z/VM
7.4

DFSMS/VM Removable Media Services

SC24-6278-74

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 145.

This edition applies to version 7, release 4 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2024-09-18

Contents

Figures

Tables

About This Document

This document is a user's guide and reference for DFSMS/VM Removable Media Services (RMS) functions that support the IBM® 3494 Tape Library Dataserver and the IBM 3495 Tape Library Dataserver in a z/VM® environment.

Intended Audience

This document is intended for storage administrators who need to perform Removable Media Services tasks.

RMS functions are a subset of the DFSMS/VM fuctions, and you should be familiar with these DFSMS/VM functions.

Syntax, Message, and Response Conventions

The following topics provide information on the conventions used in syntax diagrams and in examples of messages and responses.

How to Read Syntax Diagrams

Special diagrams (often called *railroad tracks*) are used to show the syntax of external interfaces.

To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►—— symbol indicates the beginning of the syntax diagram.
- The ——► symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The ►—— symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The ——►◄ symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in Table 1 on page xiii.

Table 1. Examples of Syntax Diagram Conventions

Syntax Diagram Convention	Example
Keywords and Constants A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	►►— KEYWORD —►◄

Table 1. Examples of Syntax Diagram Conventions (continued)

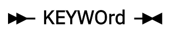
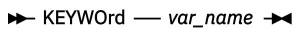
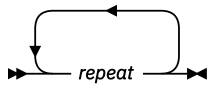
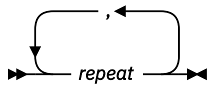
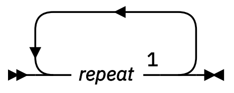
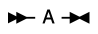
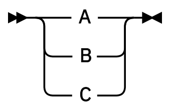
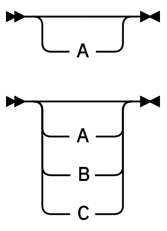
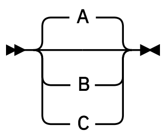
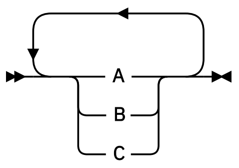
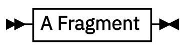
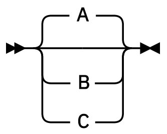
Syntax Diagram Convention	**Example**
Abbreviations Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	
Symbols You must specify these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period
Variables A variable appears in highlighted lowercase, usually italics. In this example, *var_name* represents a variable that you must specify following KEYWORD.	
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	   Notes: [1] Specify *repeat* up to 5 times.
Required Item or Choice When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	 

Table 1. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
Optional Item or Choice When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	
Defaults When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragment Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 **A Fragment** 

Examples of Messages and Responses

Although most examples of messages and responses are shown exactly as they would appear, some content might depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

xxx
: Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.

[ ]
: Brackets enclose optional text that might be displayed.

{ }
: Braces enclose alternative versions of text, one of which will be displayed.

|
: The vertical bar separates items within brackets or braces.

...
: The ellipsis indicates that the preceding item might be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, might be repeated.

Where to Find More Information

See “Bibliography” on page 149 at the back of this document.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of Changes for z/VM: DFSMS/VM Removable Media Services

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

SC24-6278-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

SC24-6278-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6278-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

SC24-6278-01, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

Part 1. User’s Guide

Chapter 1. Introducing DFSMS/VM Removable Media Services

Support for the IBM 3494 and 3495 Tape Library Dataservers in the Virtual Machine (VM) environment is provided by the DFSMS/VM® Removable Media Services (RMS). RMS includes a high-level interface to 3494 and 3495 functions for use by VM users and tape applications.

The 3494 and 3495 functions, accessible to the user as both interactive commands and programming interfaces, are handled by a dedicated DFSMS/VM service machine, the RMS master. The RMS master is an application authorized to use the 3494 or 3495.

This chapter helps you understand:

- RMS functions in relation to DFSMS/VM
- Hardware and software requirements for RMS
- Volume management support
- Interactions RMS can have with a tape management system (TMS)
- Interactions RMS can have with end-users

RMS Functions in Relation to DFSMS/VM

Figure 1 on page 3 shows DFSMS/VM's four major functional components: user interface, minidisk management, space management, and Removable Media Services. The RMS interactive commands and CSL programming interface are included in the user interface component and RMS master support is provided by the Removable Media Services component.

