Lights Out! Advanced Tape Automation Using VM/ESA

Document Number GG24-4347-00

July 1994

International Technical Support Organization
San Jose Center

Take Note!

Before using this information and the products it supports, be sure to read the general information under “Special Notices” on page ix.

First Edition (July 1994)

This edition applies to VM/ESA 1.2.0 (and later releases) and DFSMS/VM FL 221.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 471 Building 70B
5600 Cottle Road
San Jose, California 95193-0001

Abstract

This document describes the use of the IBM 3494 and 3495 Tape Library Dataservers in three different environments. The three samples can be used for similar projects at your installation, or as a starting point for how you can combine the automation capabilities inherent in VM/ESA and the hardware automation introduced by the IBM Tape Library Dataservers.

The first environment shows a procedure that allows VSE guests of VM/ESA to use the Tape Library Dataservers without the need for native VSE support of the 3494 or 3495. The second environment demonstrates a method of integrating Tape Library Dataserver functions and the services of a host tape management system (TMS). The third environment shows how a large IBM installation developed procedures to perform multisystem and multilevel sharing of tape devices within a Tape Library Dataserver.

This document is written for IBM and customer technical personnel who plan to implement an IBM Tape Library Dataserver in a VM/ESA environment. Some knowledge of the 3494 and 3495, DFSMS/VM Function Level 221, and VM/ESA is assumed.

ST VM (67 pages)

Contents

Figures

Special Notices

This publication is intended to help system support staff install and implement an IBM Tape Library Dataserver in a VM/ESA environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by VM/ESA, DFSMS/VM and the IBM 3494 or 3495 Tape Library Dataservers. See the PUBLICATIONS section of the IBM Programming Announcement for VM/ESA, DFSMS/VM and IBM 3494 or 3495 Tape Library Dataservers for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AD/Cycle
IBM
VSE/ESA
DFSMS/VM
VM/ESA

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

VM:Tape
Sterling Software

Preface

This document is intended to help IBM and customer technical personnel install and use an IBM Tape Library Dataserver in a VM/ESA environment. It contains a description of the support available in the VM/ESA environment as well as three *sample* implementations.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, “Introduction to IBM Tape Library Dataservers”
- Chapter 2, “VSE Guests of VM/ESA”
- Chapter 3, “VM:Tape and IBM′s Tape Library Dataservers”
- Chapter 4, “Multiple VM Systems Sharing a 3495”
- Appendix A, “ VSE Automation”
- Appendix B, “Special Considerations”

Related Publications

The following publications are considered particularly suitable for a more detailed discussion of the topics covered in this document:

- *DFSMS/VM Function Level 221: The Undiscovered Feature*, GG24-4098
- *VM/ESA DFSMS/VM Function Level 221 Installation and Customization*, SC26-4704
- *VM/ESA DFSMS/VM Function Level 221 Planning*, GC35-0121
- *VM/ESA DFSMS/VM Function Level 221 Removable Media Services User′s Guide and Reference*, SC35-0141
- *IBM 3494 Tape Library Dataserver Introduction and Planning Guide*, GA32-0279
- *IBM 3494 Tape Library Dataserver Operator′s Guide*, GA32-0280
- *IBM 3495 Tape Library Dataserver Installation Planning and Migration*, GC35-0135
- *IBM 3495 Tape Library Dataserver Introduction*, GA32-0234
- *IBM 3495 Tape Library Dataserver Operator′s Guide*, GA32-0235

International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

Acknowledgments

The advisor for this project was:

Craig Welch
International Technical Support Organization, San Jose Center

The author of this document is:

Erich Amrehn
IBM Germany

This publication is the result of a residency conducted at the International Technical Support Organization, San Jose Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Kathy Eldred
IBM Tucson

Erik Stangerup
IBM Denmark

Norbert Hoeller
IBM Canada

Steve Olson
IBM San Jose

Scott McIntyre
IBM San Jose

Kris Buelens
IBM Belgium

Catherine Gibson
IBM San Jose

Vo Cao
IBM San Jose

Peter Kasper
IBM Austria

Peter Koekkoek
IBM Netherlands

Wiebe Elzinga
Fokker Aircraft B.V.

Bill Heller
Fokker Aircraft B.V.

Tony Noto (and unnamed others)
Sterling Software

Finally, thanks to Maggie Cutler, whose editorial support made this document readable!

Chapter 1. Introduction to IBM Tape Library Dataservers

Tape provides the lowest cost storage (on a per gigabyte basis) for most types of data. Tape storage is used effectively to hold backed up or archived data, very large files, infrequently referenced data, and other data whose use does not require immediate, online access. While the cost of tape media has declined, operational costs to handle tape media have risen. Automation is often regarded as a way to keep tape handling costs down and improve tape access time.

IBM offers two automated tape library systems, the IBM 3494 and the IBM 3495 Tape Library Dataservers. Although these two products address the needs of different markets (the 3494 addresses the needs of small to large customers, and the 3495, the needs of large and very large customers), they share common design ideas. Both have a robot accessor for mounting and demounting cartridges, specific models of 3490 tape drives, cartridge storage racks, and a library manager.

Figure 1 shows a small IBM 3494 (one drive unit and one cartridge storage unit) and an IBM 3495 Tape Library Dataserver.

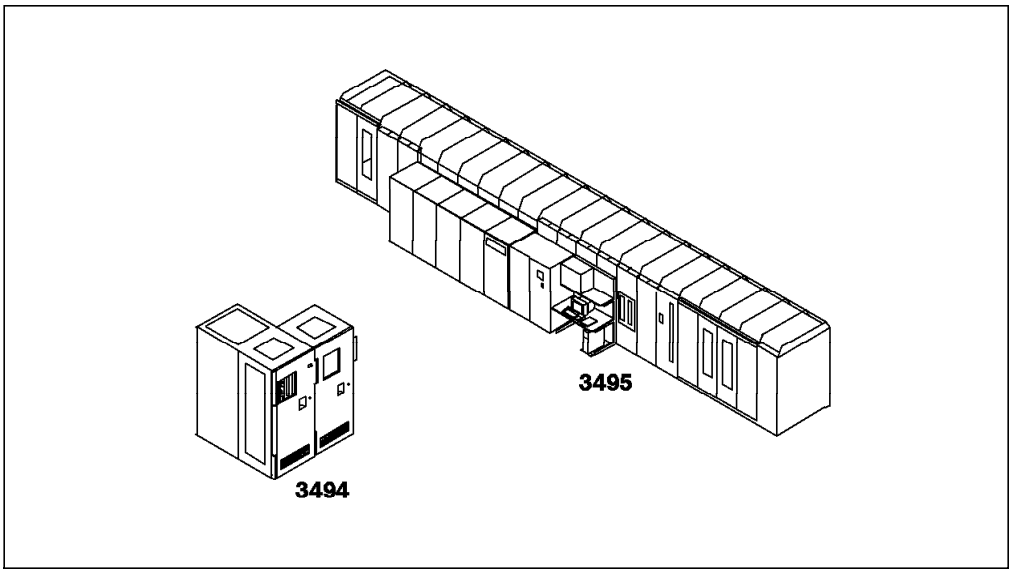

Figure 1. IBM 3494 and IBM 3495 Tape Library Dataservers. Note: Relative sizes are not to scale.

1.1 Terminology

When discussing Tape Library Dataservers, some of the terminology used is the same as that used for manual tapes, and some of it is new or unique. The following list defines the terms we use in the remainder of this book:

Accessor The robotic assembly capable of mounting and demounting tapes in a Tape Library Dataserver

Library manager Hardware and software that directs the operations of the accessor and maintains an inventory of what is stored in the library

Volume	A tape cartridge
Category	A logical grouping of volumes allowing reference to one or a group of tapes by a single name
Mount	The action where the accessor retrieves a volume from a cartridge storage rack and inserts it into a tape drive
Demount	The action where the accessor removes a volume from a tape drive and stores it in a library rack
Insert	The process of adding a volume to the library.
Insert (category)	A transient category used by the library manager to identify tapes that have been added to the library
Eject	The process of removing a tape from the library
Eject (category)	A category used to identify volumes that are to be ejected from the library

1.2 Overview of the IBM 3494 and 3495 Tape Library Dataservers

This section describes some of the physical components of the 3494 and 3495 Tape Library Dataservers and explains how the tape cartridge inventory in both tape libraries is organized.

1.2.1 3494 Tape Library Dataserver: Physical Components

Figure 2 shows a 3494 Tape Library Dataserver. Some of its key components are described below.

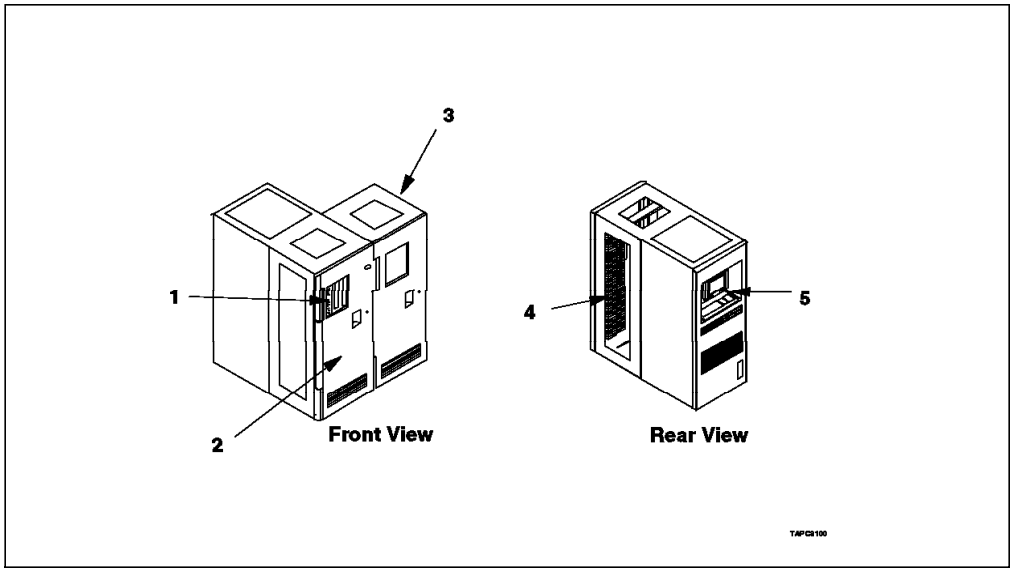

Figure 2. The IBM 3494 Tape Library Dataserver

1 Convenience I/O Station. An area for insertion or ejection of a small number of cartridges. Insertion or ejection of cartridges using this station does not require the accessor to be quiesced.

2 Control unit. A 3494 module that provides cartridge storage, tape drive control unit functions, and 3490-C1A or 3490-C2A tape drives

(and associated electronics).[1]

3 Cartridge storage frame. An expansion frame that adds cartridge storage racks for the 3494.

4 Cartridge storage rack. Used to store cartridges inside the 3494. Part of a cartridge storage rack can be used as the high-capacity I/O facility.[2]

5 Library manager. The control and inventory point for the Tape Library Dataserver. The library manager receives requests from host systems and schedules work for the accessor. The library manager translates requests for "volumes" into instructions for locating a specific cartridge.

1.2.2 3495 Tape Library Dataserver: Physical Components

Figure 3 shows a 3495 Tape Library Dataserver. Some of its components are described below.

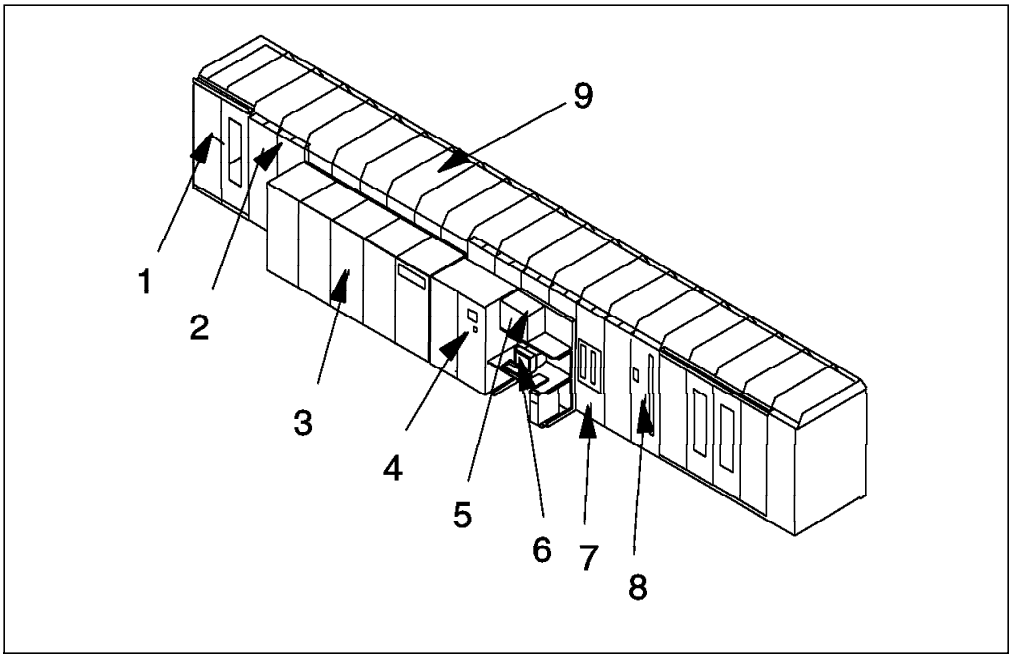

Figure 3. The IBM 3495 Tape Library Dataserver

1 Operator access area. This door allows an operator to enter the library enclosure. The accessor must be quiesced before the door is opened. A system of interlocks ensures that entry is not possible while the accessor is operating.

---

[1] "Drive units" appear similar to the control unit. Drive units have only the 3490-CxA tape drives without the associated control unit functions.

[2] The high-capacity I/O facility is an optional, installation-defined cartridge storage rack used to insert and eject large numbers of cartridges. For insertion of cartridges, an operator quiesces the accessor and places cartridges in the input area of the high-capacity I/O facility. After restarting the library, the accessor examines the inserted tapes and the library manager adds the new cartridges to the library inventory. For ejection of cartridges, the accessor places the tapes in the output area of the high-capacity I/O facility. An operator quiesces the accessor and enters the library to remove the tapes.

2 High-capacity input/output facility. A predefined area of cartridge storage used for removing large numbers of cartridges. For brevity, we call this the high-capacity I/O facility when discussing both the 3494 and 3495.

Note: IBM recently introduced an *active high-capacity I/O facility*. The active high-capacity I/O facility allows installations to add and remove up to 240 cartridges at a time *without* interrupting normal access operation.

3 3490/3490E tape subsystem. The tape devices used by the 3495 Tape Library Dataserver.

4 Cartridge accessor controller. The electronics that control the operation of the accessor.

5 Manual mode terminal. A terminal located inside the library enclosure. An operator uses this terminal to locate tapes to be mounted when the accessor is not operational.

6 Library manager. The control and inventory point for the Tape Library Dataserver. The library manager receives requests from host systems and schedules work for the accessor. The library manager translates requests for "volumes" into instructions for locating a specific cartridge.

7 Convenience I/O Station. An area for insertion or ejection of a small number of cartridges. Insertion or ejection of cartridges using this station does not require the accessor to be quiesced.

8 Service area. The entry area for servicing the accessor.

9 Cartridge storage frames. Special racks on the walls of the library enclosure used to store 3490 tape cartridges.

1.2.3 Volume Organization

The volumes in a 3494 or 3495 are organized into categories. Categories facilitate physical management of the volumes by efficiently defining (at a high level) how the cartridges are to be used. VM/ESA recognizes three categories:

- Operating categories (INSERT, EJECT, EJECTB)
- Scratch categories (SCRATCH0-SCRATCHF)
- Volume-specific category (volspecific).

The operating categories are used for entering volumes into and removing them from the Tape Library Dataserver. When a tape is inserted into an input station and recognized by the 3494 or 3495, it is placed in the INSERT category for later assignment to another category. When a tape is to be removed from the tape library, it must be assigned to either the EJECT or EJECTB category. EJECT instructs the accessor to place the volume in the Convenience I/O Station; EJECTB instructs the accessor to place the volume in the high-capacity I/O facility.[3]

Scratch categories have several uses. The most familiar use of a scratch category is to request "any" tape for either temporary output or for use and

[3] The 3494 offers a Mount from Input Station capability that allows a cartridge to be inserted in the Convenience I/O Station, picked up by the accessor, mounted, used, and then ejected from the 3494. The volume never becomes part of the 3494 inventory and is not checked for an external label.

assignment to another category. Sixteen scratch categories are offered so that different applications can use different pools.

The volume-specific category is used for volumes you want to be mounted by requesting a specific volume label; typically such volumes already have user data stored on them or have been designated for use by specific users.

1.2.4 Comparison of the IBM 3494 and IBM 3495

Table 1 compares some selected features of the IBM 3494 and 3495 Tape Library Dataservers.

Table 1. IBM 3494 and 3495 Tape Library Dataservers: Selected Features

	IBM 3494	**IBM 3495**
Supported by VM/ESA and DFSMS/VM	Yes	Yes
Volume capacity	210 to 3000	5600 to 18900
3490 models	3490-C1A and 3490-C2A	3490-A01, 3490-A02, 3490-A10, 3490-A20, 3490-B04, 3490-B20, 3490-B40
Number of 3490s	8	64
Convenience I/O Station	Yes	Yes
High-capacity I/O facility	Yes	Yes
Active high-capacity I/O facility	No	Yes
Any free cell can be used for insert (requires library to be paused)	Yes	No
Mount from Input Station	Yes	No

Note:

1. Volume capacity is affected by several options. 3490 drive units and use of the high-capacity I/O facility reduce volume storage capacity but do offer additional function or performance. See the appropriate 3494 or 3495 planning guide for exact details.
2. You will have to order the appropriate Library Attachment Feature for each 3490 you want to attach to the Tape Library Dataserver.
3. You can upgrade the 3490-C12 and 3490-C22 drives to 3490-C1A and 3490-C2A drives.

