tmm_$mount indicate what the Mount Package believes about the tape. This flag word is described by tape_registration_flags (see Appendix) and is a copy of the flag word in the tsb. The information of greatest interest to a DIM is that concerning the logical format of the tape and the status of the label.

Mounting a Tape Without and With the Mount Package

```
Set up SDB and tseg
Create an event channel
Parse ",7track" from volume name, if present,
     and set tseg.drive_number accordingly
Attach a drive (hcs_$tdcm_attach)
Rewind and unload tape
Tell tdcm to signal all special interrupts
Set tseg.write_sw
Write reassuring message to user
Write mount message to operator
Block
Set density
Check for drive in ready state
Check for write ring
Read/Write label
```

Diagram 1.A: Old Style Mount Code in DIM

```
Set up SDB and tseg
Create an event channel
Check whether Mount Package available
     (hcs_$tdcm_mount_bit_get)
Call trm_$mount
Write reassuring message to user
Block
Examine the event message
Call trm_$mount_check
```

Diagram 1.B: New Style Mount Code in DIM

The Tape System Control Function

The tape system control function provides smooth communication with the human operator and with the accounting programs. It also provides for dismounting tapes left mounted during a crash or a process termination.

The various parts of the tape system control function, like the other system control functions in the Initializer process, are invoked directly or indirectly as event call procedures. One entry is invoked directly upon receipt of a wakeup from a user process. The message-coordinator/system_control_ mechanism calls another in response to the "tape" operator command. Other entries are called by accounting and Answering Service programs.

Initialization of the Tape System Control Function

When the system is brought up, one part of initializing the Answering Service is to set up the tape system control function. The responsibility for this setup is shared by tape_opr_ and tape_master_.

The ring4 entry tape_opr_$init creates an event call interprocess communication channel which will invoke tape_opr_$tape_opr_ when it receives a wakeup.

Next tape_opr_$init calls into ring1 (through tape_admin_) to tape_master_$init which initiates the TCS(or creates one if necessary) and partially initializes it.

Upon return from tape_master_$init, tape_opr_$init looks in the TCS to see whether any tapes were left mounted at the time of the last shutdown or crash. If so, it explicitly dismounts them.

Finally, tape_opr_$init calls into ring1 again (to tape_master_$init1) to complete the initialization. When the TCS is completely initialized, tape_master_$init1 calls into the hardcore to inform tdcm that the Tape Mount Package is ready for business.

Informing Users of Mount Package Availability

To allow a DIM to abort gracefully if called at a time when tapes cannot be mounted, and to allow tape_ to distinguish a cold boot situation in which backup tapes must be mounted without the benefit of the Tape Mount Package, the entry hcs_$mount_bit_get is provided. This entry takes a bit(1) argument which is returned with value "0"b if the Tape Mount Package is not available, "1"b if it is.

Operator's Interaction with the Tape Mount Package

The tape system control function writes mount messages on an I/O stream attached through mrd_ with source name "tape". The typical message is produced by a call of the form:

```
call ioa_("^5d (^a ^6a) drive ^2d ^a.^a ^a -- ^a ^a ^a",
          tcsx, reg_chars,
          tsb.tape_reel_id,
          tsb.drive_id,
          tsb.uname, tsb.uproj,
          tsb.ourcomment,
          tsb.act_chars,
          ring_chars, auth_mode);
```

The typical message looks like this when typed:

```
1013 tape   4 ( 070090) drive  3 User.Project -- MOUNT noring auth
```

The variables in the above call to ioa_ are:

1) tcsx — is the index of the tsb for the mount request.

2) reg_chars — is "" or "UNREGISTERED".

3) tsb.tape_reel_id — is the volume name.

4) tsb.drive_id — is the number of the drive on which the volume should be or is mounted.

5) tsb.uname — is the user's login name.

6) tsb.uproj — is the user's project.

7) tsb.outcomment — is normally "". In unusual circumstances it gives information to the operator such as "is not ready", "not at ld pt".

8) tsb.act_chars — tells what to do; e.g. "MOUNT", "DISMOUNT", "REMOUNT", "REREADY", "RING NEEDED", "REMOVE RING".

9) ring_chars — is "ring" or "noring" indicating whether a write ring is wanted.

10) auth_mode — is "auth" or "noauth" indicating whether authentication is required.

Whenever any mount requests are pending a timer is set to repeat every five minutes all pending requests more than five minutes old. A repeated message has the string "*** repeat ***" before tcsx. This timer is reset when all requests have been

satisfied.

The tape system control function does not try to read a reply from the operator and therefore no sentinel appears on the tape console and the operator does not respond with the "reply" command. Rather, a new "tape" command is provided which invokes an entry in the tape system control function.

The usage for this command is:

tape tcsx keyword authentication

1) tcsx — is the tsb index for the mount request.

2) keyword — is one of:

"ok"	means the tape has been mounted.
"redun"	means the tape was already mounted.
"notape"	means the tape could not be found.
"nodrive"	means the specified drive could not be used.
"shutdown"	means the tape cannot be mounted because of imminent system shutdown.
"repeat"	asks the tape system control function to repeat the request tcsx.
"kill"	causes the mount request to be aborted and the drive unloaded and released.
"list"	(only if tcsx is omitted) produces a list of all pending requests and all mounted tapes.

3) authentication — is the three character authentication code found on a sticker on the tape reel. Authentication is meaningful only with the "ok" and "redun" keywords, and is optional for requests marked "noauth".

A tcsx which refers to a request which is not pending or an unrecognized keyword will result in an error message on the tape operations console. If the typed authentication does not match the calculated value, the operator is asked to try again.

History of a Tape Mount Request

Diagrams 2, 3, and 4 illustrate "spatially" and temporally the path of a successful mount request. Supporting registration procedures are not shown.

Diagram 2 shows how various elements involved in a mount request are related and the cooperation necessary among two processes and three rings.

Diagram 3 shows the relationships in time of the events in a successful mount request. This sequence can be broken conveniently into four sections corresponding to the four main entries in the Tape Mount Package.

A) The mount entry in the user-process mounting routines is responsible for setting up the request, acquiring a drive, and notifying the tape system control function.
B) The entry tape_opr_$tape_opr_ in the tape system control function writes a mount message to the operator. (This entry also handles dismount requests.)
C) The entry tape_opr_$reply deals with the operator's response.
D) The entry mount_check in the user-process mounting routines verifies the tape and if satisfied promotes the drive.