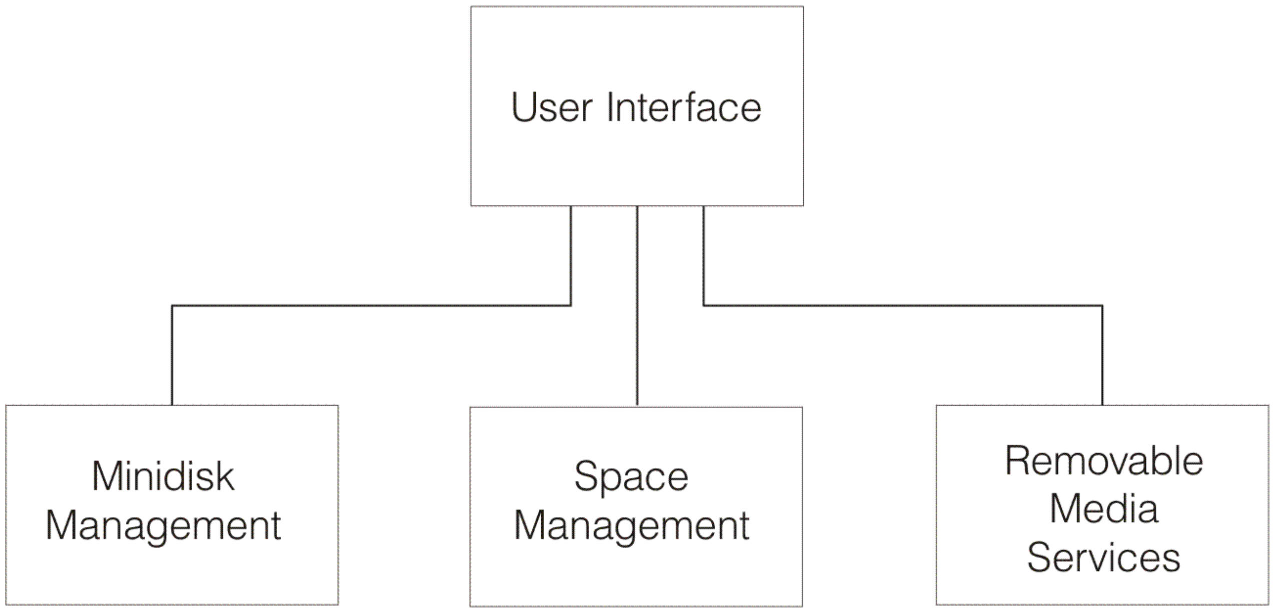


Figure 1. Overview of DFSMS/VM

Together, the 3494 and 3495 Tape Library Dataservers and RMS provide several important installation advantages. The 3494 and 3495 improve productivity of installation-support personnel and reliability and response-time for end-users, while RMS provides an easy-to-use, high-level interface to end users for accessing and managing library volumes in the 3494 and 3495.

Hardware and Software Requirements

A single RMS master can support multiple 3494s or 3495s attached to a VM system. Optionally, multiple RMS masters, each supported by its own VM system, can use a single 3494 or 3495 concurrently. In order for RMS to interact with your 3494 or 3495 in any way, you must have the appropriate hardware and software support. This support is detailed below.

Hardware Support

RMS interacts with the IBM 3494 and 3495 Tape Library Data Servers and their supported tape subsystems including 3490, 3490E, Magstar® 3590, and Magstar Virtual Tape Subsystem (VTS).

Note: When a VTS is installed in a 3494 in combination with other tape subsystems, the VTS is defined to RMS with its own unique library sequence number. Define the virtual 3490E addresses to z/VM and in the RMS configuration file; the host software is not aware of the actual 3590 drives in a VTS subsystem.

Subsystem configurations, cartridge capacity, and other attributes of the 3494 and 3495 libraries and their associated tape subsystems are available in the hardware manuals listed in "Bibliography" on page 149.

Attention: Be careful when mixing incompatible drives within the same hardware library. If such a mixture should occur, either the tape librarian product or an RMS exit would need to ensure that a tape mount request uses a drive which supports the media appropriate for that request.