1.3 Tape Library Dataserver Operation with VM/ESA

Support for the IBM Tape Library Dataservers was introduced with VM/ESA 1.2.0 and DFSMS/VM Function Level 221. DFSMS/VM delivered the support using a new service virtual machine (default name RMSMASTR). RMSMASTR provides a default set of functions for mounting tapes, managing volume inventory, and other tape library functions. RMSMASTR is designed to be incorporated into an installation's existing tape handling procedures by working with tape management systems or installation-written EXECs.

The current implementations of the 3494 and 3495 require that library commands be passed from the host to the library manager by means of a 3490 tape drive and the Library Attachment Feature. In practice, any Tape Library Dataserver function on VM requires the following sequence of steps (as shown in Figure 4 on page 6):

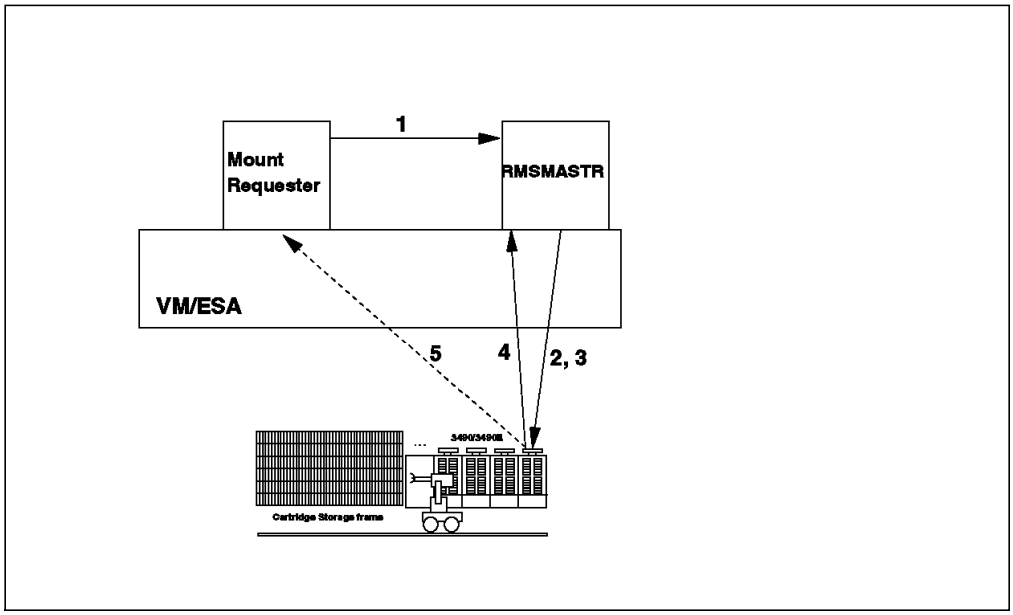

Figure 4. Basic Control Flow for a Tape Request

1. The removable media services virtual machine (default name is RMSMASTR) receives the request for a library function.
2. RMSMASTR uses the 3490 device specified on the request or attempts to find an available device if one was not specified.
3. If the specified device is available (or if one is free for a non-device-specific request), the device is attached to RMSMASTR and the command is issued. The library command is passed from the tape drive to the Tape Library Dataserver using the Library Attachment Feature.
4. Status is returned to RMSMASTR when the command is complete.
5. The device is given to the function requester (on a mount request) or detached from RMSMASTR (on other requests, such as queries).

Notes:

- If a specific device is requested and that device is not available, the request fails.
- If no specific device is requested and no available device can be found, the request fails.

1.4 Understanding Removable Media Services

This section discusses the structure of the removable media services virtual machine (RMSMASTR). Note that the discussion in this section is less specific than a similar discussion in Chapter 4, “Multiple VM Systems Sharing a 3495” on page 39. This section is based on Chapter 4 of *DFSMS/VM Function Level 221: The Undiscovered Feature*, GG24-4098.

1.4.1 Overview of Removable Media Services

Removable media services (RMS) is provided by DFSMS/VM Function Level 221 to support the IBM 3494 and IBM 3495 Tape Library Dataservers in a VM/ESA Release 2 environment.[4] The support is delivered as part of DFSMS/VM Function Level 221. Figure 5 shows the three major components of DFSMS/VM Function Level 221.

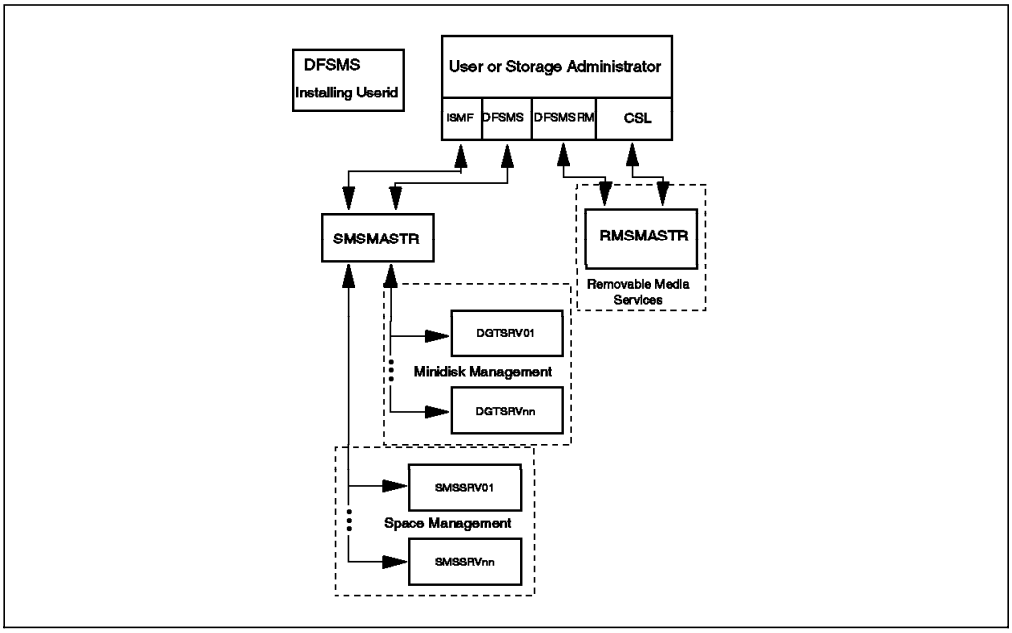

Figure 5. DFSMS/VM Function Level 221 Components

Figure 5 shows several virtual machines. Dashed lines surround three functional "groups" of virtual machines. The three groups are removable media services, minidisk management, and space management. Minidisk management offers installations the ability to automate minidisk relocation using service virtual machines. The service virtual machines can move minidisks between like and unlike device types and will normally use the high-speed data mover (provided by DFSMS/VM) to complete the minidisk relocation process faster. Space management offers installations the ability to automatically migrate and expire files from Shared File System (SFS) storage. Migration and expiration is controlled and defined by the installation's storage administrator.

Space management and minidisk management are covered in *DFSMS/VM Function Level 221: The Undiscovered Feature* and will not be discussed further in this book.

The RMS support code runs in the RMSMASTR virtual machine. Access to the 3494 or 3495 is provided by an interface that includes both RMS commands for interactive control and Callable Services Library (CSL) routines for program control.

RMS support does not include TMS services such as maintaining a removable-media inventory, performing tape-label verification, performing authorization access checks at the volume level, or managing and selecting tape

[4] RMS also supports manual tape devices, such as the IBM 3490, 3490E, and 3490C. This support is limited to forwarding mount requests to an operator on a device basis.

drives. RMS support is designed to interface with a TMS. For systems without a TMS, TMS-like functions can be added by means of tailoring installationwide exits.

A single RMS master can support multiple 3494s or 3495s. However, if you want multiple VM hosts to share a single 3494 or 3495, each VM host must have its own RMS master virtual machine. For operating systems that do not support the 3494 or 3495, but do support 3490s and have TCP/IP socket communication capability, DFSMS/VM provides foreign host support for mounting and demounting volumes from the 3494 or 3495 library pool on behalf of the foreign system.

1.4.2 DFSMS/VM RMS Command Overview

DFSMS/VM provides a set of commands for interacting with the 3494 or 3495 Tape Library Dataserver. Access to these commands can be controlled with an external security manager, such as RACF, or via an authorization file.

The commands discussed in this chapter are:

DFSMSRM MOUNT	For requesting tape mounts
DFSMSRM QUERY	For investigating the status of tape drives and inventory status
DFSMSRM SET VOLCAT	For assigning volumes to categories
DFSMSRM SET DEVCAT	For mounting a group of volumes on a device

See the *VM/ESA DFSMS/VM Function Level 221 Removable Media User's Guide and Reference* for more information on these and other DFSMSRM commands.

1.4.3 Hardware Requirements

RMS supports all models of the 3495: L20, L30, L40, and L50. The 3490 models A01, A02, and B04 and 3490E models A10, A20, and B40 can be attached to the 3495 and are supported by RMS. RMS also supports the 3494 model L10 and the appropriate 3490E models C1A and C2A.

1.4.4 Software Requirements

The following IBM software must be installed to use RMS:

- DFSMS/VM Function Level 221
- VM/ESA Release 2.0 (native support for the 3495)
- SFS (RMS control files are located in an SFS directory).

For Foreign Host Support both TCP/IP Version 2.2 and Language Environment/370 (LE/370) must be installed on your system.

1.5 3494 and 3495 Tape Library Dataservers and RMS

This section describes how the 3494 and the 3495 are organized and explains how to use RMS to control them.

1.5.1 3494 and 3495 Tape Library Dataserver Operation

The basic functions of the 3494 and 3495 include getting volumes into the library, mounting volumes, managing the inventory of volumes, and removing volumes from the library.

1.5.1.1 Getting Volumes into a 3494 or 3495

In the physical sense, there are two ways to load tape volumes in a 3494 or 3495: through the high-capacity I/O facility and through the Convenience I/O Station.

New volumes inserted into the library are automatically assigned to the INSERT category. Once inserted into the library, these volumes must be assigned to a scratch or volume-specific category. This can be accomplished in either of two ways:

- Manually assigning volumes to categories by invoking an RMS interface function
- Automatically assigning volumes to categories using automatic bulk processing, which is initiated without human intervention when volumes are detected in the INSERT category.

Manually Assigning Volumes to Categories: To manually assign a single volume to a new category, you can issue the DFSMSRM SET VOLCAT VOL *volser targetcat* command, where *volser* is the volume label and *targetcat* is the target category you want to assign to the volume.

To assign a group of inserted volumes to a new category, use an “on-request bulk processing file.” An on-request bulk processing file is a simple text file that lists the volume label (or range of volume labels), the category to which the volume is to be assigned, and an optional source category. The name of the on-request bulk processing file is specified as a parameter of the DFSMSRM SET VOLCAT BULK command. Figure 6 shows a sample on-request bulk processing file.

```
* vol_label     target    source
*
  002001-003500  SCRATCH2  INSERT
  VM0005         SCRATCHA
  VM0011         SCRATCHA
```

Figure 6. Sample On-Request Bulk Processing File

The sample in Figure 6 indicates that volumes with a label between 002001 and 003500 will be assigned to the SCRATCH2 category, and volumes VM0005 and VM0011 will be assigned to the SCRATCHA category.

Any inserted volume that has a label that matches a specific volser or range of volsers specified in the on-request bulk processing file will have a new target category defined. Specifying the source category is optional; if it is used it must be correct, any mismatch will leave the volume assigned to the current category.

Note: Those labels that do not match remain assigned to the INSERT category. RMSMASTR will periodically try to reassign these tapes to a scratch or volspecific category.

Automatically Assigning Volumes to Categories: If you want to have an inserted volume's category changed automatically, you should define an "automatic-insert bulk processing file." All volumes that match the criteria specified in the automatic-insert bulk processing file (with a required file name of RMBxxxxx DATA in VMSYS:DFSMS.CONTROL, where xxxxx is the sequence number of the 3494 or 3495) are automatically assigned a new category. Figure 7 shows a sample automatic-insert bulk processing file.

```
* vol_label     target    source
*
  VM1000-VM1999  SCRATCH2  INSERT
  VM2000-VM2999  SCRATCHA  INSERT
```

Figure 7. Sample Automatic-Insert Bulk Processing File

The sample in Figure 7 specifies that volumes inserted in the 3494 or 3495 with a volume label between VM1000 and VM1999 will automatically be assigned to the SCRATCH2 category, and volumes inserted with a volume label between VM2000 and VM2999 will automatically be assigned to the SCRATCHA category. Specifying the source category is optional; if it is used it must be correct, any mismatch will leave the volume assigned to the current category.

1.5.1.2 Mounting Volumes

The DFSMSRM MOUNT command is used to request a volume mount. This is true whether a mount request includes a specific volume label or a scratch category name.

The DFSMSRM SET DEVCAT command is used to assign categories to a device. This preassignment causes the integrated cartridge loader on the device to be filled with volumes in that category so at mount time the cartridge loader is indexed without accessor movement.

Using DFSMSRM MOUNT: The DFSMSRM MOUNT command allows authorized users to specify parameters for volume mounting. Users can specify many parameters, including those for mounting:

- A scratch tape
- A specific tape
- A tape on a specific drive
- A tape with read or write intent.

A scratch tape could also be mounted using the volspecific category instead of the scratch category.

In 1.5.1, "3494 and 3495 Tape Library Dataserver Operation" on page 9 we describe the 3494 and 3495 volume categories. Once a volume has been assigned to the volspecific category, its volume label must be specified when attempting to mount it.

Using DFSMSRM SET DEVCAT: The SET DEVCAT command allows you to fill the loader of a 3490/3490E unit with six cartridges from the specified scratch category. This improves the mount performance of your 3495.

Note: Only 3490E tape units used with the 3495 have an integrated cartridge loader.[5]

The two commands in Figure 8 show how to associate the SCRATCH2 category with real devices 683 and 688.

```
DFSMSRM SET DEVCAT CAT SCRATCH2 (RDEV 683
DFSMSRM SET DEVCAT CAT SCRATCH2 (RDEV 688
```

Figure 8. Sample DFSMSRM SET DEVCAT Commands

AUTOFILL, the default option, causes RMS to fill up the 3490 cartridge loader with volumes from the SCRATCH2 category.

Figure 9 shows a complex set of category assignments and some typical commands and actions.

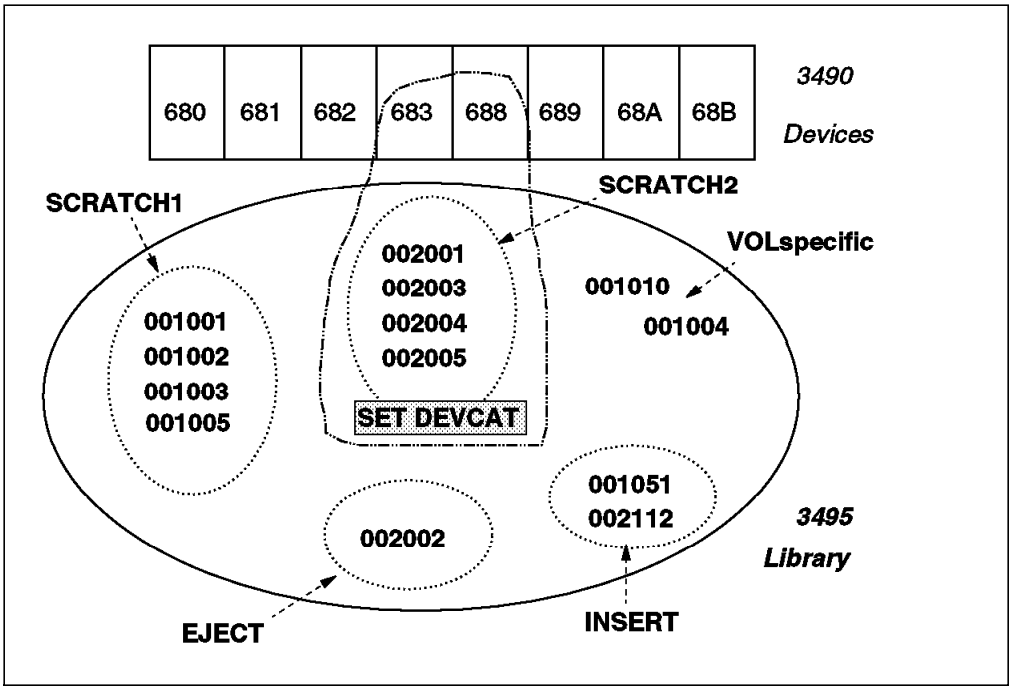

Figure 9. RMS Categories. Assigning a scratch category to 3490 devices using the SET DEVCAT command.

In our example we have two scratch categories: SCRATCH1 and SCRATCH2. Any of the volumes in these two categories can be specifically selected for a mount request without a scratch category specified. Category SCRATCH2 has been assigned to real devices 683 and 688 using the DFSMSRM SET DEVCAT command.