Diagram 4 outlines the sequence of events in a tape dismount request. This is similar to the mounting sequence, but no response is expected from the operator and the tape system control function sends no wakeup to the user. The tape system control function, rather than the user-process routines, releases the tape drive.

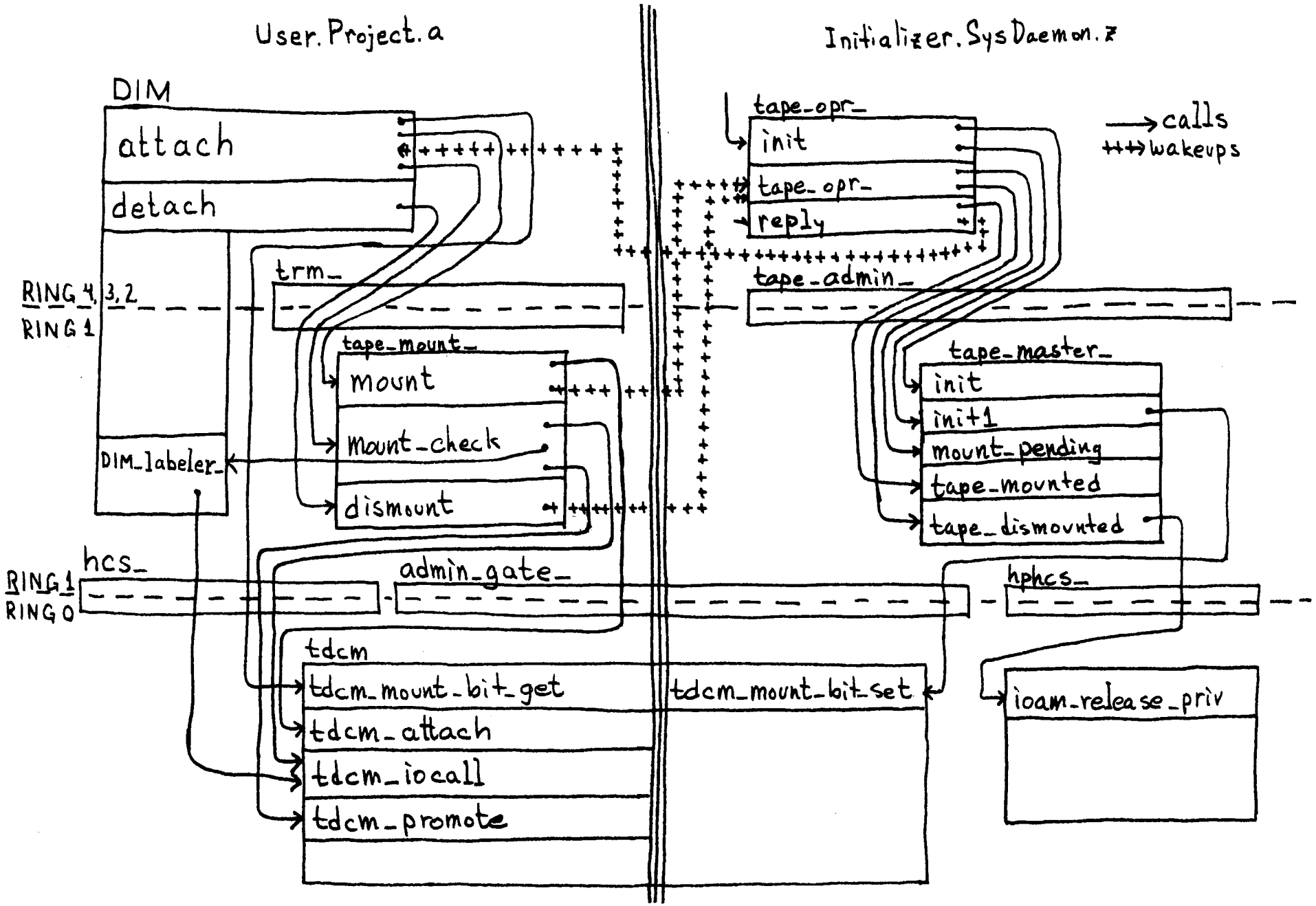

Diagram 2: Modular Relationships Among Programs Involved in a Tape Mount

User.Project r4	r1	Initializer.SysDaemon r4
		call ios_$read ==> block
DIM		
set up SDB, etc.		Random logins & logouts,accounting updates,operation message coordination, etc., all event call driven
call hcs_ $tdcm_mount_bit_get		
call trm_$mount	tape_mount_$mount	
	set validation level = 1	
	initiate TCS	
	parse description	Random operator commands
	initiate VDS	
	set up tsb entry in TCS (entry) (tsb.state = 1)	
	make tseg	
	check access	
	attach drive	
	set action indicators in tsb & in event message	
	send wakeup to tape scf	tape_opr_ (event call)
	set return args drivenumber writesw tcsx flags	
	reset validation level	
	return;	
block		

Diagram 3.A: Sequence of Events in a Mount Request -- mount

Setting Up a Mount Request

The first task that the user-process mounting procedures need to perform is to initiate the TCS. This is done only once per process. Next it needs to initiate the VDS for the volume to be mounted. To this end the description string is parsed. As described above, if the qualifiers indicate that the volume is unregistered, that fact is recorded and no VDS is sought. Any error in initiating these data bases or in calculating the VDS pathname causes the mount to be aborted. Unrecognized qualifiers are ignored.

Allocation of the tsb is straightforward. If there are no blocks on the free storage list, the number of blocks in the table is incremented and the last one used. During this allocation the TCS is locked since the tape system control function might simultaneously want to free a tsb.

The attachment mode("r" or "w") is compared against the access string obtained when finding the VDS and if the user has appropriate access the ring switch in the tsb is set accordingly. There is no access information for an unregistered tape so the user is assumed to have full access to it. This is another reason to discourage use of unregistered tapes.

The Mount Package acquires a drive in the same way that a DIM acquired one previously, except that the call goes through admin_gate_ rather than through hcs_. The tseg used is a temporary ring1 segment created for the duration of the mount.

The event message associated with the wakeup sent to the tape system control function contains the index of the tsb and an action code. Possible actions are mount and dismount.

Having successfully sent a wakeup, trm_$mount sets its return arguments, resets the validation level, and returns to its caller.

Diagram 3.8: Events in a Mount Request -- operator message

Receipt of Mount Request by the Tape System Control Function

When called by the wait coordinator, tape_opr_ first checks to see whether the wakeup came from a user or the repeat timer. If the timer, it repeats all pending mount requests more than five minutes old and returns.