Peer-to-Peer (PtP) and TS7700 Support

IBM TotalStorage™ Peer-to-Peer Virtual Tape Server (PtP VTS) is used for data backup and recoverability capabilities by providing dual-volume copy, remote functionality, and automatic recovery and switchover, where PtP VTS is fully supported by VM for guest exploitation; RMS supports PtP as a single node VTS. PtP VTS no longer supports specifying a target category on a library mount (DFSMSRM mount) or demount, as well as no longer supporting the use of the insert category as a source category on a mount or set volume category. In addition, TS7700 configurations do not support a target category change while the volume is mounted. RMS will continue to fully support target and source categories for those ATLs that still do.

Software Support

You must have the following minimum level of IBM software programs installed to use RMS.

- DFSMS/VM Function Level 221—supports Removable Media Services
- z/VM Version 6 Release 2—supports the 3494 or the 3495
- SFS (Shared File System)—supports creation and sharing of files used by RMS

Note: If you are using RMS support to enable communication between another host and the 3494 or 3495, you must install both SAA® AD/CYCLE® Language Environment® and VM TCP/IP, both included with z/VM. See Appendix B, "3494 and 3495 Control for Foreign Processors," on page 95 for details.

IBM Virtualization Engine TS7700 Copy Export Support

z/VM provides support within the Removable Media Services (RMS) component of DFSMS/VM for the IBM Virtualization Engine TS7700 Copy Export functionality. This function allows a copy of selected logical volumes written to the TS7700 to be removed and taken offsite for disaster recovery purposes. The benefits of volume stacking, which places many logical volumes on a physical volume, are retained with this function. In addition, since the data being exported is a copy of the logical volume, the logical volume data remains accessible by the production host systems.

Volume Management Support

To manage devices and volumes[1] associated with a 3494 or 3495, RMS relies on the capabilities of the library manager. The library manager controls all operations within the 3494 or 3495 library, including the movement of volumes, in response to commands from the RMS master or the 3494 or 3495 library operator.

To aid in volume management, the 3494 and 3495 support logical groupings of volumes known as categories. RMS supports special categories for use as scratch pools. With the 3495, a specific scratch pool can be assigned for automatic volume loading on one or more devices to enhance mount performance. Additionally, category designations are used to control movement of volumes in and out of the 3494 or 3495.

Chapter 3, "Using Removable Media Services," on page 17 gives detailed information on volume management tasks performed by RMS, such as:

- Accessing volumes in the 3494 or 3495
- Utilizing RMS categories
- Moving volumes in and out of the 3494 or 3495

Interactions RMS Can Have with a Tape Management System

RMS is designed to work with a tape management system (TMS) machine. In a typical usage scenario, TMS issues requests to the RMS master on behalf of an end-user. Figure 2 on page 5 shows the typical interaction involving a TMS machine and RMS.

