To: Distribution

From: Dennis Capps

Date: January 7, 1974

Subject: The Tape Mount Package, A Tape DIM, and tdcm

This interim memorandum explains how the new tape mount package will interact with a DIM and tdcm. It is intended to give a general understanding of this package, but not to be an exhaustive description. It will be expanded, cannibalized, and combined with the information in MSB-69 and included in other memoranda, SPS sections, and MPM sections in the near future.

Figure 1 is a graphic representation of this interaction. The tape mount package is indicated by the name trm_ and I will refer to it by that name in the remainder of the memorandum. This is actually the name of the ring1 gate which leads to the package.

The attach entry of a tape DIM calls trm_$mount where previously it would have called hcs_$tdcm_attach. This entry acquires a drive, sets up communications with the initializer process, notifies the initializer that a tape is to be mounted, and returns to its caller. The caller is then expected to block or otherwise wait for an interprocess wakeup from the initializer. An event call procedure in the initializer 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 DIM should call trm_$mount_check to verify that the correct tape has been mounted. This entry will also check for a write ring, set density, etc.

After a phase-over period, use of trm_ will be enforced in two ways. The entry tdcm_attach will be removed from hcs_ and such attachment will be available only in ring1 through admin_gate_$tdcm_attach. When a drive is assigned by tdcm, the validation level of the caller is recorded in tape_data. No subsequent call to tdcm from a higher numbered ring is honored until the drive is explicitly "promoted" by a call to admin_gate_$tdcm_promote.

During the mounting and verification of a tape, two data bases are used to obtain and store information about the tape and the current mount. The Tape Communication Segment(TCS) is a

per-system communication segment with entries for each tape mounted or whose mount is pending. In addition to information necessary for interprocess communication between trm_ and the initializer, an individual entry serves as a trm_-internal description of the tape in question.

The other major data base is the Volume Description Segment(VDS). (1) This per-volume ring1 segment, residing in a hierarchy under >system_control_1, is created when a tape is registered by the tape librarian and remains in the system until the tape is unregistered by the tape librarian. A command is provided to allow the user to modify parts of the registration for his/her tape. At mount time, it is the description in the VDS which trm_ uses to verify the user's access, allocate the proper kind of drive, set the proper density, etc.

A provision is made for mounting unregistered tapes. In this case all relevant information must come from the description argument (see descriptions of entries below).

The tape mounted may be verified 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 trm_ 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.

The actual reading and verification or writing of the label is performed by an entry in the DIM (DIM_labeler_). 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 trm_ 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.

The entry trm_$dismount cleans up and sends a wakeup to the initializer, which informs the operator, releases the drive, and frees the tape's entry in the TCS.

The preliminary version of this package is in the final shakedown stages and installation is planned for the end of January.

---

(1) In the past this has been called TRDS and BVDS.

Descriptions of Entries

trm_$mount

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

call trm_$mount(description,evchn,mode,tsegp,tcsx,flags,code);
```

1) description — is of form volid,qualifier1,...,qualifiern The qualifiers describe the format, density, number of tracks, etc. Unless one of the qualifiers is "unregistered", all qualifiers are ignored. If "unregistered" is present, the qualifiers are taken as the sole description of the tape and a VDS is not sought. (Input)

2) evchn — is the event channel over which the caller should be awakened when the tape has been mounted. (Input)

3) mode — must be "r" (for read) or "w" (for write). (Input)

4) tsegp — is a pointer to the caller's tseg data base with which it intends to communicate with tdcm. (Input)

5) tcsx — is the index of this tape's entry in the TCS. It must be specified on subsequent calls to trm_. (Output)

6) flags — is trm_'s description of the tape. (Output)

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

trm_$mount_check

```
declare    trm_$mount_check    entry(fixed   bin,ptr,bit(1),fixed
bin(35));

call trm_$mount_check(tcsx,tsegp,attsw,code);
```

1) tcsx — is as above. (Input)

2) tsegp — is as above. (Input)

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

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

trm_$dismount

```
declare trm_$dismount entry(fixed bin,ptr,fixed bin(35));

call trm_$dismount(tcsx,tsegp,code);
```

The arguments are as above.

The labeler program for a DIM (represented here by DIM_labeler_) is called by trm_ with the following calling sequences.

DIM_labeler_$read

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

call DIM_labeler_$read(r1tsegp,r4tsegp,tsbp,volid,type,code);
```

1) r1tsegp — is a pointer to a temporary tseg created by trm_. It is a copy of the user's ring4 tseg up to but not including the first buffer. This is the tseg that should be used for I/O. (Input)

2) r4tsegp — is a pointer to the user's tseg as passed to trm_. This allows the labeler to pass information from the label to the DIM. (Input)

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 trm_.