After extracting from the event message the action indicator and the index of the tsb containing the request, tape_opr_ verifies the authenticity of the wakeup by means of several tests. The wakeup must have originated in ring1, the sender's process id must match the process id in the tsb, that process must exist, and the state indicator in the tsb must be set to requested or pending.

The tsb is updated by a call into ring1 to reflect the receipt of the request. Then, as described above, a message is written to the operator and a timer set to ensure that the request is not forgotten.

```
User.Project                          Initializer.SysDaemon
  r4     r1                    r4                          r1

blocked          Random logins, etc.

                 Operator command:
                      tape   ok AUT
                      invokes
                       tape_opr_$reply

                 check index

                 check authentication

                 accounting

                 call tape_admin_              tape_master_
                      $tape_mounted                 $tape_mounted

                                               check tsb.state

                                               set tsb.state = 4

                                               decrement tcs.n_pending
                                               increment tcs.n_mounted

                                               return;

                 send wakeup to user

                 return;



                 More logins, etc.
```

Diagram 3.C: Events in a Mount Request -- Operator Response

Processing the Operator's Response

Upon receiving a positive reply from the operator the tape system control function verifies the tape by matching the typed authentication code with that previously calculated and placed in the tsb. If authentication is not required for the particular tape, this may be done anyway if the operator typed the authentication code, or may be skipped.

Next, accounting routines are called to start charges for the mount and the tsb is updated to reflect the tape's new state.

Upon receipt of a negative keyword the tape scf turns off any accounting that may be running, dismounts the tape, and frees the tsb.

Finally, an interprocess wakeup is sent to the user. The event message for this wakeup contains the keyword from the "tape" command line. The "kill" keyword does not cause a wakeup to be sent to the user.

```
          User.Project                        Initializer.SysDaemon
    r4                  r1                      r4                  r1

blocked
                                              send wakeup to user

                                              return;

call trm_    tape_mount_
$mount_check    $mount_check
                                              More logins, etc.
            set validation
                level = 1

            verify tcs_index, etc.

            check tsb.state

            reset drive status

            look for tape at
                load point

            set density

            look for write ring

            read/write vol label

            promote drive

            set tsb.state = 5

            reset validation level

            return;


   .
   .
   .
   .
   .
return;
```

Diagram 3.0: Events in a Mount Request -- mount_check

Final verification and Promotion of Mounted Tape

Upon receiving a wakeup from the tape system control function, and before attempting to manipulate the tape, the user's JSM must call trm_$mount_check for final verification of the tape. Primarily this consists of reading the label or of reading and then writing the label on the tape. But before doing so mount_check takes some precautions.

First mount_check verifies that the supplied tcs index is a valid one, that the designated tsb belongs to the same process, and that the tape is mounted and awaiting final approval. An error here is not immediately fatal; no more processing is done and an error code is returned to the caller. Some of these are errors from which the DSM cannot recover and the DSM should abort the mount by calling trm_$dismount if the attach switch is on. Others occur on premature calls; i.e. calls made before the tape scf has notified the user that the tape is ready.

Next the status of the drive is examined. If the drive is not in the ready state a new wakeup is sent to the tape system control function to cause the operator to remount the tape. The action and message variables in the tsb are updated to inform the operator of what is happening. An error code is returned to the DSM to indicate that it should block again to await correction of the problem. If the tape is not at the load point it is unloaded and similarly remounted.

When the tape is properly mounted and positioned, the drive is set to operate at the recording density indicated in the registration. An error here causes the mount to be aborted.

If the tape is mounted for writing, a write ring must be present in the reel; for reading the ring must not be present. Upon failure of either condition the tape is remounted as described above.

Once the tape is ready to go, label verification proceeds apace. Most errors cause the mount to abort. However, when reading some errors cause the tape to be remounted. This is in cases where the drive appears to need operator attention or where the wrong tape may have been mounted and the remount should be the correct one.

After writing a label on a not yet labeled tape, the registration is updated to reflect the presence of the label.

The tape is turned over to the user by a call into the hardcore to promote the drive. This raises the validation level to the validation level of the calling DIM. The state indicator in the tsb is changed once again to take note of the promotion and mount_check returns to the caller. The DIM is now free to go about its business.

```
        User.Project                                  Initializer.SysDaemon
    r4              r1                            r4                    r1

 DIM                                  Random logins, etc

call trm_   tape_mount_
$dismount      $dismount

            verify tcs_index

            check tsb.state

            set action indi-
             cators in tsb &
             in event message

            send wakeup to          tape_opr_
                 tape scf
                                    verify message
            reset validation
                 level              extract action
                                         indicator
            return;
                                    write message to
                                        operator
     .
     .                              finish accounting
     .
     .                              call tape_admin_      tape_master_
     .                               $tape_dismounted     $tape_dismounted
 return;
                                                        check tsb.state

                                                        release drive

                                                        decrement
                                                          tcs.n_pending or
                                                          tcs.n_mounted

                                                        free tsb

                                                        return;


                                    return;


                                    More logins, etc
```

Diagram 4: Sequence of Events in a Tape Dismount

Name: Volume Description Segment

A Volume Description Segment (VDS) is maintained for each registered magnetic tape (see writeup of tape_register in the SPS). It contains information which is used in mounting the tape (see tam_$mount in the SWU).

A special mechanism is used to reduce the number of segments of non-zero length needed to register all the tapes in the system. The bitcount in the branch of a VDS is used to indicate whether to use the VDS itself or whether to use an archetypical VDS which serves to record the attributes of several identical VDS's. The value in the bitcount is zero if the VDS itself is to be used, an index in a table of archetypes if an archetype is to be used. The archetype VDS has the same structure as an individual VDS.

All references to an individual VDS should be made through vds_manager_ which automatically locates the archetype if necessary. This is especially important when modifying a VDS; vds_manager_ is careful to copy the archetype into the individual before making changes.

These administrative-ring segments reside in a special system library. The entry name of a VDS must end in ".vds".

Usage:

```
declare
1 vds aligned based (vdsp),
     2 flags like tape_registration_flags,

     2 version                          fixed bin,
     2 size_vds                         fixed bin,
     2 start_nonconst_data              fixed bin,
     2 retention_date                   fixed bin (71),
     2 read_error_count                 fixed bin,
     2 write_error_count                fixed bin,
     2 density                          fixed bin,
     2 int_id                           char(8),
     2 accessibility_indicator          char (1);
```

Volume Description Segment

1)	vds	is the structure of a Volume Description Segment.
2)	flags	contain most of the description of the tape. See tape_registration_flags below.

