To: MTB Distribution

From: Bernard S. Greenberg

Subject: NSS Disk Definition

Date: 07/24/75

This memo was originally intended as an internal document within the new Storage System (NSS) design and implementation group. Many of the terms and concepts referred to within in it describe the current state of NSS, and are documented nowhere. The idea of publishing this memo as an MTB is to keep the Multics Development Community abreast of current design proposals and developments. The following other documents are prerequisite to the comprehension of this memo:

MTB 206	NSS SAVE and RESTOR
MTB 184	Resource Control Package
MTB 167	NSS Disk Usage
MTB 110	Implementation of Proposed NSS

Preamble

Disk Definition, for the purposes of this memo, may be defined as the specification, determination, and control, of what packs are mounted on what drives at any time, and the logical structure of what is on a storage system pack.

This memo outlines several schemes intended to define interfaces, protocols, and implementation of dynamic disk mounting and dismounting in the new Storage System (NSS). The current NSS specifies the mounted complement of packs via BOS CONFIG CARDS, and cannot yet change the pack complement dynamically. The data structures of the New Storage System, however, were designed to make this mounting and dismounting feasible.

The principal issues addressed herein are the dynamic redefinition of the mounted disk complement, the resource control issues of dismountable components of the storage system, the operator interfaces for controlling dismountable packs, and various protocols and policies for ensuring a unified approach to these and other issues.

Issues of double-writing volumes, pack initialization, appropriate CONFIG cards and the semantics of partitions are also covered here, as they are part of the general issue of disk definition.

I. Multics assumes all responsibility for mounting and dismounting.

Disk drives are a resource, and disk packs protectable data objects. The mounting and dismounting of disk packs is an operation performed under the supervision of the Resource Control Program (RCP). The specification of the online disk complement should be viewed as a dynamic reconfiguration operation, performed by software such as RCP, which can validate and process such operations appropriately.

It is not desirable to have BOS maintain the online disk complement, as is currently done. The specification of the online disk complement by CONFIG cards is messy and inadequate, as well as non-general and non-extensible. The restrictions on name length imposed by BOS are not reasonable. The limitation on the size of the CONFIG deck is also an unreasonable restriction. What is more, the CONFIG deck update mechanism (CONFIG U), which updates the CONFIG deck from Multics to BOS is inadequate and partly functional, and the wrong mechanism for the application. BOS does not see configuration changes made by Multics, and in general, that is a reasonable policy.

It is proposed that there be no information in BOS which describes the online disk complement. Hence, there should be no VOL cards, as NSS use them today. CONFIG cards are for specification of a configuration of hardware, not dismountable media. The CONFIG cards which currently describe software only describe maximum table sizes and other invariant information. Mounted volumes are not within this class of information.

II. The Disk Table

We propose the creation of a segment in the hierarchy which describes the online disk complement. This table, henceforth called the Disk Table, parallels the Physical Volume Table (PVT). It is not clear if hardcore even need know about this table. The Disk Table is a Resource Control data base. It will be initialized at the time the Resource Control Program initializes itself. Its header will be filled with a description of the current hardware disk subsystem configuration. The body of the table will be an array of entries, one for each drive currently configured or configurable into the system. The TRANSFORMATION FROM SUBSYSTEM ID and drive number, for a given configuration, to Disk Table index, will be canonical, and Disk Table indices are the same as PVT indices.

Mounting and dismounting of both so-called "I/O" and storage system volumes consists of the allocation and deallocation of slots in this table. Each entry contains device type and potential allocation information for a given drive, and when a pack is mounted, registration, identification, and other control information for the pack mounted on the drive. When a storage system pack is mounted, calls must be made into ring zero to place the Physical Volume ID in the PVT, and have the Volume Map accepted into the FSDCT. When a storage system pack is dismounted, a call must be made to have all resident segments deactivated, and the Physical Volume ID removed from the PVT. Mounting and dismounting of I/O packs consists of calling ring zero to route interrupts for the associated drives to and from the I/O interfacer (ioi_). The PVT will be used to tell the disk interrupt handler that this is the case for such a drive.