Also shown in Figure 9 are three other volumes. Volume 002002 belongs to the EJECT category and will be ejected from the tape library. Volume 001010 and 001004 are not in any category, that is, they are volspecific. You can assign a

[5] The 3494 allows emulation of a cartridge loader by assigning a specified category of cartridges to a specific drive for the automatic mounting and demounting of cartridges without host intervention. However, currently DFSMS/VM Function Level 221 does not support the *automatic* loading of cartridges. It does allow association of a category and a device.

volume to the volspecific category using either the SET VOLCAT command or a parameter of the MOUNT command. Figure 10 on page 12 shows the two commands that can reassign a volume to a volspecific category.

```
DFSMSRM MOUNT CAT SCRATCH1 (TARGET VOL READW
< OR >
DFSMSRM SET VOLCAT VOL 001010 TARGET VOL
```

Figure 10. Sample DFSMSRM Commands to Reassign a Volume to Volspecific Category

Finally, notice in Figure 9 on page 11 that two volumes are assigned to the INSERT category. They must be reassigned to a scratch category before they can be mounted for a user.

1.5.1.3 Demounting Volumes

DFSMS/VM and the IBM 3494 and 3495 Tape Library Dataservers handle demounting a volume automatically. When users request a mount on a drive on which they already have mounted a different volume, an implicit demount is scheduled before the mount. When users detach a tape drive (returning it to the pool of available drives), DFSMS/VM and the 3494 and 3495 will demount the volume automatically.

DFSMS/VM does provide the DFSMSRM DEMOUNT command for explicitly demounting a volume.

1.5.1.4 Removing Volumes from the 3494 or 3495

Volumes assigned to one of the two EJECT categories will be automatically ejected from the 3494 or 3495. The EJECT category requests that specified volumes be placed in the Convenience I/O Station; EJECTB specifies that volumes be placed in the high-capacity I/O facility. You can assign volumes to the EJECT or EJECTB categories by using the DFSMSRM SET VOLCAT command.

Chapter 2. VSE Guests of VM/ESA

This chapter shows a method that enables VSE systems without native 3494 or 3495 support to request tape mounts when running as guests of VM/ESA. It uses removable media support offered by DFSMS/VM Function Level 221 and a VSE console monitoring package that can execute action routines when predefined messages appear on the VSE console and in the VSE hard copy file (HCF). The VSE console monitoring software is part of an automation service offering from IBM-Denmark. See Appendix A, " VSE Automation" on page 61 for an overview of this offering.

Special Terminology Note

In the discussion that follows, the terms *VSE guest* and *VSE Guest Server* are used. *VSE guest* refers to a virtual machine running the VSE operating system. This is the traditional and familiar use of the term.

The *VSE Guest Server* is a virtual machine that provides tape mounting services for the VSE guest. The VSE Guest Server runs CMS, not VSE, and handles only VSE guest tape mount requests.

2.1 VSE Guest Tape Mount Operation

Figure 11 shows the virtual machines involved in automating VSE guest tape mount requests. The function of each virtual machine is described in 2.2, "Virtual Machines for VSE Mount Automation" on page 14.

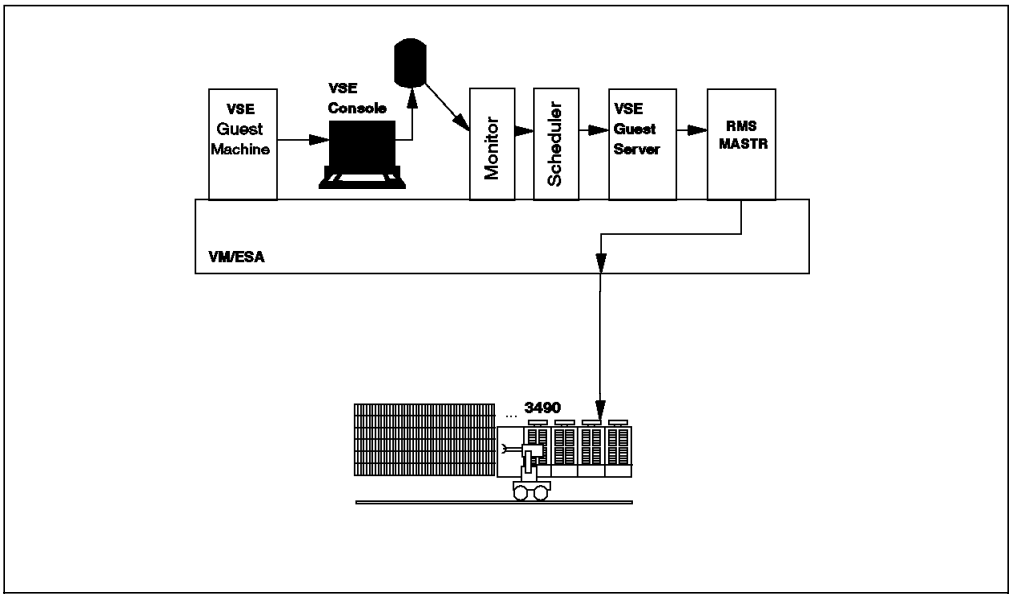

Figure 11. The VM/VSE Implementation

A mount request for a VSE guest virtual machine works as follows:

1. Users or programs running on the VSE guest issue a mount request that appears on the VSE console. Simultaneously, the VSE HCF is updated.

2. The Monitor virtual machine detects that the HCF has been updated, extracts the console message, and passes the message to the Scheduler virtual machine.
3. The Scheduler matches the console message with its table of "action messages." Each message in the table has an associated action to be performed.
4. In this case, the MOUNT message causes an EXEC to be run. This EXEC parses the mount message and sends the pertinent mount parameters to the VSE Guest Server virtual machine.
5. The VSE Guest Server virtual machine receives the mount parameters and passes a properly formatted DFSMSRM MOUNT command to RMSMASTR. The DFSMSRM MOUNT command has parameters that allow the VSE Guest Server to specify the volume to be mounted and the virtual machine to receive the tape when mounted (in this case, the VSE guest).
6. RMSMASTR mounts a tape at an appropriate address and gives the tape drive to the VSE guest.

Note: You may think that an extra virtual machine is involved because the function of the Scheduler and VSE Guest Server seem to overlap. This is not the case, however; a single Scheduler can take actions for multiple Monitor virtual machines, with each Monitor watching the console of a separate VSE guest system. By passing the mount request to the VSE Guest Server, the Scheduler is free to handle new messages from associated Monitor virtual machines while the VSE Guest Server deals with the somewhat unpredictable timing of the tape mount request.

2.2 Virtual Machines for VSE Mount Automation

This section provides more detail on each of the virtual machines involved in VSE guest tape automation.

2.2.1 Monitor Virtual Machine

The Monitor virtual machine continuously scans the VSE HCF for any new messages. Scanning is enabled by linking the minidisk[6] on which the (VSE) HCF resides and then reading certain parts of it. Software running in the Monitor virtual machine knows where to look in the HCF and keeps a pointer to the newest records in it. When new messages (records) are detected, they are written to a CMS-file-format copy of the VSE HCF and transmitted to the scheduler userid.

Figure 12 on page 15 shows a sample VSE guest console with a "generic" mount message displayed.

[6] The minidisk must be on count-key data (CKD) DASD (3380, 3390, 9340).

```
 CMS/VSE Console Interface  DOSESA   - FKOSJJ63  Wednesday 20 Apr 1994 14:35:42
 Display scope : 6167 to 6266 of 6266 ,  Current: 6198          REALTIME
AR 015    9 * FA V         256K     256K   680000
AR 015    9 * FB V         256K     256K   700000   17920K
AR 015    S   SVA-31       556K    1492K  1900000
AR 015        DYN-PA         0K
AR 015        DSPACE     13440K
AR 015        SYSTEM      1024K
AR 015        AVAIL      22336K
AR 015        TOTAL     102400K   <----'
AR 015 1I40I  READY
F1 001 1Q34I  RDR WAITING FOR WORK ON 00C
F1 001 1Q47I   BG PROD3589 03699 FROM LOCAL , TIME=14:35:35
BG 000 // JOB PROD3589   DATE 04/20/94,CLOCK 14/35/35
BG 000 * ------------------------------------------------------------
BG 000 * Jobname: PROD3589    Last revision: 06/11/93    Owner: DJ
BG 000 *
BG 000 * Backup PROD.CATALOG to tape.
BG 000 *
BG 000 * ------------------------------------------------------------
BG 000 *
BG 000  XXXT004A_MOUNT_VOLUME_CS0050_TESTDATA_SYS010_...
====>
PF1=Help 2=Refresh 3=Exit 4=Auto 5=Setup 6=Jobs 7=Bwd 8=Fwd 9=Retrieve 10=Cmds
```

Figure 12. Sample VSE Scheduler Console before Mount Message

The message XXXT004A (or any message you choose to detect) is all the Monitor virtual machine needs to detect in order to start the process of getting a tape mounted for this VSE guest.

Special Considerations

The Monitor needs a link to the HCF. This link must match the VSE guest's minidisk configuration.

The Monitor virtual machine is part of the VSE Automation package.

2.2.2 Scheduler Virtual Machine

The Scheduler virtual machine performs a function similar to the function provided by VM's Programmable Operator facility (nicknamed PROP). The Scheduler scans all VSE console messages sent by the associated monitors and then takes appropriate action. A table defines the messages to act on and which actions to take. When the Scheduler finds a message to act on, it takes one of the following actions:

- SUBMIT - submit a job (to VSE)
- COMMAND - send a command to the VSE console
- EXEC - run an EXEC (on VM)
- SMSG - send an SMSG to another CMS userid (on VM).

Note: The fact that the Scheduler acts on messages make the implementation very flexible. You can tailor the table to define normal mount messages as well as mount messages coming from any tape management system.

When automating tape mounts for VSE guests, the Scheduler runs an EXEC that communicates with the VSE Guest Server (described in greater detail in 2.2.3, “VSE Guest Server” on page 16). The EXEC run by the Scheduler initiates an APPC connection with the VSE Guest Server (the VSE Guest Server is ready to accept APPC connections after it has initialized). For each library request, the EXEC opens, uses, and then severs a discrete APPC session.

The Scheduler virtual machine is part of the VSE Automation package available from IBM Denmark. See Appendix A, “ VSE Automation” on page 61 for more information.

2.2.3 VSE Guest Server

The VSE Guest Server machine functions as an interface between the Scheduler machine and RMSMASTR. The VSE Guest Server supports mounting volumes on drives, querying volumes in libraries, releasing drives, canceling mounts, and ejecting volumes from the library. The VSE Guest Server machine receives parameters from the Scheduler and requests RMS functions by means of the RMS CSL routine interface. In addition, the VSE Guest Server ensures that device attachment requirements of RMS will be met, attaching the device to itself before issuing the RMS CSL call for library operations if it is currently attached to the VSE guest machine and not FREE. The code running on the VSE Guest Server machine is written in REXX. The VSE Guest Server uses CPI-C to communicate with RMSMASTR. The functions supported by the VSE Guest Server at this time are mounting a specified volume or a member of a scratch pool, querying that a specified volser is located in an automated library, and ejecting a volume.

The VSE Guest Server virtual machine has a few specific directory requirements. Figure 13 shows a sample directory for the VSE Guest Server.

```
USER VGLIBSRV XXXXXXXX 32M 64M BG
ACCOUNT   SYS      VGLIBSRV
IPL CMS
IUCV ALLOW
* Star IDENT required only if VGS customization resource type is LOCAL
IUCV *IDENT RESANY LOCAL
CONSOLE 01F 3215
SPOOL 00C 2540 READER B
SPOOL 00D 2540 PUNCH B
SPOOL 00E 1403 A
LINK       MAINT      190 190 RR
LINK       MAINT      19D 19D RR
LINK       MAINT      19E 19E RR
LINK       DFSMS      1B5 192 RR
MDISK 0191 3390 xxxx  15  volser  MR
SETOPTN    SETTINGS :
  LOGGING OFF
  RCVMSGS ON
  LINKS   ENABLE
```

Figure 13. VSE Guest Server Directory Entry

The directory entry establishes the IUCV connect authority (IUCV ALLOW), the privilege classes necessary for tape commands (privilege class B unless

restructured by the installation), and links to the DFSMS code disk (LINK DFSMS 1B5 192).

The VSE Guest Server became generally available from IBM in June 1994 via APAR VM58436.

2.2.4 RMSMASTR

RMSMASTR is the virtual machine that communicates with the Tape Library Dataserver. It requests all library functions (mount, query, and so on) on behalf of authorized users. In this environment, RMSMASTR accepts tape mount requests from the VSE Guest Server.

Special Consideration

The VSE Guest Server must be authorized to issue RMS commands.

The RMSMASTR virtual machine and the software that it runs are delivered by DFSMS/VM Function Level 221 and are a no-charge feature of VM/ESA.

2.3 Hardware Used

The following hardware was used to accomplish the automation of VSE guest tape mounts on VM/ESA:

- An IBM 3494 Tape Library Dataserver (note that an IBM 3495 will work as it uses the same support offered by DFSMS/VM, however the 3495 was not used)
- A 3490-C1A or 3490-C2A attached to an IBM 3494
- A processor capable of supporting both VM/ESA Release 2.1 and the 3490 tape drive(s) as specified above
- A CKD minidisk for the VSE HCF.

2.4 Software Requirements

The following software was used to implement the automation of VSE guest tape mount requests:

- VM/ESA Release 2.1 (check for all available DFSMS/VM corrective service)
- A virtual machine running VSE (running as a second-level guest of VM/ESA).

The software used for implementing the Monitor, Scheduler, and VSE Guest Server machines is based on REXX, XEDIT, CMS Pipelines, the CMS Utilities Feature, and a few assembler programs.

2.5 The VSE Tape Library Dataserver Application Programming Interface

When IBM announced the 3494 Tape Library Dataserver it also stated its intent to provide an application programming interface (API) so that VSE guests of VM could request tape mounts. The following is a discussion of how to choose between the API and the VSE Automation package.

The API will allow VSE guests to directly interact with the VSE Guest Server. While the API may appear to eliminate the need for the VSE Automation package, it does not. The following considerations apply when deciding if you should use the API or the VSE Automation package:

- **If you have a tape management system (TMS) on your VSE guest** and that TMS has customizable mount exits, you will want to use the API.
- **If you do not have a TMS and do not want one** you will want the VSE Automation package.
- **If you want to automate job scheduling and other operator functions (in addition to automating tape mounts)** you will want the VSE Automation package.
- **If you want to extend your tape mount handling "intelligence"** the VSE Automation package provides access to simple and familiar CMS tools and techniques.

In summary, to use the API you must remember that something (either a TMS or your own programming logic) is needed to drive the API. Installations with a well-supported TMS and no need for additional automation potential will welcome the introduction of the API. The VSE Automation package provides more function and will be easier to use for most installations.

2.6 Turning Out the Lights for VSE Guests of VM

Review the material in Appendix A, " VSE Automation" on page 61. It gives a brief description of some of its capabilities and describes how to get the VSE Automation package. Also, obtain APAR VM58436 from IBM for the VSE Guest Server.

Chapter 3. VM:Tape and IBM′s Tape Library Dataservers

This section describes an effective way to get an existing tape management system (TMS) to work with an IBM Tape Library Dataserver. For the work described here, VM:Tape** was used as the TMS, and an IBM 3495 Tape Library Dataserver was used as the tape library. An extra virtual machine running the Programmable Operator facility was used to capture mount messages sent to an operator.

News Flash!

The version of VM:Tape used for our work was not the most current version. VM:Tape has been greatly enhanced with new exits that will make some of the work described here easier, and will enable other functions not possible with the version we used. The most recent version of VM:Tape has enhancements to allow tape drive pooling, tape drive allocation (an exit), and mounting from a drive pool. Sterling Software recommends that you use VM:Tape 1.1 or higher, and adapt the procedures described here to the newest version of VM:Tape.

Please Note!

The code samples shown in this chapter worked at a specific installation to solve a specific problem. While the technique used should be helpful when working in similar environments, the samples may have to be tailored for your installation.

3.1 Mount Processing with VM:Tape

In a manual tape environment, VM:Tape sends tape mount messages to a specified userid. The person responsible for acting on messages sent to that userid responds by mounting a tape. In an automated environment, VM:Tape still sends messages to a specified virtual machine, but that machine is not being watched by an operator; it is being “watched” by PROP. So as not to interfere with any normal messages on which operators still need to act (both from VM:Tape and normal system operation), PROP is run in an extra virtual machine. The extra virtual machine monitors the tape operator′s console for predefined messages. When one is detected, PROP executes specified actions.

For our work, we defined a set of messages to watch for, and a set of actions to take when the messages were trapped. For a VM:Tape mount message, PROP executes an EXEC that, in turn, executes a DFSMSRM MOUNT command (after parsing the original message to get the necessary parameters) on behalf of VM:Tape. When the mount is completed, control of the tape drive is passed to VM:Tape. VM:Tape then continues with processing the mount request on behalf of the requestor.

Figure 14 on page 20 shows the real and virtual machines present when the work was performed to allow VM:Tape to mount tapes using an IBM Tape Library Dataserver. The MVS/ESA system is shown simply for completeness.