3)	version	is included in case this structure must be changed without destroying all old VDS.
4)	size_vds	is not used.
5)	start_nonconst_data	is not used.
6)	retention_date	is not used.
7)	read_error_count	is not used.
8)	write_error_count	is not used.
9)	density	is the recording density if vds.flags.not_high_density = "1"b. 0 = 200bpi, 1 = 556bpi, 2 = 800bpi, 3 = 1600bpi.
10)	int_id	is the internal volume name in the tape label if it is different from the slot number.
11)	accessibility_indicator	is not used.

Note: Several variables have been included in this declaration in anticipation of future enhancements.

The declaration below is for the flag word in both the Volume Description Segment (VDS) and the Tape Communication Segment (TCS). This declaration may also be used to reference the registration flags argument returned from trm_$mount_ (tape_mount_$mount).

```
declare
1 tape_registration_flags based,
          2 not_multics_standard        bit(1) unal,
          2 ans_label                   bit(1) unal,
          2 os_label                    bit(1) unal,
          2 dos_label                   bit(1) unal,
          2 pad_label                   bit(5) unal,
          2 no_label                    bit(1) unal,
          2 not_yet_labeled             bit(1) unal,
          2 no_avr                      bit(1) unal,
          2 no_authenticate             bit(1) unal,
          2 tape_mark                   bit(1) unal,
          2 has_retention_date          bit(1) unal,
          2 has_access_indicator        bit(1) unal,
          2 int_id_diff                 bit(1) unal,
          2 unreg                       bit(1) unal,
          2 pad_mountact                bit(3) unal,
          2 seven_track                 bit(1) unal,
          2 not_high_density            bit(1) unal,
          2 small_reel                  bit(1) unal,
          2 pad_physical                bit(3) unal,
          2 free                        bit(1) unal,
          2 user_owned                  bit(1) unal,
          2 user_charged                bit(1) unal,
          2 pad_admin                   bit(6) unal;
```

Tape Registration Flags

1)	tape_registration_flags	is the structure of the regsitration flag word.
2)	not_multics_standard	is "1"b if the tape is not written in the multics standard format.
3)	ans_label	is "1"b if the tape is written in the ANSI tape format.
4)	os_label	is "1"b if the tape is written in the IBM OS tape format.
5)	dos_label	is "1"b if the tape is written in the IBM DOS tape format.
6)	pad_label	is unused.
7)	no_label	is "1"b if the tape is written in an unsupported format.
8)	not_yet_labeled	is "1"b for a new tape or one whose volume label should be rewritten when the tape is mounted for writing.
9)	no_avr	is "1"b if the ordinary label checking mechanism should be bypassed.
10)	no_authenticate	is "1"b if the operator is not required to type authentication characters when responding to a mount request.
11)	tape_mark	is unused.
12)	has_retention_date	is unused.
13)	has_access_indicator	is unused.
14)	int_id_diff	is "1"b for a tape whose internal volume name in the label is different from the slot number.
15)	unreg	is "1"b for an unregistered tape. This is meaningless in vds.flags.
16)	pad_mountact	is unused.
17)	seven_track	is "1"b for a seven_track tape.
18)	not_high_density	is "1"b for a tape with recording density other than 1600bpi.
19)	small_reel	is unused.
20)	pad_physical	is unused.
21)	free	is "1"b for a tape in a generic pool if the tape may be assigned.
22)	user_owned	is "1"b if the physical volume belongs to the user rather than to the installation.
23)	user_charged	is "1"b if the user rather than the system is responsible for the slot rental costs.
24)	pad_admin	is unused.

Name: tcs

The Tape Communication Segment(tcs) is an administrative-ring segment which is used for communication between the administrative-ring tape mounting programs in a user's process and the tape system control function. An entry is allocated in this table whenever a tape mount is requested. The entry is freed when the tape is dismounted.

Usage:

```
dcl 1 tcs based (tcsp) aligned,
    2 nents fixed bin,
    2 mount_proc bit (36),
    2 mount_chan fixed bin (71),
    2 n_mounted fixed bin,
    2 n_pending fixed bin,
    2 freep fixed bin,
    2 lock bit (36),
    2 pad (24) fixed bin,
    2 array (100),
      3 fill (128) fixed bin;
```

Tape Communication Segment

1)	tcs	is the structure of the Tape Communication Segment.
2)	nents	is the size of the table. It is initially 0 and is incremented only when the free storage list is exhausted.
3)	mount_proc	is the process id of the tape system control function.
4)	mount_chan	is the event channel over which the tape system control function is notified of a mount or dismount request.
5)	n_mounted	is the current number of reels mounted.
6)	n_pending	is the current number of reels waiting to be mounted.
7)	freep	is the index of the first block on the free storage list.
8)	lock	is a lock used during allocation and freeing of tape status blocks.
9)	pad	is unused.
10)	array	is the array of tape status blocks as described below.

```
dcl 1 tsb based (tsbp) aligned,
    2 state fixed bin,
    2 procid bit (36),
    2 evchn fixed bin (71),
    2 uname char (24),
    2 uproj char (12),
    2 tape_reel_id char (16),
    2 drive_id fixed bin,
    2 channel_id fixed bin,
    2 devx fixed bin,
    2 authentication char (4),
    2 switches,
      3 attsw bit (1) unal,
      3 comment bit (1) unal,
      3 ring    bit (1) unal,
      3 authreq bit (1) unal,
      3 pad     bit(32) unal,
    2 time_requested fixed bin (71),
    2 time_mounted fixed bin (71),
    2 time_dismount fixed bin (71),
    2 flags like tape_registration_flags,
    2 density        fixed bin,
    2 vdsp           ptr,
    2 tsegp          ptr,
    2 mode           char(8),
    2 int_id         char(8),
    2 act_chars      char(12),
    2 ourcomment     char(12),
    2 user_comment   char(120),
    2 qualifiers     char(64),
    2 tape_label     char(80),
    2 pad(17)        fixed bin,
    2 chain          fixed bin;
```

Tape Status Block

1) tsb — is the structure of the information kept for each individual mount request.

2) state — keeps track of the progress of the mount request: 0 = tsb is free, 1 = mount requested, 2 is not used, 3 = mount pending, 4 = mounted, 5 = promoted.

3) procid — is the identifier of the process requesting the tape.