The Resource Control Program (RCP) will be responsible for the maintenance of the Disk Table. Calls into RCP will effect the orderly requesting of disk mounts, for both storage system and I/O packs. These programs, which run in the Administrative ring, will thus be responsible for determining the legitimacy of mount requests, and issuing appropriate operator messages. RCP will also make the necessary calls into the supervisor to allow use of mounted volumes. In order to mediate these requests, RCP will make use of data describing the ownership and usage rights of I/O disks and storage system Logical Volumes. This so-called Registration Data is described below.

Via the above scheme the operator can cause volumes to be mounted at startup time, either by explicit command, or exec_com. The calls into RCP to request a mount are the same as those made by user processes when asking for logical volumes to be mounted. RCP is responsible for determining the number of drives necessary for the logical volume (as defined in Registration Data), and the availability of that number of drives. Allocation of "privileged" or "reserved" drives is also a function of RCP. The intended use of a given drive as a "storage system drive" versus an "I/O drive", if desirable, will also be a policy matter enforced by RCP.

At boot time, ring zero registers all drives of all configured disk subsystems which are given as "potentially configurable" in the CONFIG deck. This registration consists of allocating PVT entries, and hence PVT indices, for these drives, and connecting the proper interrupt and device control linkages in ring zero. Note that ring zero has no responsibility for validating what is mounted on any of these drives, or automatically configuring these packs into the storage system.

Having described the ring zero (PVT) and ring one (Disk Table) conceptions of the storage system, we now introduce the concept of volume recognition. Volume Recognition consists of

ring one being told that a given selection of volumes is mounted on a given selection of drives, by some highly privileged agent, and believing it, as though a mount request had been issued. As with a mount request, label checks are made, date/time salvaged/dismounted is checked, etc. As with a mount request, the issuer of the request is trusted, i.e., he could mount a fraudulent volume, or have one recognized, as easily as he could mount the correct volume, and there is no way to certify volumes. Label checking is a check against non-intentional error.

With Volume Recognition and the Disk Table, we have all of the mechanism to bring up the system automatically without operator intervention, with a full complement of disks online from the previous bootload. When the system comes up, the physical drive configuration as given in the CONFIG deck is compared against that in the Disk Table, which is a remnant of the previous bootload. Any discrepancy in physical configuration causes the entire contents of the Disk Table to be scrapped, with an appropriate message. In any case, entries for two Physical Volumes (The Root Physical Volume (RPV), described in more detail below, and the optional Paging Device) are made, and marked as in use and validly mounted. The RPV is found via CONFIG card, described in more detail below.

At this point in startup, the Disk Table can be viewed as a driving table for generating Volume Recognition requests. Operator command can cause the entire contents of the table to be automatically recognized, recreating the online disk complement of the previous bootload. However, the operator can also "edit" the Disk Table, asserting that certain volumes have been mounted, dismounted, etc. Note that there is no way to prevent the operator from physically moving volumes, as he can press all of the available buttons. Note also the implication of the canonical transformation from subsystem-name drive-number into PVT index, given a supplied physical configuration, as this is prerequisite to accepting an old Disk Table. In the case of unattended operation, it is reasonable that the system start_up exec_com might issue the command which simply accepts the existent contents of the Disk Table wholesale.

Having possibly edited the Disk Table, the operator can then issue the command to recognize all of its contents. Thus, various degrees of boot are possible, the controlling factor being only the automatic recognition of volumes specified in the Disk Table.

Needless to say, all of the software and data necessary to carry this out, including the Disk Table itself, must be on the Root Physical Volume. This implies some special mechanism to cause system segments which are necessary to disk configuration to be allocated on the RPV, as opposed to any physical volume of the Root Logical Volume. A simple, if brutal, solution for this requirement is to place all such modules on the Multics System

Tape (MST).