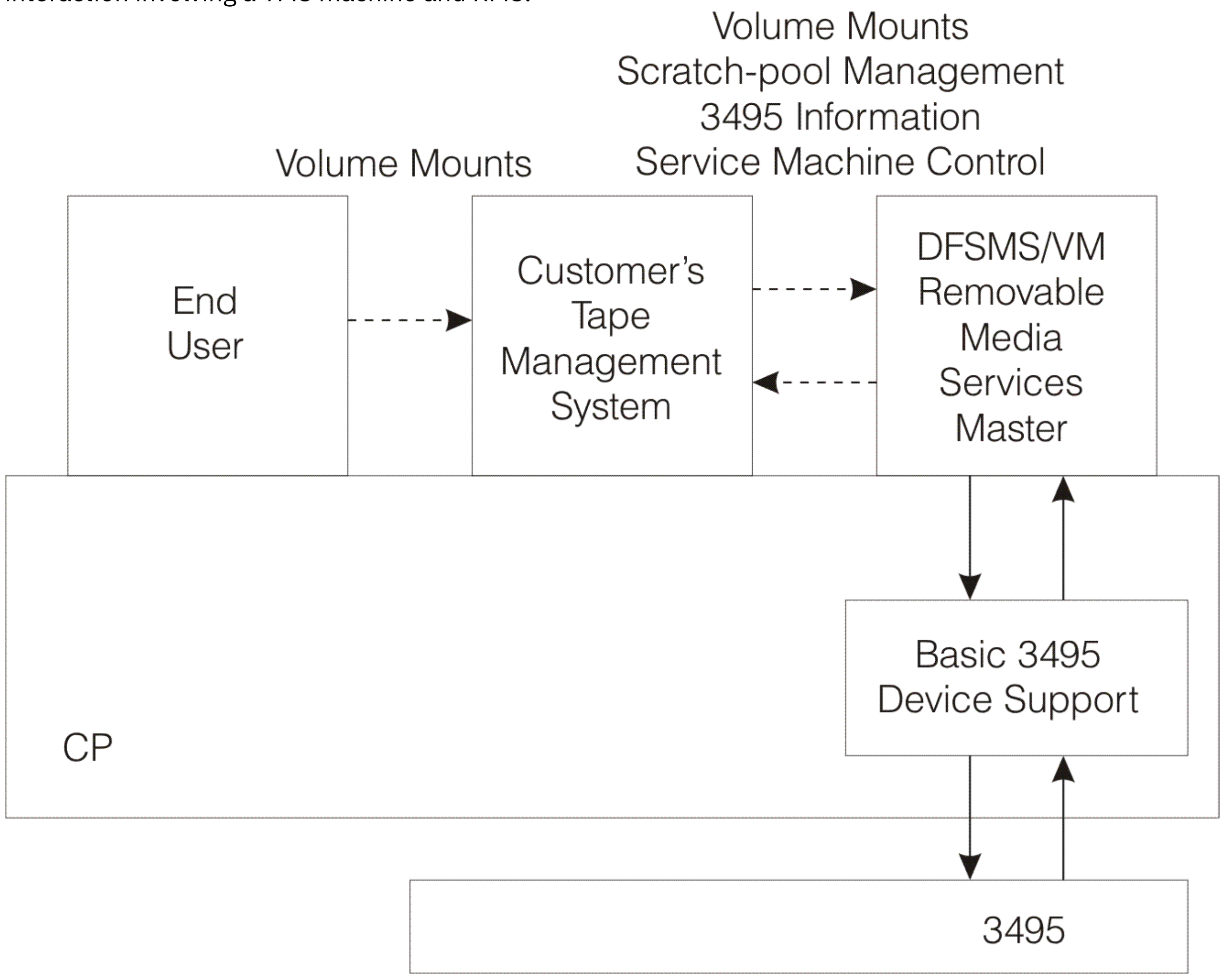


Figure 2. Typical Interaction with a Tape Management System

[1] The term *volume* is used to describe a certain portion of data, together with its data carrier, that can be mounted on the system as a unit. When this term pertains to the 3494 or 3495, it is synonymous with *tape cartridge*.

Interaction with RMS can be handled either by using the RMS subset of the DFSMS/VM command set, or by using the application programming interfaces (APIs), which are provided as a set of callable services library (CSL) routines.

Requests issued by a TMS machine are received, processed, and responded to by the RMS master. When an end-user issues a mount request to the TMS, the TMS:

1. Selects an appropriate device
2. Performs volume-access-authority checking (prior to issuing the mount request)
3. Performs internal label verification (when the mount request is completed, but before attaching the device to the end-user)

The tape librarian associated with the TMS machine performs inventory management functions, such as assignment of volumes to scratch pools. These functions are requested either through the same TMS virtual machine that issues volume mounts or through a different virtual machine.

The TMS machine provides functions not provided by the RMS, such as:

- Maintaining a removable media inventory
- Performing volume label verification
- Performing authorization access checks at the volume level
- Performing device selection sensitive to media type

Interactions RMS Can Have with End-Users

RMS provides an alternative approach for customers without the full-function tape management support described above.

Figure 3 on page 7 illustrates virtual machine interaction without a TMS machine, with end-users interacting directly with the RMS master for mount requests.