4) evchn — is the event channel over which the requesting process is to be notified when the tape has been mounted.

5) uname — is the login name of the user requesting the tape.

6) uproj — is the project name of the user requesting the tape.

7)	tape_reel_id	is the volume name of the requested tape.
8)	drive_id	identifies the drive on which the tape is (to be) mounted.
9)	channel_id	is unused.
10)	devx	is unused.
11)	authentication	is the authentication code calculated from the volume id which is to be matched to the authentication code that the operator types.
12)	switches	indicate how to handle this mount or drive.
13)	attsw	is "1"b if a drive has been assigned for this request, "0"b otherwise.
14)	comment	is "1"b if a user comment is supplied for inclusion in the mount message, "0"b otherwise.
15)	ring	is "1"b if the tape is to be mounted for writing(ring needed), "0"b for reading.
16)	authreq	is "1"b if the operator is required to type authentication characters for this tape.
17)	pad	is unused.
18)	time_requested	is unused.
19)	time_mounted	is unused.
20)	time_dismount	is unused.
21)	flags	describe the requested tape. See the writeup of the Volume Description Segment(VDS).
22)	density	is the recording density if tsb.flags.not_high_density = "1"b. 0 = 200bpi, 1 = 556bpi, 2 = 800bpi, 3 = 1600bpi.
23)	vdsp	points to the Volume Description Segment for the requested tape.
24)	tsegp	points to the internal (ring1) tseg used by the user-process mounting routines in dealing with tdcm.
25)	mode	is the attachment mode as specified by the user. It may be "r" for read or "w" for write.
26)	int_id	is the internal volume name in the tape label if it is different from the slot number.
27)	act_chars	is used in messages to the operator to specify what action is needed.
28)	ourcomment	is used in messages to the operator. It is normally "" but in unusual circumstances supplies additional information to the operator.
29)	user_comment	is a comment supplied by the user to aid the operator. This comment is currently not printed, even if supplied.

30) qualifiers	is a copy of the qualifier string from the tape description argument passed by the user DIM to frm_$mount.
31) tape_label	is unused.
32) pad	is unused.
33) chain	is the index of the next free tsb when the tsb entry is free and on the free storage list.

Tape Qualifiers

The calling sequences to the tape registration commands and to trm_$mount contain a character string description argument with a special internal syntax. The first six characters must be the volume name of the volume to be registered or mounted. This volume name may be followed optionally by a comma and qualifiers, separated by commas, which describe the tape. Some examples:

```
"070090,ml,d=800,7track"
"CAT073,asl,unregistered,c=Marked from Waltham,d=800"
```

The following is a list of these qualifiers and their meanings. Note that a qualifier ending in "=" is extended to include a value, as in "d=800" in the examples above.

Qualifier	Meaning
"ml"	Multics Label: The tape is written in the Multics tape format.
"asl"	Ansi Label: The tape is written in the ANSI tape format.
"osl"	OS Label: The tape is written in the IBM OS tape format.
"dosl"	DOS Label: The tape is written in the IBM DOS tape format.
"nl"	No Label: The tape is written in a format which is not supported by the system.
"nyl"	Not Yet Labeled: New tape or one whose volume label should be rewritten when mounted for writing. When such a label has been written, the registration will be changed automatically to reflect the presence of the label.
"blp"	Bypass Label Processing: No automatic volume recognition should be attempted; i.e. ignore the label. (This feature currently is not implemented.)
"7track"	Seven track tape.
"9track"	Nine track tape.
"den="	Recording Density: The value following the "=" should be "0", "1", "2", or "3" with meanings: "0" = 200bpi, "1" = 556bpi, "2" = 800bpi, "3" = 1600bpi.
"density="	Recording Density: The value following the "=" should be "200", "556", "800", or "1600".
"d="	Recording Density: Abbreviation for "density=".
"retention="	Retention Date: The tape should not be mounted for writing until after the time indicated by the value following the "=". (This feature currently is not implemented.)

"rd="	Retention Date: Abbreviation for "retention=".
"unregistered"	Unregistered Tape: The tape has no VDS. This qualifier is meaningful only when mounting a tape.
"c="	Comment: The value following the "=" is included in the mount message written to the operator. This qualifier is meaningful only when mounting a tape. (This feature currently is not implemented.)
"label="	Internal volume name in the tape LABEL: if different from slot number.

The default both for registration and for mounting is:

```
mt,9track,d=1600
```

Name: tape_register

This privileged command may be used by an operator or system administrator to register one or more magnetic tapes (creating a Volume Description Segment(VDS) for each tape registered), or to create an archetype VDS (which can be used to reduce the storage required for registration of several tapes with identical registrations. See the write up of the VDS in the SPS.)

Entry: tape_register

This entry registers one or more individual tapes.

Usage

tape_register description person.project -ctargs

or

tape_register -from description -to volid2 person.project -ctargs

description	is the volume name(volid1) followed by qualifiers which describe the tape, separated by commas. These qualifiers are listed in a separate Appendix.
person.project	is the access name of the user to whom the tape is to be registered. A user name and a project name must be typed in full: asterisks are not allowed. The instance tag, if present, may be an asterisk.

-ctargs (optional) may be as follows:

-free	indicates that the given volume, if a member of a generic pool, may be assigned when a volume is requested from the pool.
-user_owned -uo	indicates that the user, rather than the system, owns the physical volume(e.g. the plastic tape).
-user_charged -uc	indicates that rental and storage charges for this tape will be billed to person.project.
-no_authenticate -na	indicates that the operator should not be required to type authentication characters for this tape if it can be verified in some other way(i.e. from the internal label).

The second form of the calling sequence allows several tapes to be registered with one invocation of the command. If "-from" is specified, "-to volid2" must also be specified. volid2 must not be followed by qualifiers; all qualifiers and options specified will apply to all tapes registered in this sequence. Notice that the first three characters of volid1 and volid2 are treated as a prefix; thus only 1000 tapes may be registered in one invocation of tape_register. The prefix part of volid1 and volid2 must be the same.

When a volume is first registered, the user has rwaco access.

Examples

```
tape_register 070088,7track,d=800 PDQSmith.Forge -uc
```

This registers tape 070088 with the attributes seven track and Multics format, and recording density 800bpi. The owner, PDQSmith of the Forge project, is charged for tape and slot rental.

```
tape_register -from CAT001,nyl -to CAT050 Backup.SysDaemon -na
```

This registers the fifty tapes CAT001 through CAT050 with attributes nine track, Multics format, not yet labeled, and authentication not required; recording density 1600bpi; and owner Backup.SysDaemon.