III. What Mounting and Dismounting Mean.

When RCP has received and validated a request to mount a logical volume, drives are allocated, and Disk Table entries set up. Operator messages are then issued to cause physical mounting of the necessary packs. As the operator acknowledges each request, ring zero is invoked to hand control of the concerned drive to ioi_. This consists of marking the PVT so that interrupts will be directed to ioi_, and assigning the drive to the process performing the mounting. RCP now uses ioi_ to read the label of the pack mounted, and validate it against the registration information. Note that the PVT index may be supplied to ring zero, for it has been defined to be the same as the Disk Table index for the drive. If the label checks out, ring zero is called to deassign the drive from ioi_, and once more to assign the drive to page control. This assignment to page control consists of placing the Logical Volume and Physical Volume ID's in the PVT entry, marking the PVT entry so that page control handles the interrupts, and copying the Volume Map into the FSDCT. RCP may wait until all physical volumes of a logical volume are mounted before calling in to assign the drives to page control. If RCP dislikes the label, the drive is simply deassigned from ioi_, and error status returned.

In certain cases, partial logical volumes can be mounted. This is necessary in case part of a logical volume is physically damaged, and in the case of recognizing the root logical volume during startup. The access required to do this and the error codes returned upon attempt to initiate a resident of the nonexistent half of a partially mounted volume are policy issues. The ring zero handling of partial volumes is clear.

Requests to mount I/O disks are much the same. RCP receives and validates the request, again based on policy and registration data. A Disk Table entry representing an avaliable drive on a subsystem of appropriate type is allocated. Note that parameters representing installation policy as to I/O vs. storage system usage of given drives may also be a factor here. An operator message is issued, and a call made to assign the drive to ioi_. ioi_ is used to validate the label, and the drive is ultimately handed over to the outer ring user. It is deemed unwise to have volume recognition apply to I/O disks, as an I/O disk is always requested by a given process, and not asserted to be mounted by the operator.

Recognition of disks by explicit operator command consists of checking the Disk Table to insure that the pack is not already mounted, and for the availability and applicability of the specified drive, and the assignment of the Disk Table entry to the volume specified. This "drive/pack specification" is

also one of the Disk Table "editing" options described above.

Dismounting of a storage system volume is initiated by an agency yet unspecified. Whether the operator issues a command, RCP finds a more worthy user of a drive, or the agency who requested the mount logs out is as yet an open question. At any rate, RCP is notified, and the request is validated and processed by RCP. If the request is legitimate, ring zero is notified of the dismount status of the applicable PVT entries. From that point on, a switch in the PVT prohibits further segment faults, boundsfaults and initiations on that volume. The AST is scanned, and all segments residing on that physical volume are deactivated. This deactivation implies flushing them off of the paging device as well. The walking or recursing nature of this AST scan is a function of the policy decision which prohibits dismounting volumes containing directories. When the scan is complete, a call to vtoc_man is made to flush all VTOC buffers of the physical volume, and the PVT entry is cleared out. Control returns to RCP, and the Disk Table entry is deallocated.

Dismounting of I/O volumes is performed by RCP as well. Here, the PVT entry is cleared, the drive returned to RCP, and the Disk Table entry deallocated.

IV. Cold Boots

The definition of a so-called "cold boot" must be changed somewhat, from both the current Multics and current NSS Multics definitions. A "cold boot" is usually defined as a bootload which assumes that no valid information exists anywhere on disk, and that Multics must entirely initialize the hierarchy. In both the current system and the current NSS, "COLD" is a parameter which is passed to Multics by the BOOT command of BOS. Current NSS Multics also initializes all mounted volumes at cold boot time, writing labels and VTOCs. In both the installed and NSS systems, a root directory is created at cold boot time.

It is proposed that the initialization of volumes be removed entirely from ring zero. Since the system will always come up with only one volume, as the proposals above outline, only the initialization of the RPV at cold boot time is an issue. Other packs may be initialized by the online disk initializer program (see below), which can be run once the system is up.

The RPV may be initialized by an earlier run of the Multics system, or by a BOS utility program. The task of initializing the label and the VTOC is quite simple, and the only real argument against such a program is the embedding of knowledge of VTOC formats in BOS.

If Multics is booted on a void RPV, i.e., one which is not indicated as posessing a root directory, and the BOOT COLD command was issued, a root directory is created. Hence, BOOT

COLD will never void the contents of an existing RPV. In order to scratch a pack, so that a root may be created there, one must first reinitialize that pack.