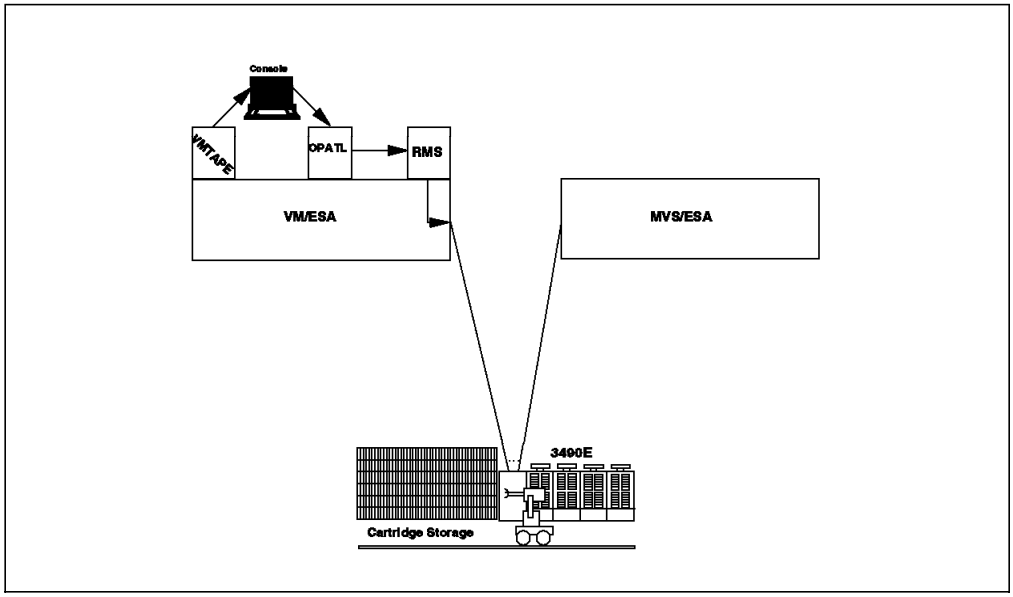

Figure 14. Automated Mount Processing with VM:Tape

The IBM 3495 Tape Library Dataserver is shared between a processor running VM/ESA 2.1 and a processor running MVS/ESA. This installation controlled tape sharing by relying on VM:Tape and MVS. To ensure that the VM system did not try to mount any tapes beloning to other systems, the installation specifically defined three ranges of volumes (one for MVS, one for VM, one to be shared by both). The Automatic Bulk Insert file and MVS exits controlled assignment of newly inserted volumes. (MVS handled its own and the shared volumes, VM handled its own volumes.)

On the VM side, three virtual machines are involved: VMTAPE, OPATL, and RMS (changed from the default RMSMASTR). VMTAPE (the virtual machine running VM:Tape) functions without any changes (with the exception of a few changes to its control file, see 3.1.1, “VMTAPE CONFIG File” on page 21). OPATL runs PROP. The OPATL virtual machine uses a routing table to intercept MOUNT requests from VMTAPE and converts these requests to DFSMSRM MOUNT requests for RMSMASTR. The RMSMASTR virtual machine provides the removable media support for IBM′s Tape Library Dataservers.

The following list describes the events that occur when mounting a volume using VM:Tape and DFSMS/VM′s removable media support.

1. A user requests a volume using existing VM:Tape request procedures.
2. The VMTAPE virtual machine issues a mount request. This request is normally sent to a specified user or operator; the mount message is sent to OPATL.

   The mount message is defined in the PROP routing table (RTABLE) and triggers OPATL to execute the MOUNTATL EXEC. The MOUNTATL EXEC performs the following tasks:

   a. Scans a “tape contents data base” (TCBD) to verify the volume is in the library
   b. Detaches the tape unit from VMTAPE (save device address)
   c. Mounts the requested volume using DFSMSRM commands

d. Returns (attach) the tape unit to VMTAPE using the saved device address.

3. The return of the tape to VMTAPE causes an interrupt. VMTAPE interprets this interrupt as tape mount notification. After the interrupt, VMTAPE proceeds as if there were no tape library. This processing includes label verification, updating the tape management catalog, and passing control of the tape unit to the requester.

To automate the mounting of tapes in an automated environment with VM:Tape, two files must be modified. First, the VMTAPE CONFIG file must be modified. Then the PROP routing table must be set up to recognize VM:Tape mount messages. When the VMTAPE CONFIG file and the PROP routing table have been modified, the installation must define the action routines (in this case the action routines are EXECs).

3.1.1 VMTAPE CONFIG File

The default VM:Tape configuration repeats a mount request every minute on the assumption that a tape operator will manually mount the tape. With a 3495 a tape mount may take less than 1 minute to more than 2 minutes (depending on the presence of other requests).[7] It is necessary to increase the “message repeat” timer to 5 minutes to avoid repeating mount messages (programming logic is much more complex if repeat requests have to be handled). The 5 minute “message repeat” setting is also sufficient for normal manual tape operator mounts. When VMTAPE receives an interrupt, it verifies the internal label and attaches the tape drive to the user who has requested this tape.

VMTAPE checks the status of its tape drives at regular intervals. If the tape unit is not found (that is, it has been detached), VMTAPE fails the requested tape mount operation. However, RMSMASTR detaches the tape from VMTAPE to perform the requested tape operation. Thus, if VMTAPE checks its drive status while RMSMASTR has the tape, the mount request will fail. To avoid this, the requested mount should either succeed or fail before the interval for checking the tape drive status expires. Therefore, we set the interval timer to 10 minutes, which should allow RMSMASTR and the 3495 sufficient time to get a tape mounted.[8]

Setting the message repeat time to 10 minutes worked in our environment when the 3495 was in manual mode (for manual operator actions such as removing cartridges from the bulk I/O station).

VM:Tape allows users to request scratch cartridges using a pool name. VM:Tape will request that a tape operator mount any available scratch cartridge (requiring a tape operator to decide which cartridge to mount). When using the 3495 we changed the default scratch processing (specified in the VMTAPE CONFIG file) to accept only volume specific scratch mounts. This forced VMTAPE to select scratch tapes from a list.

Figure 15 on page 22 shows the parameters we changed in the VMTAPE CONFIG file to accomodate using the 3495.

---

[7] Note that the 3494 generally can mount tapes faster than the 3495.

[8] If your 3495 cannot mount tapes in 5 minutes or less (this is possible in heavily used 3495s), increase the interval timer to a higher value.

```
REMINDER 600 999999 600
TAPEOPER OPATL
AUTOPICK NONE ACCEPT
```

Figure 15. Changed VMTAPE CONFIG File Parameters

The REMINDER parameter specifies the number of seconds to wait before repeating a mount message (600), the maximum number of seconds for a mount (999999), and the number of seconds to wait before reminding the operator of the number of pending allocations (600). The TAPEOPER parameter specifies the virtual machine that will receive the mount messages from VMTAPE. In this case, OPATL is specified. The AUTOPICK record configures VM:Tape to automatically select specific volumes to satisfy scratch mount requests ranther than requesting a generic “SCRATCH” request.[9]

3.1.2 Setting Up the Programmable Operator for Tape Mounts

To get the PROP virtual machine, OPATL, to act on mount messages from VMTAPE, you should first specify the OPATL service machine as the secondary user of RMSMASTR. To enable OPATL as a secondary user of RMSMASTR, simply update the CONSOLE directory statement of RMSMASTR, adding OPATL to the CONS statement. Once you have completed these actions, OPATL will log all messages that appear on the RMSMASTR console. Then, in the PROP routing table, you should define the messages on which OPATL is to act.

Figure 16 shows the routing table we used.

```
*        ----- SPECIFY THE PROP CONFIGURATION -----
*                           IDENTIFY THE LOGICAL OPERATOR
LGLOPR OPERATOR SYO3
*               BLANK IS '/',  ARBCHAR IS '$', "NOT" IS '¬'
TEXTSYM / $ ¬
*                    DO LOGGING WHEN THIS TABLE IS IN EFFECT
LOGGING ALL
ROUTE    ----- END OF CONFIGURATION SPECIFICATION -----
*-----------------------+ +-+ +-+ ++ +------+ +------+ +------+ +------+
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*-----------------------+ +-+ +-+ ++ +------+ +------+ +------+ +------+
/VMTMNT057A /             OO1 O11 O1 VMTAPE   SYO3     MOUNTATL
/VMTMNT280A /             OO1 O11 O1 VMTAPE   SYO3     MOUNTATL
/VMTMNT541A /             OO1 O11 O1 VMTAPE   SYO3     MOUNTATL
/VMTMNT058A /             OO1 O11 O1 VMTAPE   SYO3     MOUNTATL
                                     RMSMASTR SYO3
/VMTDRV142I /             OO1 O11 O1 VMTAPE   SYO3     RESTARTU
/VMTLABO72R /             OO1 O11 O1 VMTAPE   SYO3     VMTREPLY
*-----------------------+ +-+ +-+ ++ +------+ +------+ +------+ +------+
```

Figure 16. PROP Routing Table Used for Capturing VMTAPE Messages

[9] While not used by this installation, Sterling Software recommends that you use AUTOPICK NONE CANCEL. This will cancel scratch mount requests when there are no more scratch tapes in the pool.

The format of the table is:

TEXT The specific console text on which you want PROP to take action

SCOL The column on the console to start scanning for the text

ECOL The column on the console to end scanning for the text

TYPE The type of message (type 01 represents MSG NOH Other types include warnings (02), CP and CMS responses (03 and 05), other types of messages (SMSG - 04, EMSG - 06, and IMSG - 07), and secondary console messages (08).

USER The virtual machine that issued the message

NODE The system from which the message came

ACTN The EXEC to be executed when the specified message is found

PARM The parameters to be passed to the EXEC

Note!

Note: VM:Tape 1.1 changed some messages from VMTMNTxxx to VMTMANxxx. Scan for the messages that are appropriate for your release of VM:Tape.

Based on the message received, the following actions are taken:

- VMTMNT057A, VMTMNT280A, and VMTMNT541A messages are transformed to DFSMSRM MOUNT requests. See 3.1.3, “MOUNTATL EXEC.”
- The VMTMNT058A message should not occur because it is a generic request using a pool name. If it does occur, it is transformed to a VMTAPE CANCEL command by the MOUNTATL EXEC.
- The VMTDRV142I message is intercepted to start a possibly drained drive. VM:Tape will set a tape drive in drained status due to what it considers a non-recoverable I/O error. However, VM:Tape 1.1 just exempts the drive from future allocations by removing it from the eligible drive list.
- The VMTLAB072R message is intercepted to show operators the cartridge number to reply with.
- All RMSMASTR SCIF messages are logged.

3.1.3 MOUNTATL EXEC

When the PROP virtual machine calls the MOUNTATL EXEC, the EXEC executes the following steps:

1. Parses the input from VMTAPE for conversion to a DFSMSRM MOUNT request
2. Verifies that RMSMASTR is running
3. Verifies that the volume is available
4. Mounts the tape if it is found in the library.

3.1.3.1 Parsing MOUNTATL EXEC Input

The EXEC extracts the following variables from the mount messages:

- Volume label (external) to be mounted
- Real device address as selected by VMTAPE
  - From the real device address the EXEC extracts the VM:Tape userid (normally VMTAPE) and the virtual device address that VM:Tape is using for this request.
- Userid that requested the mount
- Virtual device address the user requested
- Read/write status the user requested.

3.1.3.2 Verify That RMSMASTR Is Operational

Next, the MOUNTATL EXEC verifies that RMSMASTR is logged on. If not, the mount request is canceled, and the EXEC terminates.

3.1.3.3 Verify That the Tape Volume Is in the Library and Mount If Found

Next, the MOUNTATL EXEC verifies that the requested tape volume is in the library inventory by scanning the TCDB. The TCDB is generated daily by OPATL (OPATL issues a DFSMSRM QUERY LIB INVENTORY ALL command and parses the result) and updated during the day if volumes are added to or ejected from the library.

- If the requested scratch volume is found in the tape library, the EXEC uses the DFSMSRM MOUNT command to mount the tape. It then attaches the tape drive to OPATL and writes a label to avoid the following two problems:
  - The tape is not labeled correctly
  - The cartridge R/W switch is accidentally set to R/O

    Note: Attaching the tape drive to OPATL instead of VMTAPE handles both problems properly. OPATL will write a volume label and indirectly determine the status of write protection. If writing the labels fails, then the tape should be demounted and ejected from the library, and the MOUNTATL EXEC should generate a VMTAPE REJECT command to fail this request and select a different scratch tape from the list.
- The MOUNTATL EXEC then attaches the tape drive to VMTAPE, and normal VM:Tape processing continues.
- If the requested non-scratch volume is found in the tape library, the EXEC issues a DFSMSRM MOUNT command to mount the tape. Then, the MOUNTATL EXEC attaches the tape drive to VMTAPE, and normal VM:Tape processing continues.
- If the requested scratch volume is not in the tape library, the EXEC issues a VMTAPE REJECT command for this volume. Then the EXEC terminates. VMTAPE issues a new scratch mount request (using a different scratch tape from the list).
- If the requested non-scratch volume is not in the tape library, the EXEC issues a VMTAPE CANCEL command for this volume. Then the EXEC terminates.

3.1.3.4 Mount Volume and Verify Label

If the volume is in the library, the MOUNTATL EXEC continues the mount by:

- Detaching the real device from VMTAPE (using the LEAVE option)
- Requesting a mount
- Verifying the label (if a nonscratch volume was requested) or writing a label (if a scratch tape was requested).

Figure 17 shows the MOUNTATL EXEC used for the prototype work described in this section. Use it as a sample for your own installation.

The MOUNTATL EXEC takes the VM:Tape mount message, parses the input, and converts the command into one that RMSMASTR will understand. The EXEC queries the tape library to ensure that the requested volume is in the library inventory and, if it is, requests that the tape be mounted (virtual and real addresses provided).

```
/*********************************************************************
* Function:                                                          *
*                                                                    *
*    Mounting of ATL cartridges.                                     *
*                                                                    *
* Called by:                                                         *
*                                                                    *
*    OPATL PROP on receipt of one of the following mount messages    *
*    from VMTAPE:                                                    *
*       VMTMNT057A , VMTMNT280A , VMTMNT541A and VMTMNT058A .        *
*                                                                    *
* Called execs:                                                      *
*                                                                    *
*    RMSRETRY : Perform recovery processing for failed mounts.       *
*                                                                    *
* Operation:                                                         *
*                                                                    *
*    After preliminary checks (volser existence, ATL status, volser  *
*    inuse on other unit), the unit on which the tape is to be mounted *
*    is detached from VMTAPE, and a DFSMSRM MOUNT command is generated *
*    and executed with the WAIT option in effect.                    *
*    The tape is mounted to OPATL 181, label checking is performed,  *
*    and the tape is given to VMTAPE at the original virtual address. *
*    The elapsed time during the MOUNTATL EXEC execution is logged to *
*    today's MOUNTATL log file.                                      *
*                                                                    *
* Created by: W.T.Elzinga , dec 1993 , RIFB/T1 , Fokker b.v.         *
*********************************************************************/
```

Figure 17 (Part 1 of 8). Sample MOUNTATL EXEC for Volume Mount

```
Address Command ; Signal On Syntax ; Numeric digits 12
Arg ¢req ; Pull tapemsg command ; 'DROPBUF'
stime = date('b')right(time('s'),5,0) ; sdate = time()

/* Process the VMTAPE mount message */
Call PROC_MSG

/* Get the VMTAPE VDEV */
'PIPE CP QUERY' raddr,
   '| VAR EMSG',
   '| FIND TAPE',
   '| SPECS /-VMTID-/ 1 w5 nw',
     'WRITE /-VMTVADDR-/ 1 w6 nw',
   '| VARLOAD'
If rc <> 0 Then call CAN_LOG emsg
If vmtid <> 'VMTAPE' Then call CAN_LOG 'Tape' raddr 'not at VMTAPE:' emsg

/* Cancel the mount if RMSMASTR not logged on */
'PIPE CP QUERY RMSMASTR',
   '| VAR EMSG'
If rc <> 0 Then call CAN_LOG emsg

/* Check if volume present in library */
'PIPE < ATL INVENTOR',
   '| FIND' volser||,
   '| SPECS 8.1 c2b n',
   '| VAR VOLATL'
Select
   When symbol('volatl') = 'LIT' Then reason = 'Volume' volser 'NOT',
      'found in the ATL'
   When volatl = '00000000' Then reason = ''
   Otherwise reason = ATL_STATUS(volatl)
End
Select
   When reason = '' Then nop                          /* Volume found */
   When voltype = 'SCRATCH' Then do          /* Reject a scratch mount */
      'VMTAPE REJECT' volser
      If rc <> 0 Then call CAN_LOG reason ', and REJECT failed.'
      Else do
         'CP SLEEP 1 SEC' ; 'VMTAPE START' raddr
         Call LOG reason
      End
   End
   When symbol('volatl') = 'LIT' Then call CAN_LOG reason,'Use',
      '"VMTAPE2" to mount foreign tapes'
   Otherwise nop       /* Non-scratch mount flagged volser: try anyway */
End
```

Figure 17 (Part 2 of 8). Sample MOUNTATL EXEC for Volume Mount

```
/* Check if library active */
'GLOBALV SELECT ATLSTAT GET STATUS'
If status <> '' & status <> 'AUTOMATED' Then ,
   Call LOG 'Mount not processed, ATL in' status 'status.'

/* Check if volume present in another tape unit */
'PIPE COMMAND GLOBALV SELECT ATLSTAT LIST',
   '| LOCATE 2.5 /UNIT./',
   '| LOCATE 12.6 /'volser'/',
   '| SPECS 7.4 1',
   '| VAR UNIT'
If symbol('unit') = 'VAR' & unit <> raddr Then do
   'PIPE CP Q' unit,
      '| SPECS /-USTAT-/ 1 w3 n',
        'WRITE /-UOWNR-/ 1 w5 n',
      '| VARLOAD'
   If ustat = 'ATTACHED' & uownr <> 'RMSMASTR' Then cal LOG 'Unit',
      unit 'containing the volser still ATTACHED by' uownr
   Do wtim = 5 By 5 To 580 while ustat = 'ATTACHED' & uownr = 'RMSMASTR'
      'PIPE LITERAL Q' unit,
         '| CP SLEEP 5 SEC',
         '| SPECS /-USTAT-/ 1 w3 n',
           'WRITE /-UOWNR-/ 1 w5 n',
         '| VARLOAD'
   End
   If ustat = 'FREE' Then do
      'DFSMSRM DEMOUNT RDEV' unit '(NOASSIGN VOLUME' volser
      'GLOBALV SELECT ATLSTAT SET UNIT.'unit
   End
   If wtim // 600 > 120 Then call LOG 'Waited' wtim 'seconds before',
      'DEMOUNT' unit 'could be started'
End
```