The first time each tape is mounted for writing a label will be written and the not_yet_labeled bit turned off in the VOS. Once this proper label is written, the operator will not be required to type authentication characters when responding to subsequent requests for this tape; until then authentication is required.

Entry: tape_register_archetype

This entry creates an archetype VOS.

Usage

```
tape_register_archetype description -ctargs
```

description is as above except that the volume name (which in this context is the type name) may be up to 28 characters long.

-ctargs (optional) may be any of the control arguments above or as follows:

-directory -dr indicates that the archetype VDS is to be placed in the directory whose full pathname is given by the next argument. The default pathname is >system_control_1>tape.

Example

```
tape_register_archetype old_backup,d=800,nyl -na -dr >sc1>tape>UF
```

This creates an archetype VDS for a class of tapes with name "old_backup". Such a tape is a Multics format tape recorded in nine tracks at a density of 800bpi. Currently it is not labeled and consequently authentication is required when it is mounted; however, once a valid label has been written on it authentication will not be required.

Subsequent to this use of tape_register_archetype the command:

```
tape_register UFY001,nyl,d=800 Backup.SysDaemon -na
```

will register tape UFY001 with its registration attributes in the segment >sc1>tape>UF>old_backup.vds, as described in the writeup of the VDS in the SPS.

Name: tape_unregister

This privileged command may be used by an operator or system administrator to terminate the registration of a tape, or to deactivate an archetype VDS. The Volume Description Segment(VDS) for the tape is deleted; an archetype VDS may be deleted or not according to control arguments.

Usage

tape_unregister volid

or

tape_unregister -from volid1 -to volid2

volid is the name of the tape to be unregistered.

The second form of the calling sequence allows several tapes to be unregistered with one invocation of the command. If "-from" is specified, "-to volid2" also must be specified. Notice that the first three characters of volid1 and volid2 are treated as a prefix; thus, only 1000 tapes may be unregistered in one invocation of tape_unregister. The prefix part of volid1 and volid2 must be the same.

Entry: tape_unregister_archetype

This entry deactivates an archetype VDS.

Usage

tape_unregister_archetype typename -ctargs

typename is the name of the archetype to be deactivated.

-ctargs (optional) may be:

-stop -sp indicates that the archetype should not be used for any subsequent registrations, but does not delete it. This is the default.

-delete -dl indicates that the archetype is to be deleted. This is expensive for it entails a search of all tape registrations to find all that use this archetype.

Examples

```
tape_unregister 070088

tape_unregister -from CAT001 -to CAT050

tape_unregister_archetype old_backup
```

Names: tape_registration_print,trp

The tape_registration_print command prints a description of the named volume as found in the Volume Description Segment(VDS) for that volume.

Usage

tape_registration_print volid

volid is the volume name of the tape for which a description is desired.

Example

trp 070088

produces output of the form:

```
Description of tape 070088
          multics standard
          seven track tape
          density                   800
          user charged
```

Names: tape_registration_change,trc

The tape_registration_change command allows the owner of a tape to change the tape's registration in a well controlled and well supervised way. Only information pertaining to the specified qualifiers will be changed unless the "-reset" control argument is specified also.

Usage

tape_registration_change description -control_args

description is the volume name followed by qualifiers which describe the tape, separated by commas. These qualifiers are listed in a separate Appendix.

-control_args (optional) may be as follows:

-reset -rs indicates that the user wishes to reset all fields in the Volume Description Segment(VDS) to default values before setting the specific fields implied by the qualifiers.

Notes:
1) The user must have o access to the volume whose registration is to be changed.
2) Some administrative and other privileged information will not be changed by the -reset control argument.

Examples

Let tape 070088 be registered as follows:

```
Description of tape 070088
          multics standard
          seven track tape
          density                    800
          user charged
```

Then the command:

```
trc 070088,asl,nyl
```

will change the registration to:

```
Description of tape 070088
          ansi standard
          not yet labeled
          seven track tape
          density                    800
          user charged
```

Given the same initial registration as above, the command:

```
trc 070088,asl,nyl -reset
```

will change the registration to:

```
Description of tape 070088
          ansi standard
          not yet labeled
          density                    1600
          user charged
```

Names: tape_setacl, tsa

This command adds items to the Access Control List (ACL) of a magnetic tape.

Usage

tape_setacl volname ac1 acname1 ... acn acnamen

volname is the volume name of the tape whose ACL is to be updated. It must be a six character string. The star convention is not supported.

aci is the access of the access control name, specified by the next argument, with respect to volname. It may consist of any or all of the letters "rwaco" or, to deny access to acnamei, it may be "n" or "null" or "".

acnamei is the access control name which is to receive aci access to volname. It must be of the form personi.projecti.tagi. Any of the components may be omitted, but if one is missing on the left, it must still be delimited by dots. The character "*" will be substituted for any missing component. (See Examples below.) Only the argument acname1 may be omitted, in which case the user's process group ID with a tag of "*" will be used.

Note A Brief Explanation of the Access Modes for a Tape Volume

access code	mode name	explanation
r	read	User may read the contents of the volume.
w	write	User may write anywhere on the volume.
a	append	User may write on the volume after data which is currently on the volume, but may not alter what is currently on the volume.
c	control	User may give to or remove from another user (user2) r,w, or a access if user has it, but may not give or remove c access.
o	owner	User may give or remove any access except o. There may be only one owner per tape and this access is set by a system administrator at the time the tape is registered.

Any combination of rwa is allowed.

Examples

.Multics .Multics. *.Multics *.Multics.
.Multics.* *.Multics.*

will all appear on an ACL as *.Multics.*

tsa IPC007 rw Joe.Proj

will add to the ACL of the tape IPC007 the access control name Joe.Proj.* with read and write access if the User issuing the command has o or rwc access, read if the User has rc, write if wc.

tsa 070090 rwac *

will add the access control name *.*.* to the ACL of tape 070090 with rwac access if the User issuing the command has o access, or the same combination of rwa that the User has if the User has c access but not o.

Names: tape_listacl,tla

This command lists some or all of the items on the Access Control List(ACL) of a magnetic tape.