V. The Root Physical Volume, and Defining it.

The Root Physical Volume has been spoken about loosely up to this point, and conflicts, even in name, with the current definition of Physical and Logical Volumes in the New Storage System. Let us therefore devote some space to defining this entity.

The Root Logical Volume is currently well defined in NSS. Further define the Root Physical Volume as that Physical Volume of the Root Logical Volume (henceforth RLV) on which the Root Label resides.

We have specified above that all software necessary to reconfigure the online disk complement is on the RPV. As we envision adding and recognizing packs into a logical volume, it seems reasonable that the rest of the RLV, i.e., other than the RPV, can be added or recognized into the RLV at boot or any other time.

It is a design goal that the system should be capable of running with just one pack, the RPV. Not only is this useful for debugging and test purposes, but allows the system to come up in a uniform fashion, and have disks added dynamically. It has also been given as a marketing requirement.

The contents of the RPV at boot time defines the entire consciousness of a bootload. All software, the Disk Table, and hierarchy definition are found from this volume. Hence, Multics must know, at boot time, where the RPV is located. I propose one CONFIG card, ROOT, or RPV, which specifies the subsystem name and drive number on which the RPV is mounted. Some label-checking information might, but need not, also be included. One might use this flexibility to advantage to switch ROOT cards between development and service runs, given that one has enough drives for this sort of thing.

VI. Partitions

Partitions are disk areas not used for ordinary Multics segments, but for special objects, such as EDUMP images, the syserr log, and salvager temporary allocation. They are currently described by PART cards. The current format of PART cards describing address extents is inadequate. Only a pack label can describe the extents and locations of various areas of the pack: external cards cannot attempt to specify this information without useless redundancy or error. It is proposed that the only definition of partition locations and extents be that in pack labels. The online disk initializer program (see

below) can be used to define partitions, arbitrarily-named extents, on any physical volume. PART cards will specify a disk subsystem and drive number, like the ROOT card. The disk pack pointed to for such a card will have its label read, and the partition of the appropriate name located and used. The treatment of partitions in the volume map is a function of the intended use of each partition.

The following CONFIG deck excerpt shows the intended discipline:

```
DISK D19A A 30. 191. 8 8 1 *defines a disk subsystem
                             named "D19A", on chan. 30.
DISK D19B A 40. 191. 8 8 1 * defines another
ROOT D19A 3                  * The RPV is found on
                               drive 3 of D19A.
PART DUMP D19B 6             * The DUMP partition is on
                               6 of D19B, as per
                               its label.
PART LOG D19A 3                * Syserr log on the RPV.
```

Each drive which is pointed to by a PART card is "wired" in the Disk Table, and cannot be dismounted. It can only have its mounted volume recognized at startup time.

When BOS needs to know about the location and extents of some partition, it can read the appropriate PART cards, check the label of the indicated volume, and determine the location and extents as necessary. When BOS is booted, it will check the label of the disk on to which it is being booted for a BOS partition, and use it. The BOS WARM/COLD card will indicate the physical location of such a drive. If a special indication is given on this card, BOS will not attempt to read the pack, but put itself on a place indicated by the card, thus overwriting the previous contents of the pack. The following WARM/COLD cards illustrate this.

```
WARM 30. 4 191.       *BOS exists in BOS partition,
                       channel 30, drive 4, D191.
COLD 30. 4 191.       *Same place, but no BOS is there
                       now.
COLD 30. 4 191. 19200. 70. OVERWRITE * Pack on
                       channel 30, drive 4 has no label.
                       Write BOS in records
                       19200-19269. (Long names
                       in BOS is an upcoming feature).
```

Volumes may be repartitioned (partitions redefined or moved around) by an online program, the partition editor. Since a freshly initialized pack is actually an I/O pack, not a storage system pack, no particular access is needed to run such a program. However, the use of this program on a storage system

pack which has been recognized as such is a necessary feature, and a highly privileged process must be allowed to use this program on such a pack. This would involve a call to RCP to temporarily allow the use of the pack as an I/O pack, while maintaining its legitimacy as a storage system volume.