Figure 17 (Part 3 of 8). Sample MOUNTATL EXEC for Volume Mount

```
/* Steal the unit from VMTAPE for our usage */
'PIPE LITERAL DETACH' raddr 'FROM VMTAPE LEAVE',
   '| CP SLEEP 1 SEC',
   '| VAR EMSG'
Select
   When rc = 0 Then nop
   When rc = 1120 Then do /* Wait for the detach to succeed */
      Say emsg
      Do 20 until free = 'FREE'
         'PIPE LITERAL QUERY' raddr,
            '| CP SLEEP 3 SEC',
            '| CONSOLE',
            '| VAR EMSG',
            '| FIND TAPE',
            '| SPECS w3 1',
            '| VAR FREE'
      End
      If free <> 'FREE' Then call CAN_LOG 'Tape unit could not',
         'be detached from VMTAPE'
   End
   Otherwise call CAN_LOG emsg
End

/* If status unknown, then determine it here */
If status = '' Then do
   'PIPE COMMAND DFSMSRM QUERY LIB OPS (WAIT NOASSIGN RDEV' raddr,
      '| CONSOLE',
      '| FIND FSMBCF2160I',
      '| SPECS w6 1',
      '| VAR STATUS',
      '| SPECS /GLOBALV SELECT ATLSTAT SET STATUS/ 1 w1 nw',
      '| COMMAND'
   If symbol('status') = 'LIT' Then status = 'UNKNOWN'
   If status <> 'AUTOMATED' Then do
      Call GETBACK
      Call LOG 'ATL in' status 'status.'
   End
End
'FINIS * * A'
```

Figure 17 (Part 4 of 8). Sample MOUNTATL EXEC for Volume Mount

```
/* Now we let RMSMASTR mount the tape to ourselves at 181 */
If rwstatus= 'RING IN' Then rmstat= 'READWRITE' ; Else rmstat= 'READONLY'
Call diag 8,'DETACH 181'
'PIPE (endchar ?) COMMAND DFSMSRM MOUNT VOL' volser,
     '(WAIT RDEV' raddr rmstat 'TARGET VOL ATTACH * VDEV 181',
   '| VAR RMSMSG'
If rc <> 0 Then do
   Say rmsmsg ; retry = RMSRETRY(rmsmsg,0,raddr,volser)
   If retry = 2 Then do
      'PIPE COMMAND DFSMSRM MOUNT VOL' volser,
         '(WAIT RDEV' raddr rmstat 'TARGET VOL ATTACH * VDEV 181',
         '| VAR RMSMSG'
      If rc <> 0 Then do
         Say rmsmsg ; retry = RMSRETRY(rmsmsg,1)
      End
   End
   If rc <> 0 Then do
      Call GETBACK
      If retry Then call LOG rmsmsg
      Call CAN_LOG rmsmsg,'Call the HELPDESK'
   End
End
'GLOBALV SELECT ATLSTAT SET UNIT.'raddr volser
If voltype = 'SCRATCH' Then call SCRATCH ; Else call NONSCRATCH
Call LOG

NONSCRATCH: /* Perform label verification */
'PIPE TAPE',
   '| TAKE 1',
   '| FIND VOL1'||,
   '| SPECS 5.6 1',
   '| VAR LABVOLSER'
Select
   When rc <> 0 Then call OWNER_CHECK
   When volser = labvolser Then call TRANSFER
   Otherwise call OWNER_CHECK
End
Return
```

Figure 17 (Part 5 of 8). Sample MOUNTATL EXEC for Volume Mount

```
OWNER_CHECK: /* Odd label: Check ownership */
'PIPE (endchar ?) COMMAND VMTAPE LIST' volser,
   '| LOCATE /OWNER=/',
   '| chop: CHOP AFTER STR /OWNER=/',
   '? chop:',
   '| SPECS w1 1',
   '| VAR OWNER'
If owner = userid Then do
   If symbol('labvolser') = 'LIT' Then text = 'no label.'
      Else text = 'wrong label:' labvolser'.'
   Call diag 8,'WNG' userid 'Volume' volser 'contains' text
   Call diag 8,'WNG' userid 'Mount permitted because you own the tape.'
   Call TRANSFER
End ; Else do
   Call GETBACK
   Call CAN_LOG 'Missing or wrong tapelabel on volser:' volser
End
Return

SCRATCH: /* For scratch tapes, (re)write the VOL1 label */
'PIPE LITERAL' volser userid,
   '| SPECS /VOL1/ 1 PAD 0 w1 n.6 r PAD BLANK /0/ n.31 w2 n.39',
     'WRITE PAD 0 /HDR1/ 1.80',
   '| TAPE',
   '| TAKE 1',
   '| VAR RMSMSG'
If rc = 0 Then call TRANSFER ; Else do
   Call GETBACK
   'VMTAPE REJECT' volser ; rt = rc
   'DFSMSRM DEMOUNT RDEV' raddr '(WAIT'
   'DFSMSRM SET VOLCAT VOL' volser 'TARGETCAT EJECT (WAIT RDEV' raddr
   If rt = 0 Then call LOG rmsmsg
             Else call CAN_LOG rmsmsg,'Please retry the mount'
End
Return

GETBACK: /* Transfer the unit quietly back to VMTAPE */
'PIPE (endchar ?) LITERAL SLEEP 2 SEC',
   '| DETACH' raddr '* LEAVE',
   '| CONSOLE',
   '? COMMAND VMTAPE CMS GETBACK' raddr vmtvaddr,
   '| SPECS /MSG */ 1 1-* nw',
   '| CP'
Return

TRANSFER: /* Transfer the tape to VMTAPE at the desired address */
'PIPE CP GIVE 181 TO' vmtid vmtvaddr,
   '| VAR EMSG'
If rc = 0 Then return
'CP DETACH 181'
Call CAN_LOG emsg,'Call the HELPDESK'
```

Figure 17 (Part 6 of 8). Sample MOUNTATL EXEC for Volume Mount

```
ATL_STATUS: /* Convert first nonzero ATL status bit to description */
Arg $bit 2 $stat
Do pos = 1 By 1 While $bit = 0
   Parse var $stat $bit 2 $stat
End
Select
   When pos = 1 Then return 'Volume inaccessable'
   When pos = 2 Then return 'Volume mounted or queued for mount'
   When pos = 3 Then return 'Eject pending for volume'
   When pos = 4 Then return 'Volume being ejected'
   When pos = 5 Then return 'Volume misplaced'
   When pos = 6 Then return 'Volume label unreadable'
   When pos = 7 Then return 'Volume used during manual mode'
   When pos = 8 Then return 'Volume manually ejected'
   Otherwise return ''
End

CAN_LOG: /* Cancel the mount and LOG the reason */
Parse arg msg,extra_msg
'CP SMSG VMTAPE CANCEL' vaddr userid
'CP WNG' userid 'MOUNT' vaddr 'cancelled:'
'CP WNG' userid strip(msg)
'CP MSG *' msg
If extra_msg <> '' Then do
   'CP WNG' userid extra_msg
   'CP MSG *' extra_msg
End

LOG: /* Log the used time */
Parse arg msg ; etime = date('b')right(time('s'),5,0)
elapsed = etime-stime
elap = right(elapsed%60,2,0)':'right(elapsed//60,2,0)
logline = sdate elap left(userid,8) right(vaddr,4,0) volser raddr,
   left(voltype,8) msg
'PIPE VAR LOGLINE | >>' date('s') 'MOUNTLOG A'
Exit

Error: /* Log used time and exit */
Parse arg msg ; Call diag 8,'MSGNOH *' msg
etime = date('b')right(time('s'),5,0) ; elapsed = etime-stime
elap = right(elapsed%60,2,0)':'right(elapsed//60,2,0)
logline = sdate elap 'Error:' msg
'PIPE VAR LOGLINE | >>' date('s') 'MOUNTLOG A'
Exit
```

Figure 17 (Part 7 of 8). Sample MOUNTATL EXEC for Volume Mount

```
Proc_msg: /* Process the VMTAPE messages */
Signal value tapemsg ; Return

VMTMNT057A: /* Process a mount message */
Parse var command 'MOUNT ' voltype ' VOLUME ' volser . ' ON ' raddr,
                  ', ' rwstatus ', ' 'FOR ' userid vaddr '.'
volser = strip(volser,,"'") ; If voltype='' Then voltype= 'LIBRARY'
Return

VMTMNT058A: /* Cancel non volspecific pool request */
Parse var command 'MOUNT ' poolname ' SCRATCH ' type 'TAPE',
                 "ON '"  raddr "', " rwstatus ',' 'FOR ' userid vaddr '.'
'VMTAPE CANCEL' vaddr userid
'CP WNG' userid "MOUNT cancelled because no volser is specified"
Exit

VMTMNT280A: /* Process a mount message */
Parse var command 'MOUNT SELECTED ' poolname ' SCRATCH VOLUME ' volser .,
         ' ON ' "'" raddr "'" ', ' rwstatus ', ' 'FOR ' userid vaddr '.'
volser = strip(volser,,"'") ; voltype= 'SCRATCH'
Return

VMTMNT541A: /* Process a mount message */
Parse var command 'MOUNT' '('volser altvol ') ON ' raddr ,
                  ', ' rwstatus ',' 'FOR ' userid vaddr '.'
voltype = 'LIST' ; volser = strip(volser,,"'")
Return

Syntax: /* Process syntax error */
Say 'REXX error' rc 'in line' sigl':' Errortext(rc)
If datatype(raddr,'x') = 0 Then exit
Call diag 8,'DETACH' raddr '*'
If datatype(vmtvaddr,'x') = 0 Then exit
'CP SLEEP 1 SEC'
'VMTAPE CMS GETBACK' raddr vmtvaddr
Exit
```

Figure 17 (Part 8 of 8). Sample MOUNTATL EXEC for Volume Mount

Note

While not included in this example, you may want to add code to verify that the proper tape is loaded prior to rewriting the label.

3.1.4 Supporting EXECs and Programs for MOUNTATL

As the project to automate VM:Tape mount requests moved forward, several programs became necessary and/or useful. This section describes the programs developed. All EXECs and programs described here are included in the diskette in the back of the book.

3.1.4.1 Recovering from Mount Errors: The RMSRETRY EXEC

The RMSRETRY EXEC attempts to recover from failed tape mount requests. This EXEC is called by the MOUNTATL EXEC to determine whether the mount error is a recoverable error (in which case the return code sent to the MOUNTATL EXEC indicates that the mount request should be atempted again) or an unrecoverable error (in which case the return code sent to the MOUNTATL EXEC indicates that the request should not be attempted again). The message numbers from the DFSMSRM MOUNT command are passed to the RMSRETRY EXEC. The message numbers "understood" by the RMSRETRY EXEC are those that we have encountered during the VM:Tape mount automation project.

3.1.4.2 Interpreting RMSMASTR Messages: The ATLMSG EXEC

The ATLMSG EXEC was developed during the project to interpret HCPAMR messages issued by RMSMASTR when the status of the tape library changed. The EXEC helps detect when the library is changed from *automated* status to paused or manual.

3.1.4.3 Interpreting RMSMASTR Reader Files: The RDRFILE EXEC

The RDRFILE EXEC interprets reader files received from RMSMASTR and, if appropriate, uses them to update the TCDB.

Figure 18 on page 34 shows the sample RDRFILE EXEC.

```
/*********************************************************************
* Function:                                                          *
*                                                                    *
*    Interpret incoming RDR files from RMSMASTR and update the TCDB  *
*    file. If the RDR file was the result of a command executed on   *
*    behalf of another user, then the RDR file is transferred to that*
*    user.                                                           *
*                                                                    *
* Called by:                                                         *
*                                                                    *
*    OPATL PROP on receipt of a RDR file from RMSMASTR.              *
*                                                                    *
* Operation:                                                         *
*                                                                    *
*    The spoolfile id (spid) is isolated, and the INVUPD exec is     *
*    called to update the TCDB file.                                 *
*    GLOBALV is used to determine if the RDR file is to be sent to   *
*    another user. If so, the file is transferred to that user. If not,*
*    the file is left in the reader for future references.           *
*                                                                    *
* Created by: W.T.Elzinga , jan 1994 , RIFB/T1 , Fokker b.v.         *
*********************************************************************/
Address Command ; Pull 'RDR FILE' spid . ; 'DROPBUF'
'EXEC INVUPD' spid
'PIPE (endchar ?) CP Q RDR *' spid 'AVAIL ALL',
   '| DROP 1',
   '| SPECS /GLOBALV SELECT REQUEST LIST REQ./ 1 64.8 n',
   '| COMMAND',
   '| DROP 1',
   '| chop: CHOP AFTER =',
   '? chop:',
   '| LOCATE 3',
   '| SPECS /CHANGE RDR' spid 'TO/ 1 w1 nw',
   '| CP',
   '| NLOCATE /FILE  CHANGED/',
   '| CONSOLE'
Exit
```

Figure 18. Sample RDRFILE EXEC for Handling Reader Files from RMSMASTR

3.1.4.4 Updating the TCDB: The INVUPD EXEC and INVUPD REXX

The INVUPD EXEC creates a TCDB daily for use by the MOUNTATL EXEC. INVUPD REXX is called by the INVUPD EXEC to process the reader file when it is received from RMSMASTR. When the TCDB has been created, the information is read into storage using the CMS EXECLOAD command. The MOUNTATL EXEC can then scan the TCDB directly from virtual storage.

Figure 19 on page 35 shows the INVUPD EXEC (INVUPD REXX is not shown in the interest of brevity).

```
/*********************************************************************
* Function:                                                          *
*                                                                    *
*    This exec interprets a RDR file and uses the contents to create *
*    or update the TCDB file.                                        *
*                                                                    *
* Called by:                                                         *
*                                                                    *
*    The RDRFILE exec                                                *
*                                                                    *
* Called execs:                                                      *
*                                                                    *
*    INVUPD REXX : For updating the TCDB file.                       *
*                                                                    *
* Operation:                                                         *
*                                                                    *
*    The spoolfile id (spid) argument is checked.                    *
*    The RDR file is read , and the INVUPD REXX is called to process *
*    the RDR file contents. After successfull completion, the        *
*    resulting TCDB file is read into storage using a EXECLOAD command. *
*    Scanning the TCDB is done using pipelines reading directly the  *
*    file in core.                                                   *
*                                                                    *
* Created by: W.T.Elzinga , jan 1949 , RIFB/T1 , Fokker b.v.         *
*********************************************************************/
Address command
Arg spid . ; If spid <> '' Then spid = 'FILE' spid

'ESTATE ATL INVENTOR A'
If rc = 0 Then rdpipe = '? < ATL INVENTOR | invupd:' ; Else rdpipe = ''
'PIPE (endchar ?) CP SP RDR HOLD',
   '? READER' spid,
   '| FIND' '41'x||,
   '| SPECS 2-* 1.80',
   '| DEBLOCK NETDATA',
   '| FIND' 'C0'x||,
   '| SPECS 2-* 1',
   '| invupd: INVUPD',
   '| > ATL INVENTOR A2 F',
      rdpipe
If rc <> 0 Then exit rc
'GLOBALV SELECT ATLSTAT GET SECLEVEL'
If seclevel = 1 Then msg = '' ; Else msg = ,
   '? CP QUERY SYO3',
   '| LOCATE /-/',
   '| SPECS /SMSG RSCS MSG SYO3 OPATL UPDATED/ 1',
   '| CP'
'PIPE (endchar ?) COMMAND EXECMAP ATL INVENTOR',
   '| DROP 1',
   '| SPECS /EXECDROP ATL INVENTOR/ 1',
   '| COMMAND',
   '? COMMAND EXECLOAD ATL INVENTOR',
      msg
Exit
```

Figure 19. Sample INVUPD EXEC

3.1.5 Other Support EXECs and Programs

The following list describes other EXECs and programs that are included in the package available from IBM. This package can be obtained by having your IBM representative contact the author (Craig Welch, CW1313 at STLVM4) or you may contact the author directly by sending electronic mail to CW1313@STLVM4.VNET.IBM.COM.

RMSOUT EXEC Processes asynchronous RMSMASTR output

DRAINEDU EXEC Performs cleanup functions for canceled or failed tape mounts

HELPCMD EXEC Displays information on DFSMSRM commands which OPATL can execute on behalf of users

LOGREQ EXEC Queries and logs requests to RMSMASTR from other users (done to monitor DFSMSRM usage that bypasses OPATL)

QOPS EXEC Queries the operational status of the tape library

EJECT EXEC Ejects a single volume to the Convenience I/O Station or the high-capacity I/O facility

EJECTF EXEC Ejects a group of volumes to the convenience I/O station or the high-capacity I/O facility

INVENTOR EXEC Executes a DFSMSRM Q LIB INVENTORY command on behalf of a user

PROFILE EXEC Executes the initial setup of the OPATL virtual machine

PROPPROF EXEC Initializes the PROP environment

MIDNIGHT EXEC Performs cleanup functions for the OPATL virtual machine

3.2 General Comments

Readers should recognize that the implementation described in this chapter worked at a specific installation. While the concepts are easily transferred to other installations, some modifications may be required in other installations. Some things you may want to add or consider if you wish to implement procedures similar to those described here:

- Avoid using DETACH/ATTACH processing where possible, use GIVE processing instead. In installations sharing tape drives, this will help avoid having the drive stolen.
- The VM:Tape allocation exit should be considered to augment control of mounts to the 3494 tape drives.
- If you are mixing both manual tape drives and automated tape libraries, perform thorough testing to ensure the mount requests are routed appropriately.