Usage

tape_listacl volname option acname1 ... acnamen

volname specifies the tape whose ACL is to be listed. It must be a six character string. The star convention is not supported.

option May be "-all" or "-a". See description of acnamei below.

acnamei is an access control name. If there is no acname specified, or if the option "-all" ("-a") is specified, the entire ACL of the tape will be listed. Otherwise, acnamei must be of the form namei.projecti.tagi. If all three components are present, the ACL is searched for the identical name. If one or more of the components is missing, the ACL is searched for all names with the given components. Note: any components missing on the left must still be delimited by dots; however, the dots may be omitted on the right. (See Examples below.)

Note: Access modes are described in the writeup of tape_setacl in the MPM.

Examples

tla IPC007 .Multics ..a

will list, from the ACL of tape IPC007, all items with project name Multics and all items with instance tag a.

tla 070090 Multics *.*.* Fred

will print an error message for Multics if it is not a person name on the ACL of tape 070090; will list *.*.* if it, as such, is on the ACL of tape 070090; and will list all items on the ACL of tape 070090 with the name component Fred.

Names: tape_deleteacl,tda

This command deletes some or all of the items on the Access Control List(ACL) of a magnetic tape.

Usage

tape_deleteacl volname option acname1 ... acnamen

volname specifies the tape whose ACL is to be deleted. It must be a six character string. The star convention is not supported.

option May be "-all" or "-a". See description of acnamei below.

acnamei is an access control name. If there is no acname specified, the user's process group ID will be assumed; if the option "-all" ("-a") is specified, the entire ACL of the tape will be deleted. Otherwise, acnamei must be of the form namei.projecti.tagi. If all three components are present, the ACL is searched for the identical name. If one or more of the components is missing, the ACL is searched for all names with the given components. Note: any components missing on the left must still be delimited by dots; however, the dots may be omitted on the right. (See Examples below.)

Note: Access modes are described in the writeup of tape_setacl in the MPM.

Examples

tla IPC007 .Multics ..a

will delete, from the ACL of tape IPC007, all items with project name Multics and all items with instance tag a.

tla 070090 Multics *.*.* Fred

will print an error message for Multics if it is not a person name on the ACL of tape 070090; will delete *.*.* if it, as such, is on the ACL of tape 070090; and will delete all items on the ACL of tape 070090 with the name component Fred.

Administrative Ring
05/14/74

Name tape_mount_

This administrative-ring procedure contains entries for mounting and dismounting magnetic tapes in the Multics System. Entry points may be accessed through the gate trm_.

Overview

Mounting magnetic tapes is done in five steps, two of which are performed by procedures operating in the administrative ring of the user's process, two of which are performed by a tape system control function operating in a system process, and one of which is performed by the human operator.

1) Initiate Request: verify access, set up tables, get drive (user process)
2) Notify Operator: write mount message (system process)
3) Mount: Put reel on drive and push appropriate buttons (human operator)
4) Process Operator Reply: authenticate, start accounting (system process)
5) Final verification: set density, check write-protect, read/write , promote drive (user process)

This procedure performs steps 1 and 5. The user's tape Device Strategy Module(DSM) calls trm_$mount to initiate a mount request. Upon return, the DSM should block waiting for steps 2 through 4 to be completed, at which time the tape system control function will send a wakeup to the user's process. The DSM must then call trm_$mount_check to perform final verification of the tape before it may perform I/O on the attached drive. This is enforced by making the validation level of the drive equal to that of the administrative ring rather than to that of the user ring until the tape has been completely verified.

Other entries dismount the tape and supply information about the status of a request.

Entry tape_mount_$mount

This entry provides the ring 1 mechanism to attach a tape drive and mount a tape for a Device Strategy Module (DSM). A subsequent call to tape_mount_$mount_check must be performed before the DSM may issue I/O to the attached device.

Usage

```
dcl tape_mount_$mount entry (char(*),fixed bin(71),char(*),
     ptr,fixed bin,fixed bin,fixed bin,bit(36),fixed bin(35));

call tape_mount_$mount (description,evchn,mode,dimdatap,
     writesw,drivenumber,tcsx,flags,ec);
```

1) description is the tape reel identifier plus optional qualifiers which describe the tape. The tape reel identifier must be a string six characters in length. The qualifiers are listed in a separate Appendix. Unless one of the qualifiers is "unregistered", all qualifiers are ignored and the description is taken from the tape's registration in its Volume Description Segment(VDS). If "unregistered" is present, the qualifiers are taken as the sole description of the tape and no VDS is sought. (Input)

2) evchn is the event channel with which the DSM will be signalled when the operator has acted upon the attachment request. (Input)

3) mode is the tape attachment mode. Any mode specification other than "r" (for read) or "w" (for write) will result in an error condition. (Input).

4) dimdatap is a pointer which allows communication between the caller and the labeler which is to verify the tape being mounted. The value which tape_mount_ recieves from the caller is passed to the labeler. The value that the labeler returns is returned to the caller. (Input and Output)

5) writesw is 0 for a read attachment, 1 for a write. (Output)

6) drivenumber is the number of the drive on which the tape will be mounted. (Output)

7) tcsx is the index in the Tape Communications Segment (TCS) of this tape mount request. It must be supplied for all subsequent calls to tape_mount_. (Output)

8) flags is tape_mount_'s internal description of the tape. (Output)

9) ec is a standard Multics error code. (Output) It may be:

Zero indicating that no error has occurred.

An error code indicating an incorrect volume identifier:

error_table_$bad_volid	Incorrect volume id
error_table_$bigarg	Tape identifier too long
error_table_$smallarg	Tape identifier too short
error_table_$nostars	Tape id contains stars

An error code indicating multiple requests for the same tape:

error_table_$bad_processid	Tape requested or mounted for another user.
error_table_$mount_not_ready	Tape requested but tape system control function has not yet acted on it.
error_table_$redundant_mount	Tape already mounted.

An error code indicating an access error:

error_table_$moderr Incorrect access or unrecognized attachment mode.

An error code indicating trouble locking TCS for allocation:

error_table_$lock_wait_time_exceeded
error_table_$locked_by_this_process
Error code from the primitive set_lock$unlock.

An error code indicating trouble finding the VDS or some other internal data base:

error_table_$not_seg_type Segment found when seeking the VDS is not a VDS.
error_table_$unregistered_volume VDS could not be found.
Error code from the primitives hcs_$initiate_count, hcs_$make_seg, admin_gate_$fs_get_ex_mode

An error code indicating an error while attaching a drive:

error_table_$device_limit_exceeded User has already attached maximum number of drives allowed.
error_table_$invalid_channel User has provided an invalid event channel.
error_table_$no_device No device available for attachment.

An error code from the primitive hcs_$wakeup indicating an error when using interprocess communication.