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

Dim_labeler_$write

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

call DIM_labeler_$write(r1tsegp,r4tsegp,tsbp,volid,type,code);
```

1) r1tsegp — is as above. (Input)

2) r4tsegp — is as above. (Input)

3) tsbp — 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 trm_.

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

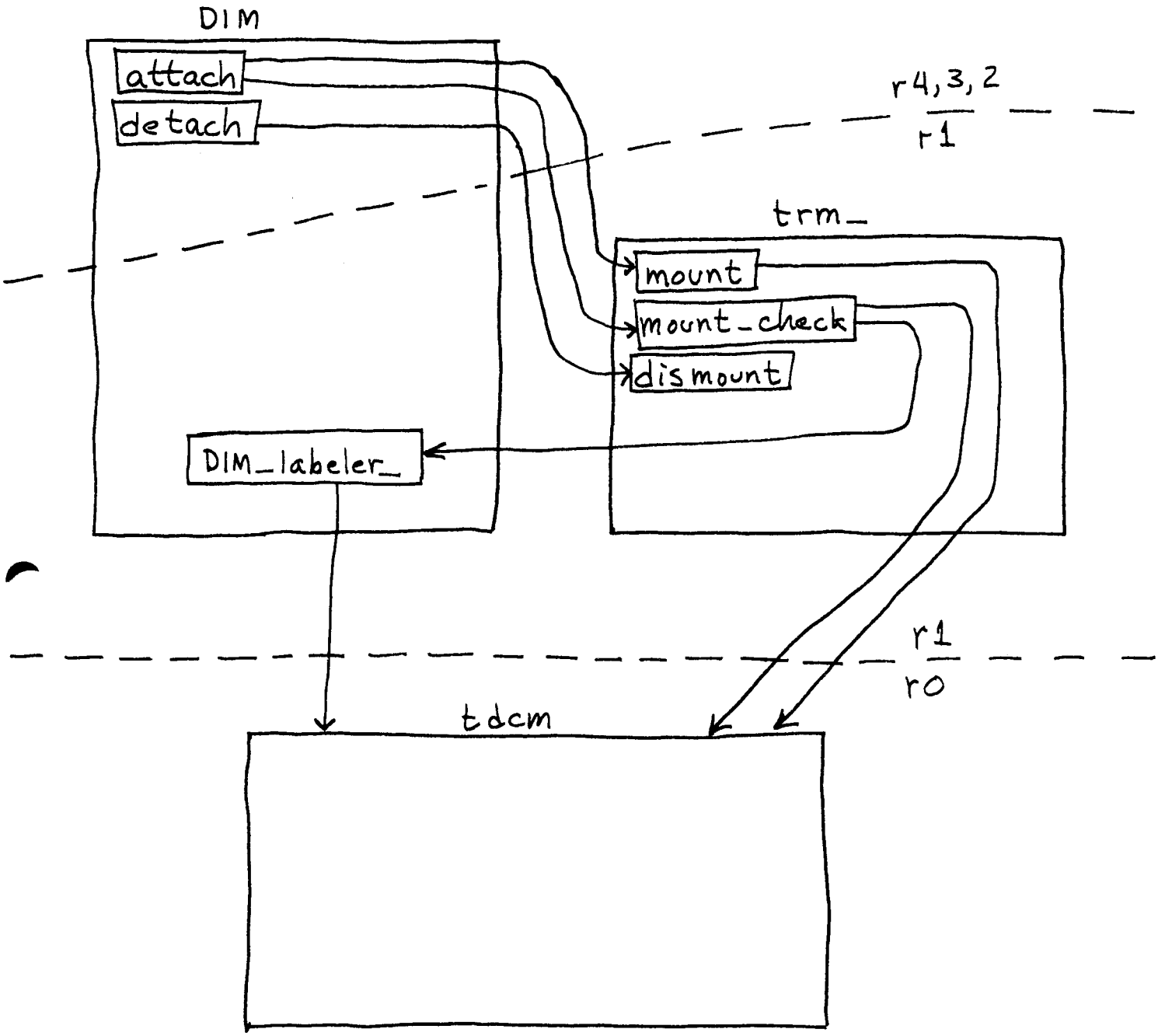

Figure 1 Interaction of The Tape Mount Package with a Tape DIM and tdcm