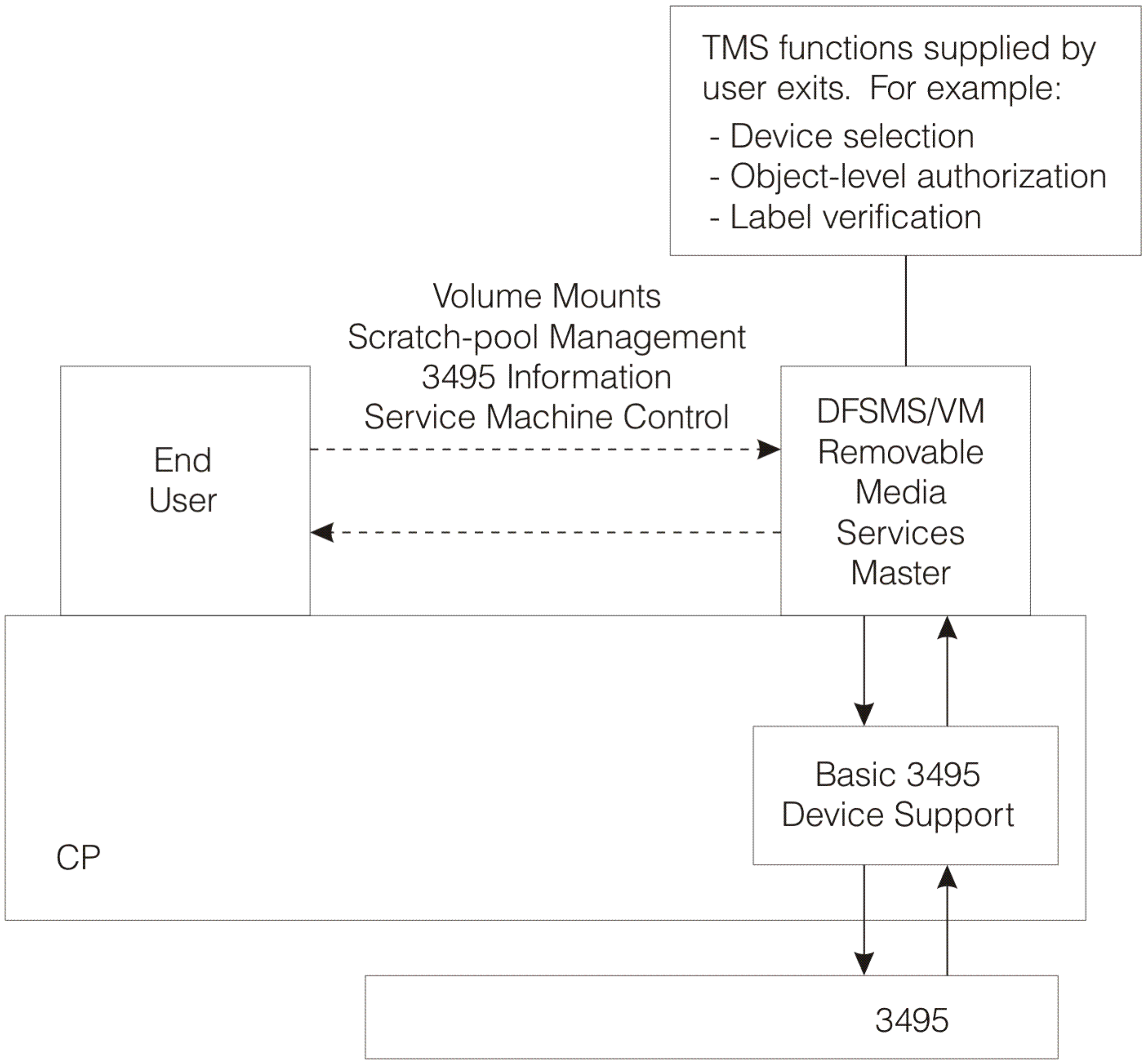

TMS functions supplied by user exits. For example:
- Device selection
- Object-level authorization
- Label verification
Volume Mounts
Scratch-pool Management
3495 Information
Service Machine Control
End User
DFSMS/VM Removable Media Services Master
Basic 3495 Device Support
CP
3495

Figure 3. Alternative Machine Interaction Using Types of Installation-Wide Exit

Several types of installation-wide exit allow you to call tape management functions during request processing, for example:

- The default device selection, which occurs early in mount processing, can be replaced with more sophisticated processing by using the FSMRMDEV installation-wide exit.
- Before the RMS master issues a mount request to the hardware, you can verify user access authorization for the requested volume with the FSMRMPRE installation-wide exit. This installation-wide exit can also be used to call another application to perform a similar function.
- Before the RMS master attaches the device to the requester, you can perform volume label verification with the FSMRMPRO installation-wide exit. This installation-wide exit can also be used to call another application to perform a similar function.