3.3 Turning Out the Lights for Tape Management Systems

To get the EXECs described in this chapter, ask your IBM representative obtain the TMSLTOUT PACKAGE from VMTOOLS.[10] Alternately, you or your representative can contact the author directly (Craig Welch, CW1313@STLVM4.VNET.IBM.COM or at the address listed with the edition notice in the front of this book).

[10] IBM representatives should use their local VMTOOLS shadow or the command TOOLS TO VMTOOLS AT RALVM17 GET TMSLTOUT PACKAGE.

Chapter 4. Multiple VM Systems Sharing a 3495

This chapter describes an installation's use of the IBM 3495 in a multisystem environment. A single 3495 and a set of 3490 tape drives are shared among four first-level systems and four second-level systems (running on two of the first-level systems). Locally developed software facilitates the sharing of 3490 tape drives and controls access to tape volumes.

Please Note!

The code samples shown in this chapter worked at a specific installation to solve a specific problem. While the technique used should be helpful when working in similar environments, the samples may have to be tailored for your installation.

4.1 Challenges of a Multisystem Environment

Sharing tape drives in a multisystem environment has always been challenging because a single tape drive can be used by only one host system at a single point in time. Tape drives are available to a user when the drive is online and *attached* to that user. Once successfully attached to a user, the drive is normally *assigned* to that system, meaning that no other system can use the drive until it is unassigned by that system. In most cases this works well. When second-level systems are involved, however, the first-level system must attach the drive to the second-level guest by means of the NOASSIGN option. Use of this option carries with it the danger that another system could find the drive "available."

IBM Canada solved the tape sharing problem by developing a "real tape manager." This software allowed the first-level system to give tapes to second-level guests and second-level guests to pass unneeded tapes back to the first-level system (where, possibly, another first- or second-level guest could use the tape drives). See 4.5, "BRUNOGTA: Real Tape Manager" on page 59 for specific details.

4.2 Overview of the IBM Canada Datacenter

The implementation of the 3495 Tape Library Dataserver (nicknamed BRUNO[11] at this installation) on the IBM Canada internal VM systems involves four components:

- A 3495-L20 Tape Library Dataserver shared by all four systems
- DFSMS/VM Function Level 221 Removable Media Services (RMS) software running in the RMSMASTR virtual machine on each first- and second-level system
- Locally developed user interface software, running in the mount manager (BRUNOMNT) on each first- and second-level system
- Locally developed guest tape allocation software, running on the two systems with second-level guests.

[11] BRUNO stands for Backup and Recovery Unit, No Operator.

Figure 20 on page 40 shows the system configuration used at the IBM Canada location.

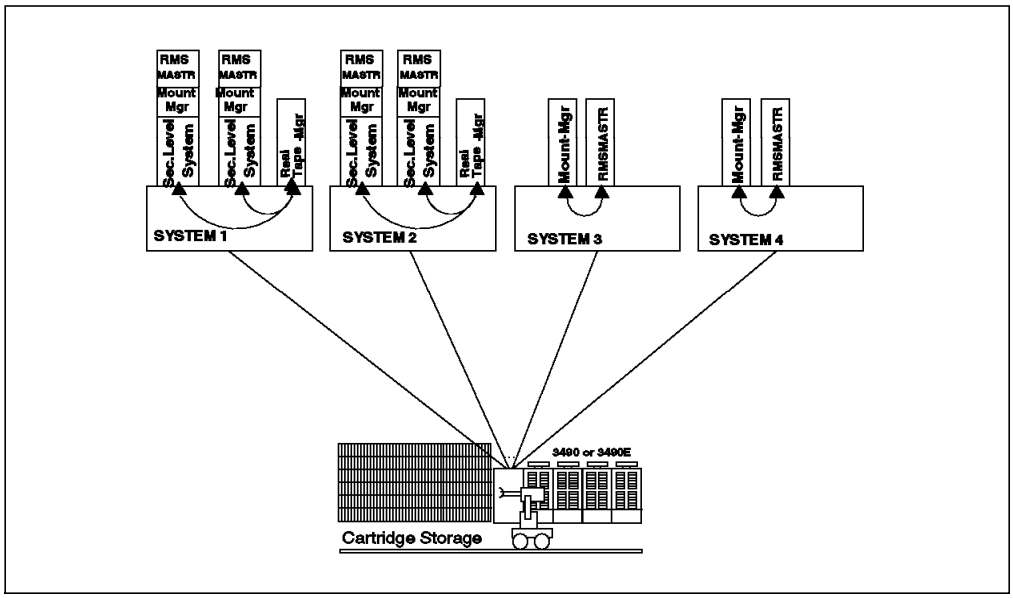

Figure 20. The Multi-VM Solution

On SYSTEM3 and SYSTEM4, the mount manager and the RMSMASTR are the only virtual machines involved in tape management. On SYSTEM1 and SYSTEM2, a “real tape manager” on the first-level system controls which real 3490 tape drives are online and available to the second-level guest systems. Each second-level system has a mount manager and an RMSMASTR to control tape cartridge mounts and demounts within that system.

The IBM 3495 Tape Library Dataserver at IBM Canada provides automated tape mounting for:

- Authorized end users
- Host system backup (using VMBARS)[12]
- Workstation backup (using WDSF (now available as ADSM)).[13]

The largest benefit for this installation has been the total automation of host and workstation backups.[14]

Note: IBM Canada has experimented with DFSMS/VM space management, storing migration level 2 data on the 3495. Although the experimental work was successful, it has not been implemented on a systemwide basis.

[12] VMBARS, the VM Backup and Archival Retrieval System, is an IBM program offering. It performs incremental backup of CMS files. Backup can be from disk to disk or disk to tape.

[13] WDSF, the Workstation DataSave Facility, is an IBM program product that backs up workstation data to host VM systems. This product has been superceded by ADSM, the ADSTAR Distributed Storage Manager.

[14] ADSM allows use of a multilevel hierarchy for storing backup data. IBM Canada places backup data on host DASD. As the DASD fills, ADSM migrates data to the next level in the hierarchy (tape in their implementation). With the 3495, this process is entirely automated.

4.3 Customizing RMSMASTR for BRUNO

DFSMS/VM Function Level 221 provides exits for installations that need to add removable media services function to the base code. IBM Canada needed to customize the tape partitioning exit (FSMRMSHR) to control access to volumes in the 3495. Additionally, the installation used RACF to authorize access to tape volumes and RACF command authorization to control access to DFSMSRM commands.

4.3.1 RMSMASTR Customization Steps

IBM Canada made the following changes (and additions) to allow controlled, multisystem sharing of a 3495:

- Set up the automatic bulk insert file (RMB10023 DATA)
- Installed a locally developed FSMRMSHR exit, which calls FSMRMSHR EXEC and FSMRMSHR REXX and references control information in FSMRMSHR CTLDATA
- Implemented RACF command authorization (see 4.3.7, “Authorization” on page 49).

4.3.2 Automatic Bulk Insert File

The automatic bulk insert file specifies ranges of volsers and target categories for volumes with the specified volsers. When an automatic bulk insert file is used, volumes inserted into the Tape Library Dataserver will automatically have their category assignment changed from INSERT to the desired category. Figure 21 shows the automatic bulk insert file used by IBM Canada.

```
*
* For automatic bulk insert processing, this file must be named
* RMBxxxxx DATA, where xxxxx = sequence number of the 3495
* associated with the file.
* This file must exist on the VMSYS:DFSMS.CONTROL directory...
*
* use this file for automatic bulk processing
*
A     -9         SCRATCHF
AA    -99        SCRATCHF
AAA   -999       SCRATCHF
AAAA  -9999      SCRATCHF
AAAAA -99999     SCRATCHF
AAAAAA-999999    SCRATCHF
```

Figure 21. Sample Automatic Bulk Insert File from IBM Canada

4.3.3 FSMRMSHR Exit

The FSMRMSHR exit allows you to partition tapes in the library among multiple systems, to ensure that each system can only access volumes assigned to it. The MOUNT, SET DEVCAT, and SET VOLCAT commands are passed to the exit, which sets the following return codes:

0 accept request
8 reject request with a return code of 8 and a reason code 3100.

The FSMRMSHR exit is an assembler routine that accepts the parameters passed by RMSMASTR and calls the FSMRMSHR EXEC on the DFSMS(1B5) disk. We chose to implement most of the logic for this exit in the FSMRMSHR EXEC, using the REXX programming language, rather than in the assembler module itself. This method made making changes to the logic easier and more reliable than assembler coding.

The FSMRMSHR EXEC in turn uses CMS Pipelines to process the FSMRMSHR CTLDATA file in the VMSYS:DFSMS.CONTROL directory. FSMRMSHR EXEC invokes a locally developed Pipeline stage called FSMRMSHR REXX, also on the DFSMS(1B5) disk.

Figure 22 shows the assembler code we used. Before using it, you should understand how to use the DFSMS/VM exits. See the *VM/ESA DFSMS/VM Function Level 221 Installation and Customization* manual and the *VM/ESA DFSMS/VM Function Level 221 Removable Media Services User's Guide* for more information.

```
*************************************************************************
* FSMRMSHR RMS Library Partitioning Exit                                *
*                                                                       *
*************************************************************************
FSMRMSHR CSLENTRY (RETCODE,VOLSER,SRCCAT,SLEN,TRGCAT,TLEN,LIBNAME,LLEN)
         L    R12,=A(FSMRMSHR)
         DROP R15
         USING FSMRMSHR,R12
*
         LR   R2,R1               Save A(parm list)
         SLR  R7,R7               Set return code to 0
*
* Clear parameter area to blanks
        MVI   PARMS,C' '
        MVC   PARMS+1(LPARMS-1),PARMS
*
* Get the 2nd parameter (volume label) and save it
GETVOL   DS     0H
        CSLGETP PLIST=(R2),PARM=VOLSER,ADDRESS=(R4)
        MVC    VOLSER,0(R4)
        CLC    VOLSER,BLANKS     If field not blank
        BNE    GETSRCCT            Then get next field
*                                   Else set it to *NONE*
        MVC    VOLSER(L'CNONE),CNONE
*
```

Figure 22 (Part 1 of 3). FSMRMSHR Assembler Exit

```
* Get the 3rd parameter (source category) and save it
GETSRCCT DS     0H
         CSLGETP PLIST=(R2),PARM=SRCCAT,ADDRESS=(R4),LENGTH=(R5)
         LA     R14,SRCCAT        -> 'to' field
         LA     R15,L'SRCCAT      Length of 'to' field
         ICM    R3,B'1000',=C' '  Pad with blanks
         MVCL   R14,R4            Move in source category
         CLC    BLANKS,SRCCAT     If field not blank
         BNE    GETTRGCT            Then get next field
*                                  Else set it to *NONE*
         MVC    SRCCAT(L'CNONE),CNONE
*
* Get the target category and save it in the parmlist
GETTRGCT DS     0H
         CSLGETP PLIST=(R2),PARM=TRGCAT,ADDRESS=(R4),LENGTH=(R5)
         LA     R14,TRGCAT        -> 'to' field
         LA     R15,L'TRGCAT      Length of 'to' field
         ICM    R3,B'1000',=C' '  Pad with blanks
         MVCL   R14,R4            Move in source category
         CLC    BLANKS,TRGCAT     If field not blank
         BNE    GETLIBNM            Then get next field
*                                   Else set it to *NONE*
         MVC    TRGCAT(L'CNONE),CNONE
*
* Get the library name and save it in the parmlist
GETLIBNM DS     0H
         CSLGETP PLIST=(R2),PARM=LIBNAME,ADDRESS=(R4),LENGTH=(R5)
         LA     R14,LIBNAME       -> 'to' field
         LA     R15,L'LIBNAME     Length of 'to' field
         ICM    R3,B'1000',=C' '  Pad with blanks
         MVCL   R14,R4            Move in source category
         CLC    BLANKS,LIBNAME    If field not blank
         BNE    DONEPARM            Then get next field
*                                  Else set it to *NONE*
         MVC    LIBNAME(L'CNONE),CNONE
*
DONEPARM DS     0H
*
*-------------------------------------------------------------*
*   set up parameters to call rexx program for validation     *
*-------------------------------------------------------------*
         LA   R6,LEXSTRNG       * SIZE OF TOKEN PARAMETER
         SCAN TEXT=(EXSTRING,(R6)),BUFFER=(SCANBUFR)
         LR   R4,R0             * EXTENDED PLIST @ IN R4
*
*-------------------------------------------------------------*
*   CALL REXX PROGRAM                                         *
*-------------------------------------------------------------*
         CMSCALL PLIST=(R1),EPLIST=(R4),CALLTYP=FUNCTION,ERROR=RC8ABND
         B    RETURN
*
```

Figure 22 (Part 2 of 3). FSMRMSHR Assembler Exit

```
*-------------------------------------------------------------*
*   reflect return code 8 after unsuccessful verify           *
*-------------------------------------------------------------*
*
RC8ABND  EQU  *
         LA   R7,8
         SPACE 1
* Pass back the return code via the CSL parameter list as well
* as the CSLEXIT macro
RETURN   DS   0H
         CSLGETP PLIST=(R2),PARM=RETCODE,ADDRESS=(R4)
         ST   R7,0(,R4)
         CSLEXIT RETURN=(R7)
*
*
         DS   0D
*
EXSTRING DS   0C
EXECNAME DC   CL8'EXEC'
         DC   CL8'FSMRMSHR'
PARMS    EQU  *
         DC   CL1' '
VOLSER   DC   CL6' '
         DC   CL1' '
SRCCAT   DC   CL32' '
         DC   CL1' '
TRGCAT   DC   CL32' '
         DC   CL1' '
LIBNAME  DC   CL32' '
         DC   CL1' '
LPARMS   EQU  *-PARMS
LEXSTRNG EQU  *-EXSTRING
         SPACE 1
BLANKS   DC   CL8' '
CNONE    DC   C'*NONE*'
         SPACE 1
SCANBUFR DS   CL300
*
         REGEQU
         END
```

Figure 22 (Part 3 of 3). FSMRMSHR Assembler Exit

4.3.4 FSMRMSHR EXEC

The FSMRMSHR EXEC is called by the FSMRMSHR exit (in FSMPSI CSLLIB) for all MOUNT, SET DEVCAT, and SET VOLCAT commands. Parameters passed are:

- The volid of the tape cartridge or *NONE* (if request is for a SCRATCHx category)
- The source category, if request is not for a specific volid
- The target category
- The library name.

The FSMRMSHR EXEC accesses the VMSYS:DFSMS.CONTROL (if not already accessed). It uses a CMS Pipeline to scan the FSMRMSHR CTLDATA file, looking for records starting with the local nodeid, or the keyword ALL. The

requested volid or SCRATCHx category is then checked against the values on the rest of the records, either directly by the Pipeline if a single value, or using the FSMRMSHR REXX Pipeline stage for ranges of values.

If a match is found, the FSMRMSHR EXEC sets a return code of 0, allowing the request to complete. Otherwise, it sets a return code of 8, which causes the request to fail with a reason code 3100 error: “FSMRMSHR Error - The return code from the installation exit FSMRMSHR indicates processing should not be allowed to continue.” BRUNOMNT translates this into the message “*(FSMRMSHR) nnnnnn is not assigned to this system.”