Repartitioning may involve the movement of data. For boundary movement between FDUMP, BOS, and log partitions, this is not a problem. However, if partitions are to be dynamically expandable into the space on a volume in which segments are allocated, pages residing in the area into which the partitions will grow must be relocated. This may be done by the partition editor by walking the VTOC of the volume, and reallocating any pages found in the eviction area. This is neither a difficult nor time-consuming operation. While doing so, the Volume Map can be reconstructed at no extra expense, for it will be changed by this operation anyway.

Repartitioning the RPV will probably be a common case. In this case, it is impossible to dismount the volume and use it as an I/O disk. If partitions are to be grown, reallocation of pages must take place. While this volume is active, these pages may belong to active segments. An extremely complex supervisor primitive could be written to find and relocate such pages, but it is not clear that this is worthwhile. If a site has two drives, a fresh Multics could be created on a spare pack, and used to edit the partitions on the real RPV. A complex BOS or stand-alone utility could also be written for the one-drive case. At any rate, it is not clear that further consideration of the one-drive case is worthwhile at this time.

Note that one cannot be allowed to repartition a pack containing an in-use BOS, because both the partition itself and Multics core contain addresses describing storage within it.

VII. SAVE/RESTOR and the Salvager

MTB 206 proposes a reasonable interpretation of SAVE/RESTOR for the New Storage System. However, in the long run, SAVE and RESTOR must be obsoleted. Although a pack-to-tape-and-back BOS utility is reasonable, BOS must not be the bastion of Backup Reliability for Multics. An online program can dump a pack in some reasonable way, either while it is in use, or after it has been quiesced in some way.

Thus, it is felt that all of proposals outlined in MTB 206, with respect to processing such requests under BOS, are interim. Until online SAVE/RESTOR capability is attained, MTB 206 must be modified to use the Disk Table as a driving table as opposed to the CONFIG deck. If packs are saved via specification of subsystem name and drive number, not even this need be done. However, if requests to save fractions of the online disk complement are to be honored, the Disk Table must be

interrogated. Two schemes, each equally unpleasant in my view, are proposed.

1. The Disk Table can reside in a partition on the RPV. This requires special treatment of its device addresses, not unlike the Syserr Log.

2. The VTOC index of the Disk Table can be stored in a special cell in the RPV label. This cell would be zeroed by bootloading, and set via a highly privileged call. Some fearsome means must be provided to prevent the deletion or invalidation of the segment, and BOS must know how to reconstruct it from its VTOC entry.

There is also a need for the Salvager to know the PVT index to volume transformation. This is because the Salvager must flush the Paging Device, whose map contains PVT indices. However, the Salvager need not read the Disk Table or anything like it. Since the physical-drive to PVT mapping is canonical, it can be reconstructed by the Salvager from the CONFIG deck. The Salvager cannot verify labels or mountings. Thus, it is necessary to define the canonical mapping such that even online drive reconfiguration cannot affect it (i.e., all possible drives must be in the CONFIG deck under some guise). Furthermore, no mounting or dismounting can be performed by the operator between a crash and a salvage. Such mounting and dismounting is as wrong as if it were performed while the system were running.

VIII. Pack Initialization

Pack initialization is the writing of the label, volume map, VTOC, and other storage system data bases on a pack. It is performed in the current NSS by system initialization, in ring zero. This is clearly interim, and wrong. Pack initialization should be performed by a utility program running in the user ring. Such a program has been written, which makes good use of rdisk_, PL/I I/O, and other such niceties available in the user ring. Note that anyone can initialize a pack, or for that matter, write anything they wish on one. The conversion of a pack from I/O to storage system volume is an administrative act, consisting of taking a pack upon which this program was run, registering it with RCP, and moving it from one shelf to another. This shelf-moving operation symbolizes the operator's authority to legitimately mount that pack in response to a storage system mount request.

Adding a physical volume to a logical volume is an RCP operation, as is the special case of recognizing a new physical volume into a logical volume. Such volumes, of course, must have previously been registered and initialized.