An error code indicating that the tape system control function is not available:

error_table_$nosys

Entry tape_mount_$mount_check

This entry is called by the DSM to validate the mounting of a tape volume. The drive is set to operate at the proper recording density, with write-protect hardware engaged as appropriate. The label is read, examined, and if appropriate rewritten. When the tape has been verified the validation level of the drive is raised to that of the caller so that the DSM may manipulate the tape.

Usage

```
dcl tape_mount_$mount_check entry (fixed bin, ptr,
      bit(1), fixed bin(35));

call tape_mount_$mount_check (tcsx, dimdatap, attsw, ec);
```

1) tcsx is the index in the TCS of the mount request. (Input)

2) dimdatap is as above. (Input and Output)

3) attsw equals "1"b if the tape is mounted, "0"b if not. If ec is non-zero and attsw = "1"b then it is the caller's responsibility to call trm_$dismount if aborting the attachment. (Output)

4) ec is a standard Multics error code. (Output)

Zero indicating that no error has occurred.

An error code indicating a premature call or unacceptable arguments:

error_table_$bad_index Incorrect tcsx: out of range.
error_table_$bad_processid Incorrect tcsx: tape mounted for or requested by another user.
error_table_$mount_not_ready Mount requested but tape system control function has not yet acted. This code may be returned either if tape_mount_$mount_check is called before the DSM recieves the wakeup from the tape scf, or if some erroneous condition requires that the tape be remounted. In either case the DSM should block waiting for the wakeup.
error_table_$redundant_mount Tape is already mounted and verified.
error_table_$bad_mount_request Mount request is in an improper state.

An error code indicating that the volume could not be verified due to lack of a label or to incorrect label:

error_table_$bad_label The label is not in the specified format.
error_table_$bad_volid The volume id in the label does not match the volume id of the request.
error_table_$no_label The tape has no label and was not authenticated.
Other fatal error codes from individual labeler

routines.

An error code indicating difficulties in dealing with the tape drive hardware:

error_table_$bad_density — Drive cannot handle the recording density specified in the registration.
error_table_$fatal_error — Serious error reported by tape drive.

An error code indicating that the Mount Package is not available:

error_table_$nosys

An error code returned by hcs_$wakeup indicating an error while using interprocess communication.

An error code from hcs_$initiate_count indicating trouble finding internal data.

Entry tape_mount_$dismount

This entry is called by the USM to rewind and unload a tape, and to deallocate the drive and to detach it from the calling process.

Usage

```
dcl tape_mount_$dismount entry (fixed bin, ptr, fixed bin(35));

call tape_mount_$dismount (tcsx,dimdatap,ec);
```

1) tcsx is the index in the TCS of this mount request. (Input)

2) dimdatap is as above. (Input and Output)

3) ec is a standard Multics error code. (Output)

Entry tape_mount_$status_i

This entry provides the status of a tape mount request given the index of its TCS entry.

Usage

```
dcl tape_mount_$status entry (fixed bin, fixed bin,
        char(*), fixed bin, fixed bin(35));

call tape_mount_$status (tcsx, status, reel_id, drive, ec);
```

1) tcsx is the index in the TCS of the request for

2) status is the status of the request. (Output) It may be: 1= tape mount request has been signalled but request has not been processed by the tape system control function; 2= unused; 3= tape mount is pending on the operator console; 4= tape has been mounted; 5= tape has been verified and promoted; 6= tape is being dismounted.

3) reel_id is the tape reel id of the requested volume. (Output)

4) drive is the tape drive number. (Output)

5) ec is a standard Multics error code. (Output)

Entry tape_mount_$status_r

This entry provides the status of a tape mount request given the identifier of the requested tape reel.

Usage

```
dcl tape_mount_$status_r entry (char(*), fixed bin,
        fixed bin, fixed bin, fixed bin(35));

call tape_mount_$status_r (reel_id, status, tcsx, drive, ec);
```

1) reel_id is the tape reel id of the requested volume. (Input)

2) status is the status of the request. (Output) It may be: 1= tape mount request has been signalled but request has not been processed by the tape system control function; 2= unused; 3= tape mount is pending on the operator console; 4= tape has been mounted; 5= tape has been verified and promoted; 6= tape is being dismounted.

3) tcsx is the index in the TCS of this mount request. (Output)

4) drive is the tape drive number. (Output)

5) ec is a standard Multics error code. (Output)

Labeler Routines

As described in the text, each tape DIM supported at an installation must provide routines to read and verify and to write volume labels. Such labelers will have different names (tape_labeler_, VOL1_labeler_), but standard calling sequences, which are given below.

Entry: DIM_labeler_$read

The responsibility of this entry is to read the volume label record(s) from the tape and to verify that they satisfy the specifications of the tape format implemented by the DIM. The volume identifier found in the volume label is returned to the Tape Mount Package for comparison with the name of the requested volume.

Usage

```
declare DIM_labeler_$read entry(ptr,ptr,ptr,char(*),
                    fixed bin,fixed bin(35));

call DIM_labeler_$read(ritsegp,dimdatap,tsbp,volid,type,code);
```

1) ritsegp — is a pointer to a temporary tseg created by the Tape Mount Package. This is the tseg that should be used for I/O. (Input)

2) dimdatap — is a pointer which allows communication between the caller and the labeler which is to verify the tape being mounted. The value which tape_mount_ recieves from the caller is passed to the labeler. The value that the labeler returns is returned to the caller. (Input and Output)

3) tsbp — is a pointer to the tape's entry in the TCS. This provides information to the labeler but should not be written in. (Input)

4) volid — is the volume id found in the label record. (Output)

5) type — currently is not used. It is always 0 when the call is made from the Tape Mount Package.

6) code — is a standard Multics error code. (Output)

Entry: DIM_labeler_$write

The responsibility of this entry is to write the volume label record(s) on the tape according to the specifications of the tape format implemented by the DIM. The supplied tape identifier should be used in the volume label.

Usage

```
declare DIM_labeler_$write entry(ptr,ptr,ptr,char(*),
                    fixed bin,fixed bin(35));

call DIM_labeler_$write(ritsegp,dimdatap,tsdp,volid,type,code);
```

1) ritsegp — is as above. (Input)

2) dimdatap — is as above. (Input and Output)

3) tsdp — is as above. (Input)

4) volid — is the volume id to be placed in the label record. (Input)

5) type — currently is not used. It is always 0 when the call is made from the Tape Mount Package.

6) code — is a standard Multics error code. (Output)