Mount requests are issued by authorized end-users or by tape librarian functions handled by a user specifically authorized for such tasks. However, it is important to remember that RMS is designed to work with a TMS. The RMS types of installation-wide exit are best used for supplementing, not replacing, TMS functions.

For more information about RMS types of installation-wide exit, see Chapter 5, “Customizing Removable Media Services,” on page 29.

Chapter 2. Preparing to Use Removable Media Services

This chapter provides background information and instructions for the following tasks in preparation for RMS use.

- Authorizing your RMS master to interact with the 3494 or 3495
- Defining the 3494 or 3495 resources
  - Defining DFSMS/VM control file parameters
  - Defining devices in the RMS configuration file
- Using standard labeling
- Authorizing users

This chapter also contains information on supporting manual tape libraries.

Authorizing Your RMS Master to Interact with the 3494 or 3495

Because your 3494 or 3495 is in a VM environment, you must authorize your RMS master within the z/VM control program (CP) to interact with the 3494 or 3495. The CP manages system resources, processor functions, processor storage, and the system's input and output (I/O) devices in the VM environment.

You can authorize your RMS master to interact with the 3494 or 3495 by adding the STDEVOPT control statement to the 3494's or 3495's CP directory entry. The STDEVOPT control statement specifies the optional storage device management functions available to a virtual machine. Figure 4 on page 9 shows a sample STDEVOPT control statement authorizing an RMS master to control a 3494 or 3495 and issue tape library control commands.

```
⋮
 STDEVOPT LIBRARY CTL
⋮
```

Figure 4. Sample STDEVOPT Control Statement

For more information on the STDEVOPT control statement, refer to *z/VM: CP Planning and Administration*. For more information on using the STDEVOPT control statement for the RMS master, see *z/VM: DFSMS/VM Customization*.

Note:

If RMS is running on a guest z/VM system, the guest virtual machine must be authorized to interact with the 3494 or the 3495. Include a STDEVOPT statement in the host's directory definition for the guest virtual machine, as well as in the directory for RMS master in the second level guest machine.

Defining the 3494 and 3495 Resources

Once you authorize your RMS master to interact with the 3494 or 3495, you can define the 3494 or 3495 resources with the DFSMS/VM control file and the RMS configuration file, which must reside on the VMSYS:DFSMS.Control SFS directory. Table 2 on page 10 provides an overview of each of these files.

Table 2. Files Used with the RMS master

Type	Name	Purpose
Control file	DGTVCNTL DATA	Defines RMS resources and customizes processing options. Also refer to "Defining the DFSMS/VM Control File Parameters" on page 10.
RMS configuration file	RMCONFIG DATA	Defines your hardware configuration. Also refer to "Defining Devices in the RMS Configuration File" on page 12.

Note: If you wish to have the control and configuration files reside in a different directory, create an entry in the UCOMDIR NAMES file on the RMSMASTR 191 A-disk similar to the following:

```
:nick.VMSYS  :tpn.MYPOOL
```

The following sections give you detailed information on using these files to define 3494 and 3495 resources.

Defining the DFSMS/VM Control File Parameters

RMS processing is supported by the RMS parameters in the DFSMS/VM control file, DGTVCNTL DATA. The control file enables you to define RMS resources and customize processing options.

You must define the RMS master and the 3494s or 3495s it uses, by specifying the following required parameters in the DFSMS/VM control file. Initialization processing fails if these parameters are not specified:

- DFSMSRM_MASTER_VM
- GLOBAL_RESOURCE_ID
- RM_AUTO_LIBRARY

If you want to customize processing for your 3494 or 3495 environment, specify the optional parameters in the DFSMS/VM control file. If you do not specify these parameters, their default values are assumed. The following are optional parameters :

- RM_ACCOUNTING
- RM_DEFAULT_SCRATCH_POOL
- RM_FOREIGN_SERVER_VM
- RM_LOG_TO_CONSOLE
- RM_LOG_TO_FILE
- RM_MANUAL_LIBRARY
- RM_REQUEST_QUEUING
- RM_USE_GIVE
- RM_WRITE_PROTECT

Table 3 on page 11 lists the RMS control file parameters (required and optional) along with the purpose of each. For more information about control file structure and use, including the defaults for parameters, refer to *z/VM: DFSMS/VM Customization*.