A pack initializer running in BOS, with some

partition-creating ability, is necessary, as outlined above, for COLD bootloads. Since it will only be used rarely, it need not be as powerful or general as the online disk initializer.

IX. Double Write Volumes

A need has been felt for the capability to duplicate an on-line volume, i.e., have all writes to one physical volume go to the same address on another. A strategy in page control has been devised to allow this to be done efficiently in parallel. Special use of the Dis- Table is necessary, however, to allow such a feature.

We define a shadow, or double-write volume to be one of a pair of "paired" physical volumes, to which all page control writes are duplicated to parallel addresses. VTOC writes as well must be duplicated. A set of paired volumes is created by creating a shadow volume of an existent physical volume, and registering it with RCP. A volume may be de-shadowed in this way as well. Either RCP may request the mounting of a pack for the initial shadow copy to be made, or RCP may assume that the operator has run an online or offline utility to copy such a volume. An option to the copy program is to mark a pack as a shadow copy. Once a volume has been registered as paired, RCP will request the mounting of both volumes whenever the containing logical volume is requested. Two Disk table entries will be allocated, and specially marked. What is more, the PVT entries of the two volumes will be specially marked. Whenever a segment is activated from a paired volume, a bit is turned on in its AST entry indicating this fact. Whenever Page Control sees this bit at page-write time, special actions will be taken to cause two writes to be queued and properly posted. The page will be considered out of service until both writes are complete. vtoc_man must take similar action upon seeing such a bit in the PVT entry of a volume for which VTOC I/O is requested.

Shadowing the RPV may be accomplished by specifying multiple fields on the ROOT card. Either two volumes were initialized at the time the system was first booted COLD, or an offline utility must be run to copy the RPV onto another pack before the first time the RPV is used shadowed.

X. Scratch Packs

A need has been expressed for the concept of a scratch pack, i.e., a pack onto which supervisor segments, process directories, and salvager temporary segments may be placed. Conflicting with this need is the goal that the new Storage System be bootable on a one-pack system. Also, as proposed here, the system must come up on one pack every time.

The problem of where to put the Salvager temporaries is solved under the next heading below. We concern ourselves

here with the issue of process directory segments.

The general idea of a scratch pack is to have a pack or set of packs whose contents need not be preserved or recovered across a system failure or shutdown. Segments inferior to process directories are the obvious candidates for residency on such packs.

The proposed implementation falls within the new Storage System concept of a Logical Volume. A site may select a given Logical Volume as the "scratch volume". It can contain any number of physical volumes. When this volume is mounted or recognized, which must happen before Answering Service Startup, >process_dir_dir is established as a Master Directory for the scratch volume. The Initializer's process directory is established as a Master Directory for the RLV. This places all process directory segments on the scratch volume.

If a scratch volume's contents need not be considered valid between bootloads, it must be defined to be empty at recognition/mount time. Hence, it is necessary to free all of the VTOC entries of such a volume at the time it is recognized. This should be done by rewriting the entire VTOC, since it cannot be assumed valid. This may take a half minute for 1000 VTOC entries. Segments created on a "scratch volume" must be specially marked in their branches at branch creation time. This flag allows deletion of the segment without referencing its VTOC entry if and only if the volume containing the segment is not mounted. The deletion primitive will be assured that the next time the volume is mounted, the entire VTOC will be cleared, and all space on the volume freed. In order to make certain that this happens, a scratch volume must be marked on its label at the time it is declared to be a scratch volume, or accepted as such, which prevents its mounting as anything but a scratch volume. Each time it is mounted, all of its segments will effectively be deleted. A utility operation to unscratch a volume is quite simple.

The label of a scratch pack will be considered trustworthy, and the partition extents within it reliable. Hence, an FDUMP partition may effectively be defined on such a pack.

XI. The Hardcore Partition

Some recent proposals for the Salvager have proposed it to be implemented within system initialization. Under these proposals, the Salvager would salvage the RLV, or at least the RPV, as the system came up, before even coming up to command level. In order to do this, the RPV volume map must be reconstructed by this Salvager. However, the hardcore supervisor is mostly a set of paged programs, and page control must allocate disk space for these paged programs before this Salvager is even

invoked. Hence, there is a recursion problem in the use of the RPV Volume Map for allocation of supervisor pages.