Figure 23 shows the FSMRMSHR EXEC we used. Remember that it is called by FSMRMSHR ASSEMBLE, the cartridge-sharing exit that DFSMS/VM provides.

```
/*********************************************************************/
/* This EXEC is invoked by FSMRMSHR library partitioning exit        */
/* It scans FSMRMSHR CTLDATA VMSYS:DFSMS.CONTROL                */
/* to ensure that the                                                */
/*   volid or source category is assigned to this system             */
/*                                                                   */
/* 931107: Accept request if source category is INSERT (automatic    */
/*         bulk insert processing                                    */
/* 930829: In the event that no volid or source category is provided */
/*         (ie. DEMOUNT RDEV ccuu), exit immediately                 */
/*********************************************************************/
Trace Off
Address COMMAND
  Parse UPPER ARG volid source_cat target_cat libname .
  fileid = 'FSMRMSHR CTLDATA'     /* authorised volids by node       */
  dirid  = 'VMSYS:DFSMS.CONTROL'/* ...and where it can be found*/
  mode = FileMode(fileid dirid)  /* ensure file associated with fmode*/
  match = 0                       /* assume NOT authorised           */
  If mode ¬= '' Then              /* If file accessable              */
    Select                        /*   Then check volid/source_cat   */
      When source_cat = 'INSERT'  /* If inserting volume             */
        Then match = 1            /*   Then accept request           */
      When volid ¬= '*NONE*'      /* If volid specified, check volid */
        Then match = FindMatch(fileid mode , volid)
      When source_cat ¬= '*NONE*' /*If source cat specified, check it*/
        Then match = FindMatch(fileid mode , source_cat)
      Otherwise match = 1         /* Accept request if both omitted  */
    End

  Return 8 * (match = 0)          /* Return 8 if no match found      */
```

Figure 23 (Part 1 of 3). FSMRMSHR EXEC

```
/*********************************************************************/
/* Find a file and ensure it is associated with a filemode           */
/*********************************************************************/
FileMode: PROCEDURE

  Parse UPPER ARG fn ft dirid .

  fmode  = ''                     /* in case of error                */
  cslrc  = 0                      /* CSL return code                 */
  cslrs  = 0                      /* CSL reason code                 */
  filespec = fn ft dirid          /* file specification              */
  lfilespec = Length(filespec)    /* length of file specification    */
  commit = 'NOCOMMIT'             /* do not commit changes           */
  lcommit= Length(commit)         /* length of commit option         */
  workunitid = 0                  /* default workunitid              */
  lwuerror = 0                    /* suppress return of 'wuerror'    */
  userdata = ''                   /* ESM user data                   */
  luserdata = Length(userdata)    /* length of user data             */
  requestid = 0                   /* synchronous request             */

/* Call CSL to check existence of file */
  Call CSL 'DMSEXIFI cslrc cslrs filespec lfilespec commit lcommit' ,
                   'rd_auth wr_auth extern_protect status' ,
                   'fileid fm_no recform rec_len num_blks num_recs',
                   'date time owner_userid' ,
                   'workunitid wuerror lwuerror' ,
                   'dateref userdata luserdata' ,
                   'create_date create_time' ,
                   'recoverability overwrite requestid migrated' ,
                   'max_blocks data_blocks system_blocks ',
                   'dra_values uniqueid fmode real_fmode'

  If cslrc ¬= 0
    Then Say 'Error' cslrc'/'cslrs 'checking existence of "'filespec'"'
    Else If fmode = ''  Then
           Do
             fmode = GetFmode()
             If fmode ¬= ''
               Then 'ACCESS' dirid fmode
           End

    Return fmode
```

Figure 23 (Part 2 of 3). FSMRMSHR EXEC

```
/*********************************************************************/
/* Find a free filemode                                              */
/*********************************************************************/
GetFmode: PROCEDURE

  fmode  = ''                    /* in case of error                */
  cslrc  = 0                     /* CSL return code                 */
  cslrs  = 0                     /* CSL reason code                 */

/* Call CSL to get a free file mode */
  Call CSL 'DMSGETFM cslrc cslrs fmode'

  If cslrc ¬= 0
    Then Say 'Error' cslrc'/'cslrs 'obtaining free file mode'

    Return fmode
/*********************************************************************/
/* Check to see if specified volid/source category is assigned to this*/
/* system (ie. nodeid from IDENTIFY)                                 */
/*********************************************************************/
FindMatch: PROCEDURE

  Parse UPPER ARG fileid , volid .

  'IDENTIFY ( LIFO'
  Parse UPPER PULL . . nodeid .

  match = ''
  'PIPE (endchar $)' ,
    '<' fileid ,                 /* read each record in file         */
    '| xlate' ,                  /* translate all to upper-case      */
    '|node: find' nodeid'_' ,    /* find records for this node       */
    '|nodeok: faninany' ,        /* common point for node/ALL records */
    '| specs words 2-* 1' ,      /* strip off the node               */
    '| split at blank' ,         /* break out each specification     */
    '| specs 1-* 1 / /next' ,    /* ensure trailing blank            */
    '|range: nlocate /-/' ,      /* If a single volid                */
    '| find' volid'_'||,         /*   Then check for match           */
    '|common: faninany' ,        /* common return point              */
    '| var match' ,              /* set MATCH if any records matched */
    '$' ,                        /* sec. stream from 'find nodeid'   */
    'node:' ,
    '| find ALL_' ,              /* find records for ALL nodes       */
    '|nodeok:' ,                 /* return to mainline               */
    '$' ,                        /* sec. stream from 'locate "-"'    */
    'range:' ,                   /* ...a range of volids             */
    '| rexx fsmrmshr' volid ,    /* Call FSMRMSHR REXX               */
    '|common:'                   /* return to mainline               */

  Return ('VAR' = Symbol('match'))
```

Figure 23 (Part 3 of 3). FSMRMSHR EXEC

4.3.5 FSMRMSHR REXX

Figure 24 shows the FSMRMSHR REXX Pipeline used to parse the control data file for tape sharing.

```
/* Handle FSMRMSHR CTLDATA range specifications                        */
/* Called by the FSMRMSHR EXEC Pipeline                                */

  Parse UPPER ARG match_on .      /* get the volid/category to check  */

  match_on = MapSCRATCH(match_on) /* remap if SCRATCHx                */

  Do FOREVER                      /* read pipeline until exhausted    */
    'readto record'               /* read a pipeline record           */
    If rc ¬= 0                    /* If errors                        */
      Then Leave                  /*   Then get out now               */
                                  /* get start/end specifications     */
    Parse VAR record start '-' end .
    start = MapSCRATCH(start)     /* remap if SCRATCHx                */
    end   = MapSCRATCH(end)

    If start > end                /* must be in sorted order         */
      Then Return 8

    If match_on >= start & match_on <= end
      Then 'output' record        /* match on range, feed the pipeline*/
  End

  Return 0

/* Remap SCRATCH pool identifiers 'A'-'F' to x'FA'-x'FF'               */
MapSCRATCH: PROCEDURE
  Parse UPPER ARG cat .

  tablei = Xrange('A', 'F')       /* remap 'A' to 'F' above '9'       */
  tableo = Xrange('FA'x, 'FF'x)
  If Substr(cat,1,7) = 'SCRATCH'  /* but only if SCRATCHx category    */
    Then cat = 'SCRATCH' || Translate(Substr(cat,8), tableo, tablei)

  Return cat
```

Figure 24. FSMRMSHR REXX

4.3.6 FSMRMSHR CTLDATA

The FSMRMSHR CTLDATA file (shown in Figure 25 on page 49) restricts each system to a specific range of volids but allows all systems to share access to:

- SCRATCH0/1 (testing)
- SCRATCHD (VMBARS disaster, backup and recovery (DBR) pool - unused volumes)
- SCRATCHE (VMBARS DBR pool - used tapes ready for EJECT)
- specific volids for tapes in the DBR pool (for mount requests by volid)

The FSMRMSHR CTLDATA file is reread on each invocation of the FSMRMSHR EXEC. Therefore, the file can be updated dynamically without requiring recycling of RMSMASTR. Because the file is stored in the Shared File System, there is no need for the FSMRMSHR EXEC to reaccess the disk.

```
* This file specifies which tape volumes and SCRATCH categories
* can be accessed by each system.  The format is:
*   node   <volid<-volid>> <SCRATCHx<-SCRATCHy>> ...
* where:
*   node      is the node of the processor, or ALL for all processors
*   volid     is a tape volume label
*   SCRATCHx  is a SCRATCH category (n = 0 to F)
*
* If a range of volumes is specified, the second volume must be higher
*   (standard sort sequence) than the first volume.

* The file can be common for multiple processors.

ALL      SCRATCH0 SCRATCH1 601200-601799 SCRATCHD-SCRATCHF

SECLEV1  600000-600099 601130-601149 SCRATCH2
SECLEV2  600100-600199 601100-601129
SYSTEM4  600200-600499
SECLEV3  600500-600799
SYSTEM3  600800-601099
SECLEV4  601150-601199
```

Figure 25. FSMRMSHR CTLDATA File

4.3.7 Authorization

Authorization to DFSMS/VM's removable media commands (DFSMSRM) and tape volume access is through RACF/VM. Although the implementation allows a wide range of authorization levels, the current implementation distinguishes between general users (limited access to RMSMASTR commands, access to specific tape cartridge volids) and RMS administrators (full access to RMSMASTR commands and all tape volids).

RMS administrator userids (including BRUNOMNT) are identified by being CONNECTed to the RACF/VM RMSADMIN group. This group is also used to control access to the DFSMS (1B5) program disk.

4.3.7.1 RMSMASTR Command Authorization

Access to privileged RMSMASTR commands is controlled through RACF/VM, as described in Chapter 6 of the *VM/ESA DFSMS/VM Function Level 221 Installation and Customization* manual and Chapter 2 of the *VM/ESA DFSMS/VM Function Level 221 Removable Media Services User's Guide and Reference*.

Generic and specific profiles are defined in the RACF/VM FACILITY class. RACF/VM users and groups of users can be authorized to use commands protected by these profiles by using the RACF PERMIT command to grant profile access. When a user issues a command, RMSMASTR uses the RACF/VM RACROUTE interface to determine whether the user has access to the associated profile.

Currently, all RMSMASTR commands with the exception of DISCARD (own) and QUERY REQUESTS (own) are considered privileged commands. Note that MOUNT is a privileged command, to ensure that BRUNOMNT can enforce RACF/VM protection of tape volumes.

Figure 26 on page 51 shows a sample RMSRACF EXEC. It can be used to set up RACF/VM to protect tape commands and tape resources. In addition to defining the various profiles and providing appropriate access to RMSADMIN and general users, the RMSRACF EXEC:

- Allows RMSMASTR and BRUNOMNT to use the RACROUTE interface
- Gives RMSADMIN userids access to the DFSMS(1B5) disk
- Creates a generic BRUNO.* profile to control access to tape volumes and gives RMSADMIN userids access to it.

```
/*********************************************************************/
/*                                                                   */
/*  Exec Name:    RMSRACF                                            */
/*                                                                   */
/*  Description:  Issue RACF/VM commands to setup authority checking */
/*                                                                   */
/*  Copyright:    5684-112 (C) Copyright IBM Corp. 1991              */
/*                Restricted Materials of IBM                        */
/*                Licensed Materials - Property of IBM               */
/*                Refer to Copyright instructions, form G120-2083    */
/*                                                                   */
/*  Status:       DFSMS/VM Function Level 220                        */
/*                                                                   */
/*  Function:     This EXEC issues RACF/VM commands to allow DFSMS   */
/*                to use RACF/VM for authority checking of DFSMS/RMS */
/*                commands.  The processing is as follows:           */
/*                                                                   */
/*                1. Issue RACF/VM commands to permit the RMSMASTR   */
/*                   to use RACROUTE                                 */
/*                2. Issue RACF/VM commands to protect all DFSMSRM   */
/*                   commands.  All commands except for MOUNT and    */
/*                   the 'OWN' versions of DISCARD and QUERY REQUESTS*/
/*                   are reserved to RMS administrators.  The three  */
/*                   commands above are set up as END USER commands. */
/*                                                                   */
/*  Assumptions:  This EXEC assumes the following:                   */
/*                                                                   */
/*                1. The userid of the Removable Media Services      */
/*                   server is the defaults supplied with DFSMS.  If */
/*                   you are not using the default userid, you must  */
/*                   modify STEP 1 with the appropriate userid.      */
/*                                                                   */
/*                2. All storage administrators are in the RACF      */
/*                   group RMSADMIN.  If this is not the case, then  */
/*                   you should update the CONSTANTS section at the  */
/*                   beginning of the EXEC to indicate the correct   */
/*                   GROUP name.  If you are not using groups, then  */
/*                   you must modify STEP 2 to issue the necessary   */
/*                   RACF/VM commands.                               */
/*                                                                   */
/*********************************************************************/
/*  Change Activity:                                                 */
/*                                                                   */
/* Flag Reason Vers'n Date  Origin:   Commentary              Fiche */
/* Id   Code   & Rlse       Code :                            Flag */
/* ----------------------------------------------------------------- */
/* $S0=HDF2220,2.2.0,910916,TUCMJA: Initial Release                  */
/* $S2=HDF2221,2.2.1,921117,TUCKGM: 221 updates                      */
/*********************************************************************/
```

Figure 26 (Part 1 of 2). RMSRACF EXEC

```
Trace E
/*********************************************************************/
/* CONSTANTS - Define some constants for use throughout the EXEC     */
/*********************************************************************/
storad_group = 'RMSADMIN'

/*********************************************************************/
/* STEP 1 - Set up profiles to allow DFSMS/VM master and server      */
/*          machines to communicate with the RACF/VM server machine. */
/*********************************************************************/

/* Define the ICHCONN profile in case it isn't already defined       */
'RAC RDEFINE FACILITY ICHCONN UACC(NONE)'

/* Give each of the default master/server userids UPDATE authority   */
'RAC PERMIT ICHCONN CLASS(FACILITY) ID(RMSMASTR) ACCESS(UPDATE)'
'RAC PERMIT ICHCONN CLASS(FACILITY) ID(BRUNOMNT) ACCESS(UPDATE)'
'RAC SETROPTS CLASSACT(FACILITY)'                          /*@01A*/

/*********************************************************************/
/* STEP 2 - Set up profiles to protect the DFSMSRM commands.  This   */
/*          step assumes that a group exists called RMSADMIN, and    */
/*          that all storage administrators are members of the group.*/
/*********************************************************************/

/* Allow generic profiles in the FACILITY class                      */
'RAC SETROPTS GRPLIST GENERIC(FACILITY) GENCMD(FACILITY)'   /*@02C*/

/* Define the generic profile to protect all commands, and permit    */
/* only the storage administrators in group RMSADMIN access to the   */
/* commands.                                                         */
'RAC RDEFINE FACILITY STGADMIN.RM.* UACC(NONE)'
'RAC PERMIT STGADMIN.RM.* CLASS(FACILITY) ID('storad_group')',
                        'ACCESS(READ)'

/* Define discrete profiles for the 'OWN' versions of DISCARD and    */
/* QUERY REQUESTS with universal access of READ, allowing all users  */
/* access to these two commands.  Note that MOUNT is not a general-  */
/* class command, since authorization checking is handled by the     */
/* BRUNOMNT server.  BRUNOMNT should be CONNECTed to the RMSADMIN    */
/* group.                                                            */
'RAC RDEFINE FACILITY STGADMIN.RM.DISCARD.OWN UACC(READ)'
'RAC RDEFINE FACILITY STGADMIN.RM.QUERY.REQ.OWN UACC(READ)'

/* Give RMSADMIN userids access to the DFSMS 1B5 disk                */
'RAC PERMIT DFSMS.1B5 CLASS(VMMDISK) ID('storad_group') ACCESS(READ)'

/* Create generic BRUNO.* entity to control tape usage               */
'RAC RDEFINE FACILITY BRUNO.* UACC(NONE)'
'RAC PERMIT BRUNO.* CLASS(FACILITY) ID('storad_group') ' ,
                   'ACCESS(ALTER)'
```

Figure 26 (Part 2 of 2). RMSRACF EXEC

4.3.7.2 Tape Volume Access Authorization

Access to tape volumes or SCRATCHx pools is through RACF/VM, using a similar principle to that described in 4.3.7.1, “RMSMASTR Command Authorization” on page 49. A generic BRUNO.* profile is defined, without any global access. RMSADMIN userids are given ALTER access over this profile.

When tape volumes are inserted into the 3495, they should be allocated to one or more systems. Specific profiles of the form BRUNO.60*nnnn* are then defined, where *nnnn* is a 4 digit number from 0000 to 9999. Similarly, specific profiles controlling access to SCRATCHx categories can be defined. Again, RMSADMIN userids are given ALTER access, allowing them to mount the volumes and permit user access to volumes. Figure 27 on page 54 shows the ADDTAPE EXEC (used by IBM Canada) that will issue the proper RACF/VM commands to protect new volumes.

```
/* Add a RACF entity to protect tapes controlled by BRUNO             */
/* By default, users connected to RMSADMIN have access to the tape(s) */

/* Both volume serials and SCRATCHx categories are protected in the   */
/* same way, through specific RACF profiles of the form:              */
/*    BRUNO.volser   or    BRUNO.source_cat                           */

/* Input parameters:                                                  */
/*    volser or source_cat    (ie. 600100, SCRATCHD)                  */
/*    owner                   (default: the RMSADMIN group)           */

trace e

  Parse UPPER ARG tape_id owner .
  If tape_id = ''
    Then Signal NoTape_Id
  If owner = ''
    Then owner = 'RMSADMIN'

  Call RACF 'RDEFINE FACILITY BRUNO.'tape_id 'OWNER('owner') UACC(NONE)'
  rc = result
  If rc > 4
    Then Signal Done
  Call RACF 'PERMIT BRUNO.'tape_id 'CLASS(FACILITY) ID(RMSADMIN)' ,
            'ACCESS(ALTER)'
  rc = result

Done:
  Return rc


RACF: Procedure
  Parse UPPER ARG racf_cmd
  'RAC' racf_cmd
  If rc ¬= 0
    Then Say 'Error' rc 'from "'racf_cmd'"'
  Return rc

NoTape_Id:
  Say 'A tape volser or source category must be specified'
  Return 28
```

Figure 27. ADDTAPE EXEC

Once the specific profiles have been defined, RMSADMIN userids can allow other users to gain access to tapes by using the RACF PERMIT command. Figure 28 shows the two RACF commands: the first command defines protection for an entity named SCRATCHD (which will be used to refer to all scratch volumes in the SCRATCHD 3495 category); the second command defines protection for an entity 600096 (which will be used to protect a single volume, 600096).

```
RAC PERMIT BRUNO.SCRATCHD CLASS(FACILITY) ID(3495U01 ..) ACCESS(UPDATE)
RAC PERMIT BRUNO.600096   CLASS(FACILITY) ID(DSSERV) ACCESS(UPDATE)
```

Figure 28. RACF Commands to Permit Access to a Volume

Using the RACF definitions in Figure 28, any user belonging to the 3495U01 RACF group will be allowed to mount volumes assigned to the SCRATCHD category. Similarly, users belonging to the DSSERV RACF group will be allowed to mount volume 600096. Readers familiar with ADSM or WDSF will recognize DSSERV as the default name for an ADSM or WDSF server virtual machine.