We propose the following simple solution to the problem of allocating the supervisor reliably. A partition is defined on the RPV, initially by the pack initializer, and editable by the partition editor. This "hardcore partition" is not pointed to explicitly by a PART card, but must exist on the RPV. When page control is brought up, the RPV label is read and the extent and location of the hardcore partition ascertained. The FSDCT map for the RPV is set up as though the RPV hardcore partition were the entire free region of the volume, and marked as entirely free. Hence, all allocations by hardcore will be placed in the hardcore partition. Since it is a partition, and its extents are considered trustworthy, no legitimate data other than supervisor segments of a previous bootload may be found there. Those supervisor segments which were not ultimately placed in the hierarchy no longer exist- their storage or its reuse are not issues. Storage used by supervisor segments from the previous bootload which <u>are</u> in the hierarchy from a previous bootload (e.g., error_table_), however, are another issue, and will be deferred for a moment.

All of the paged segments in collections 1 and 2 will be allocated in the hardcore partition, without fear of trusting a possibly unreliable volume map. When and if any special kind of Salvager has performed its action, or hardcore is ready to acces the directory hierarchy, a magical transformation occurs: At this time, the volume map of the RPV has either been reconstructed or can be believed anyway, so it is read, and redefined to be the legitimate free storage map for the RPV, in the FSDCT. Now, all pages withdrawn, whether they be for directories, supervisor segments not yet paged out, or any other type of segment, are withdrawn against the main portion of the RPV.

The only remaining issue is the deletion of supervisor segments placed in the hierarchy by a previous bootload, and the ultimate deletion of the hardcore segments of the current bootload at shutdown time. The free storage manager, free_store, can sense at the time that these addresses are deposited that they are outside of the legitimate range of addresses for the RPV volume map. The situation can either be ignored, or a check can be made that they are within the hardcore partition, and then

ignored.

XII. Registration Information

RCP needs several data bases to authenticate and validate requests for the mounting of I/O and storage system packs. The following set of data bases is proposed.

A Volume Description Segment (VDS) for every pack known to the system will exist in a dedicated directory. VDS's are created, edited, and destroyed by operator and administrative command. The ACL on each VDS will describe access in the manner in which RCP currently handles such matters. This VDS will contain device type, label, usage, and other active and important information for the pack. Data relating to the identity of the registrant and other inactive information may or may not be kept here. Included in the information in the VDS is the status of the pack as an I/O or storage system pack, and the logical and physical volume ID's if it is the latter. The pack status may only be changed by highly privileged processes.

A Logical Volume Description Segment (LVDS) will exist for each logical volume known to the system, in a dedicated directory. LVDS's are created, destroyed, and edited by operator and administrative command. An LVDS will describe completely a logical volume, specifically, the physical volume ID's and volume names of all of the constituent physical volumes. The pathnames, ASCII and UID, of all master directories for the logical volume will also be given. Access information may be encoded on the LVDS ACL, or given explicitly. Access to create master directories on a given logical volume must be kept as well. The LVDS is used by RCP at the time a mount of a logical volume is requested -- it is the only data base which defines the logical volume, and will be used to insure that all constituent physical volumes are mounted. Physical volumes are added to and subtracted from a logical volume simply by editing the LVDS. Constituent physical volumes of a logical volume do not supply the number or identity of other constituents -- only the LVDS contains this information. A given physical volume, however, does indicate on its label, not only its own Physical Volume ID, but the Logical Volume ID of that logical volume of which it is part. This is consistent with the design principle that each volume describes only what it is, and does not contain information identifying or describing other volumes.

The hardcore quota and usage mechanisms are cognizant of the concept of master directories and logical volumes. Other than this, the only involvment of the hardcore with the logical volume concept is in the segment allocator, which scans the PVT entries of all mounted physical volumes of a given logical volume, to give a segment a home.