4.3.7.3 RACFAUTH ASSEMBLE

Figure 29 shows the RACFAUTH ASSEMBLE file. The RACFAUTH ASSEMBLE file is used to create a module that calls RACF for cartridge authorization checks.

At present, the RACFAUTH MODULE does not distinguish between read (corresponding to R/O) or update (corresponding to R/W) access.

```
RACFAUTH TITLE 'RACFAUTH - ISSUE RACROUTE AUTH TO VALIDATE TAPE ACCESS'
         USING NUCON,R0
RACFAUTH CSECT
         STM   R14,R12,12(R13)     SAVE CALLERS REGISTERS.
         LR    R11,R13             SAVE CALLERS SAVEAREA ADDRESS.
         USING RACFAUTH,R15        USE R15 BASE FOR A NANOSECOND.
         LR    R12,R15             ALIGN OURSELVES.
         USING RACFAUTH,R12        LOAD OUR BASE AND SAVEAREA PTR ...
         B     START
         DROP  R15                 ENOUGH WITH TEMPORARY BASE.
SAVEAREA DS    18F                 ... AND BRANCH AROUND IT.
START    ST    R13,8(0,R11)        GIVE CALLER OUR SAVEAREA ADDRESS.
         ST    R11,SAVEAREA+4      SAVE CALLERS SAVEAREA ADDRESS.
         ST    R1,SAVEAREA         SAVE PARM LIST POINTER, TOO.
         LA    R13,SAVEAREA        POINT TO MY SAVEREA
         LR    R2,R1               PARM LIST POINTER
         B     CHKPARMS
         SPACE 3
EXIT     L     R13,SAVEAREA+4      LOAD CALLERS SAVEAREA ADDRESS.
         L     R14,12(0,R13)       LOAD ALL OF CALLERS REGISTERS ...
         LM    R0,R12,20(R13)      EXCEPT R15 (RETURN CODE).
         LTR   R15,R15             SET COND CODE.
         BR    R14                 RETURN TO CALLER.
         B     EXIT                       ALL DONE
         SPACE 3
```

Figure 29 (Part 1 of 2). RACFAUTH ASSEMBLE File

```
CHKPARMS DS    0H
         MVC   USERID,8(R2)        Save userid
         MVI   VOLID,C' '          Clear VOLID
         MVC   VOLID+1(L'VOLID-1),VOLID
         MVC   VOLID,16(R2)        Save volid or source category
         SPACE 1
         RACROUTE REQUEST=AUTH,    REQUEST AUTH CHECK                $
               RELEASE=1.9,                                          $
               CLASS='FACILITY',     FACILITY class                  $
               ENTITYX=ENTITY,       Point to BRUNO.volid            $
               USERID=USERID,        FOR TARGET USER                 $
               WORKA=WORKA
         SPACE 1
         L     R0,4(,R1)           GET RACF REASON CODE
         SLL   R0,16               MOVE INTO HIGH ORDER HALFWORD
         OR    R15,R0              R15 = AL2(REASON) AL2(RETURN)
         B     EXIT
         DROP  R12
         EJECT 1
         LTORG
         DC    0D'0'
USERID   DC    CL8' '              Userid requesting tape mount
ENTITY   DC    AL2(ENTITYL)        Length of the buffer
         DC    AL2(0)              Let RACF figure out length of name
PREFIX   DC    C'BRUNO.'           RACF entity prefix
VOLID    DC    CL8' '              Volid or SCRATCHx category
ENTITYL  EQU   *-PREFIX
         SPACE 3
WORKA    DS    512C
         SPACE 3
CC0      EQU   8
CC1      EQU   4
CC2      EQU   2
CC3      EQU   1
         SPACE 3
         REGEQU                EQUATE THOSE REGISTERS ... STANDARDLY.
         NUCON
         END   RACFAUTH
```

Figure 29 (Part 2 of 2). RACFAUTH ASSEMBLE File

4.3.7.4 Assigning Volumes to a System

When volumes are assigned to a system, the recommended steps are:

1. Create RACF-specific profiles using the ADDTAPE EXEC.
2. Add the volumes to FSMRMSHR CTLDATA (in VMSYS:DFSMS.CONTROL), checking for any multisystem access conditions.
3. Use the DFSMSRM SET VOLCAT command to assign the volumes to a volspecific category
4. Issue the appropriate RACF PERMIT commands to give userids access to the volumes.

Volumes that will be accessed by volid should be assigned to the volspecific category to prevent a request for a SCRATCHx cartridge from accidentally using a volume assigned to an application on the same or another system.

4.4 BRUNOMNT: A Small Tape Mount Manager

To implement the 3495 at IBM Canada, we needed to build our own small tape management system to:

- Share access to a pool of tape drives (from first-level and second-level VM systems)
- Control access to tape volumes
- Simplify user and server interaction with the 3495
- Track mount events and errors and provide remote operations support.

4.4.1 Tape Drive Sharing

On VM/ESA systems, 3490 drives are assigned (dedicated) to a processor when the drive is attached to a user using the ATTACH command rather than when the drive is brought online using the VARY command. This implementation allows multiple processors to share access to tape drives that are cabled to each processor, by attaching and detaching the drives as required.

There is no direct way of determining to which processor a tape drive is assigned. However, an attempt to attach a tape that has been assigned to another processor results in the message “tape drive is assigned elsewhere.” BRUNOMNT selects a tape drive by attaching each available drive in turn. When BRUNOMNT successfully attaches a drive, BRUNOMNT sends a DFSMSRM MOUNT request to RMSMASTR. RMSMASTR detaches the drive from BRUNOMNT and executes the mount request.

The situation is more complicated on second-level VM systems. Even though the tape drives can be attached to the guest with the NOASSIGN option, it is not easy to attach the same drive to multiple guests. On guest systems, BRUNOMNT communicates with BRUNOGTA (see 4.5, “BRUNOGTA: Real Tape Manager” on page 59) to allow tape drive sharing.

4.4.2 Controlling Access to Volumes

BRUNOMNT provides a single point of control for all DFSMSRM MOUNT requests. It:

- Accepts various forms of mount requests through CP messages from users and selected servers (currently VMBARS, WDSF, and ADSM)
- Verifies access to the requested physical volume or SCRATCHx category through RACF
- Allocates a tape drive, assuming that the maximum number of active tapes has not been reached
- Issues the mount request to RMSMASTR
- Logs all events and errors.

4.4.3 MOUNT EXEC Syntax

The DFSMS/VM mount command was considered nonintuitive by the staff at IBM Canada. BRUNOMNT allows users and applications to request tape mounts by using either the CP message command (MSG *userid*) or the CMS TELL command, insulating users from the format of the DFSMSRM MOUNT command. Mount commands can be issued by any user or server specifying a target userid. All authority checking will based on the target userid.

The BRUNOMNT EXEC accepts input messages in the format natural to the executing program or end user. Therefore, no changes are necessary to

VMBARS, WDSF, or ADSM messages. The authorized virtual machines and corresponding mount messages are:

VMBARS HDFxxxxxxxx Mount *volid vdev mode* for user *userid*

volid may be a volume serial, or the DBR category BRUDBR. BRUDBR requests are mapped to source category SCRATCHD and target category SCRATCHE. The *mode* is either R/W or R/O.

WDSF ANS070xI 001: Mount CARTRIDGE *volid* at *vdev mode* ....

ADSM ANR521xI 001: Mount CARTRIDGE *volid* at *vdev mode* ....

The *mode* is either R/W or R/O.

user MOUNT *cartridge vdev mode* <(<FOR *user*> <RDEV *drive*> <TARGET *category*> >

cartridge may be a volid or a SCRATCHx category. *drive* can be either a 'ccuu' specification or a 3495 drive designator (for example, 101). *vdev* defaults to 181. The *mode* may be R/O or R/W and defaults to R/O.

4.4.4 BRUNOMNT Events

BRUNOMNT is an event-driven application. Events include:

timer expiration waiting for real device allocation, checking mount completion, returning free devices to hypervisor pool

IUCVMSG messages (mount requests, BRUNOMNT commands, communication with hypervisor)

IUCVMSG CP output (tape ATTACH by RMSMASTR, tape returned by user)

console exit from application

4.4.5 Overall BRUNOMNT Flow

The following list gives details of the programming logic in the BRUNOMNT EXEC:

1. A user or server issues a mount request.
2. BRUNOMNT validates the syntax of the mount request.
3. BRUNOMNT checks to see whether a BRUNO drive is already attached to that user at the request (or default) virtual device address. If so, BRUNOMNT detaches the drive from the user.
4. BRUNOMNT uses the RACFAUTH MODULE to determine whether the user is authorized to mount the volid or SCRATCHx category.
5. If the maximum number of active tape drives has not been exceeded, BRUNOMNT searches for the first available (or user-specified) tape drive. BRUNOMNT attaches the drive to itself, to ensure that it is available . If none is found, BRUNOMNT waits for two 30-second intervals for a drive to become available. On second-level systems, BRUNOMNT requests a drive from BRUNOGTA on the hypervisor.
6. BRUNOMNT issues an asynchronous CSL MOUNT request to RMSMASTR, specifying BRUNOMNT as the target user.
7. If the initial MOUNT request was accepted, BRUNOMNT periodically polls RMSMASTR for MOUNT completion.

8. When RMSMASTR attaches the tape drive, BRUNOMNT checks completion status and then gives the drive to the requesting userid using the GIVE command with the RETURN option.
9. When the user detaches the drive, it is returned to BRUNOMNT. BRUNOMNT will wait about 10 seconds before detaching the drive and returning it to the tape pool. This prevents problems with reattaching the drive while it is still being rewound.
10. On second-level systems, BRUNOMNT waits for 180 seconds. If the drive has not been reselected within that time period, BRUNOMNT returns the drive to BRUNOGTA on the hypervisor.

4.4.6 Availability of BRUNOMNT Code

The ***sample*** BRUNOMNT EXEC is available from your IBM representative, who should request BRUNOMNT PACKAGE from the internal marketing tools repository. Alternatively, you or your IBM representative can contact Norbert Hoeller at NORBERTH@VNET.IBM.COM or Craig Welch at CW1313@STLVM4.VNET.IBM.COM and request the BRUNOMNT PACKAGE.

4.5 BRUNOGTA: Real Tape Manager

BRUNOGTA is a server running on any first-level system supporting second-level VM/ESA guests. BRUNOGTA controls tape drive allocation for the second-level guests. When BRUNOMNT on one of these guests requires an additional tape drive, it sends a message to BRUNOGTA on the hypervisor requesting allocation of a drive. BRUNOGTA selects an available drive, attaches it to the guest, and notifies BRUNOMNT of the address. BRUNOMNT can now complete processing of the MOUNT request.

When the user detaches the drive, BRUNOMNT (on a second-level system) will hold the tape, in case the user requests another mount. Once a predefined time interval has expired without any more mount requests for that drive, BRUNOMNT will VARY OFF the drive and notify BRUNOGTA that the tape drive can be returned to the hypervisor. BRUNOGTA detaches the tape from the guest, returning the drive to the pool.

4.5.1 BRUNOGTA EXEC Details

The BRUNOGTA EXEC is a modified copy of BRUNOMNT. It runs on two hypervisor systems and provides guest tape allocation support to the second-level guests. Currently, it supports the following commands:

- ALLOCATE *ANY*
- ALLOCATE *ccuu*
- RETURN *ccuu*.

Allocate requests are queued for 20 seconds, in case a tape frees up. Return requests result in the tape being detached from the guest. The tape will show as BOXED on the guest.

Authorization is limited to requests from BRUNOMNT on the second-level VM systems as well as a limited number of authorized first-level system userids.

4.5.2 Availability of BRUNOGTA Code

The BRUNOGTA EXEC is available from your IBM representative as part of the BRUNOMNT PACKAGE, who should request the BRUNOMNT PACKAGE from VMTOOLS. Alternatively, you or your IBM representative can contact Craig Welch at CW1313@STLVM4.VNET.IBM.COM or Norbert Hoeller at NORBERTH@VNET.IBM.COM and request the BRUNOMNT PACKAGE.

Appendix A. VSE Automation

IBM VM/VSE Solution Centre has announced the *IBM CMS/VSE System Scheduler*, which enables VM/VSE shops to automate VSE guest system operation.

Among the scheduler functions are:

- Automated time- and/or calendar-controlled job submission and surveillance
- Automated message response handling
- Automated time- and/or calendar-controlled initiation and shutdown of CICS environments and networks (VTAM)
- Automated journaling of system activities
- VSE systems may be operated from pseudo-consoles under VM/CMS
- A variety of expansion and maintenance facilities, including user exits for REXX programs.

Automatic batch job submission, console commands, and messages to respond to are easily defined through a user friendly interface with online help facilities. For example, a batch job can be set up to run every Tuesday and Thursday at 11 pm, or only on weekends, or once a month. To handle situations where the submission of a batch job requires other jobs to have finished (for example, job A and C must run before job B), a prerequisite-fulfilment-checking facility is offered.

Besides the ability to schedule batch jobs and VSE console commands, special messages to other CMS users and EXECs can be scheduled for execution with the same time- and calendar-control system.

Also included is a CMS/VSE console interface that allows multiple copies of the actual VSE console to be displayed by any authorized CMS userid. The CMS/VSE console interface offers time and date stamping, redisplay, optional coloring and suppression of VSE partitions, and PF-key support. XEDIT commands can be used to manipulate the display.

The *IBM CMS/VSE System Scheduler* runs under VM/CMS as a number of service machines including a *monitor* for each VSE system and a common *scheduler and message handler*.

A *Production Planning Interface* can be is available. This interface greatly facilitates the maintenance of an automated VSE operating environment.

The *IBM CMS/VSE System Scheduler* can exploit the VM/ESA′s native library control for the 3494 or 3495 Tape Library Dataserver to automate tape mounts for the VSE guest. Installation-specified mount statements appearing on the VSE console are recognized and forwarded to DFSMS/VM service machines for mount handling. These mount statements may be messages from a tape management product, JCL statements, or even JCL comment lines; whatever an operator responds to in a manual tape environment.

Software requirements are:

- VSE/ESA (at any level) as a guest of VM/ESA.

- VSE hard copy files must reside on count key data (CKD) devices (not fixed block architecture, or FBA).
- VM/ESA 1.2.0 or higher with DFSMS/VM Function Level 221 is required for the 3494 or 3495 automation support.

Appendix B. Special Considerations

This appendix is a collection of miscellaneous notes about IBM 3495 Tape Library Dataserver operation. It is provided as a starting point should you encounter unexpected errors with the interaction of the hardware and software.

B.1 Mount Error Recovery

If a cartridge does not load properly, the library manager typically will not detect the error for 5 to 10 minutes. For example, if the rails on the drives are too tight, the robot may not push the cartridge far enough to cause the drive to pull in the cartridge. The mount will remain in progress until the library manager error recovery directs the robot to “nudge” the tape.

B.2 Pause-Conditions and Handling from RMS

DFSMSRM commands can be issued, yet no action or response appears. One possible cause for this is that the RM_REQUEST_QUEUEING parameter in the control file (DGTVCNTL DATA) is set to YES. This means that the RMSMASTR will send the mount request to the library when the library is in PAUSE mode. The library manager will queue the request until the library is taken out of PAUSE mode, when it should execute.

There are two ways you can handle PAUSE mode: You can leave things as they are. If the library is in PAUSE mode for too long, and you are operating with a mount time limit from a TMS (such as the VMTAPE implementation described in Chapter 3, “VM:Tape and IBM′s Tape Library Dataservers” on page 19), the TMS should eventually give up and cancel the mount. In the VM:Tape implementation, this would result in a message being sent to the MOUNTATL EXEC, which will then send a DFSMSRM DISCARD command to the RMSMASTR. The advantage to this approach is that if the library is only in PAUSE mode for a short time, the mounts will probably still complete in time.

The second option is to set the RM_REQUEST_QUEUEING parameter to NO. This will result in RMSMASTR rejecting mount requests when the library is in PAUSE mode. RMSMASTR will issue a message that can be directed to the MOUNTATL EXEC, which can then send a CANCEL operation to VM:Tape. This option is probably more reliable and will execute more quickly.

B.3 Clearing Drives After Shutting Down the Library Manager

If the library manager is shutdown and/or rebooted, all drives must be cleared of cartridges before the library manager is initialized. The cartridges should be placed in the error recovery cells. If this procedure is not followed, the next mount will stay in a blocked state for 5 minutes.

B.4 Using the SET VOLCAT BULK Command for Ejecting Volumes

The SET VOLCAT supports a BULK option, having a format similar to the RMB10023 DATA file. For example, to move all SCRATCHE tapes to the EJECTB (high-capacity output facility), one would expect that the bulk processing file in Figure 30 could be used.

```
600000-609999 EJECTB SCRATCHE
```

Figure 30. Sample Bulk Processing File for Ejecting Volumes

Unfortunately, RMSMASTR generates all alphabetic and numeric combinations in the volid range and queries the library manager on their status. The large majority of the combinations do not exist, causing numerous error messages and significantly impacting throughput. The correct approach is to obtain a list of the specific volids by using QUERY LIBRARY INVENTORY SCRATCHE and then build a tailored bulk processing file or issue individual DFSMSRM SET VOLCAT commands.

Index

Numerics

A

B

C

T

U

V

W

ITSO Technical Bulletin Evaluation RED000

Lights Out! Advanced Tape Automation Using VM/ESA

Publication No. GG24-4347-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

Name _____ Address _____

Company or Organization _____ _____

Phone No. _____ _____

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 471, Building 070B
5600 COTTLE ROAD
SAN JOSE CA
USA 95193-0001

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

GG24-4347-00