Table 3. Control File Parameters for Removable Media Services

Parameter	Purpose	Required
DFSMSRM_MASTER_VM	Identifies which RMS master receives all DFSMS/VM requests for RMS functions. **Note:** This control file parameter is essential; if it is not found, initialization processing fails.	Yes
GLOBAL_RESOURCE_ID	Identifies the name of the global advanced program-to-program communications (APPC) resource by which this DFSMS/VM system is known. **Note:** This is a required keyword-value pair. If omitted, DFSMS/VM does not start.	Yes
RM_AUTO_LIBRARY	Identifies automated 3494s or 3495s to the RMS master. This identification process associates a 3494 or a 3495 with a library name and provides a notify user ID for it. **Note:** You must customize this control file parameter before you initialize the RMS master. Initialization processing fails if at least one occurrence of this parameter is not found. The first occurrence of this parameter becomes the default library.	Yes
RM_ACCOUNTING	Specifies whether or not the DFSMS/VM service generates accounting records for RMS processing.	No
RM_DEFAULT_SCRATCH_POOL	Specifies the name of the scratch pool treated as the default scratch pool.	No
RM_FOREIGN_SERVER_VM	Specifies the name of a service machine autologged during RMS master initialization to enable support for non-VM hosts that request 3494 or 3495 functions.	No
RM_LOG_TO_CONSOLE	Specifies the severity level of the messages logged to the RMS master.	No
RM_LOG_TO_FILE	Specifies the severity level of the messages written in a log file. This parameter enables you to create detailed activity logs for audit purposes.	No
RM_MANUAL_LIBRARY	Specifies a manual 3494 or 3495 and a user ID to receive mount requests that specify this manual 3494 or 3495. Use this parameter once for each manual library that is to be known to RMS processing.	No

Table 3. Control File Parameters for Removable Media Services (continued)

Parameter	Purpose	Required
RM_REQUEST_QUEUING	Specifies whether or not MOUNT and DEMOUNT operations are queued by the library manager component if the library is in pause mode.	No
RM_USE_GIVE	Specifies whether to use CP GIVE or CP ATTACH/DETACH to provide the tape device to the target user for any request that specifies device attachment upon completion.	No
RM_WRITE_PROTECT	Sets the function default for READOnly/ READWRite options. Thus, for MOUNT requests in which neither READOnly nor READWRite is specified, the default value specified with this parameter is in effect.	No

Note: A *Yes* in the “Required” column indicates that you must specify your options; no defaults are assigned. A *No* in the “Required” column indicates that you can choose not to specify any options; default processing will occur.

Defining Devices in the RMS Configuration File

You must define which devices can be used for RMS processing in the RMS configuration file, RMCONFIG DATA. RMCONFIG DATA lists devices associated with all 3494s or 3495s connected to the VM system. Each noncomment entry in this file is a single device address, or a range of addresses. When a Virtual Tape Server is present, these addresses describe the virtual tape drives.

To create the RMS configuration file, RMCONFIG DATA, you must follow the attribute and syntax conventions described in this part of the chapter.

Note: If you provide 3494 or 3495 support for a non-VM system, as described in Appendix B, “3494 and 3495 Control for Foreign Processors,” on page 95, the devices to be used by the other host must be included in the configuration file. These devices are connected to both the VM system and the other host.

File Name, File Type, and File Placement: The RMS configuration file name is RMCONFIG, with file type DATA. You must place this file in the SFS directory, VMSYS:DFSMS.CONTROL, for read access by the RMS master. Edit this file and restart the RMS master to customize the real device addresses of devices in your 3494 or 3495.

Note: Because the RMS master reads the RMS configuration file during initialization, you must reinitialize the RMS master in order for the new modifications to take effect.

General Format: The general format of an RMS configuration file record is:

```
rdev                    * optional comment
```

or

```
fr_rdev-to_rdev         * optional comment
```

Where:

`rdev`
: specifies a real device address

`fr_rdev-to_rdev`	specifies a range of real device addresses

As shown in Figure 5 on page 13, you must provide either a single real device address or a range of addresses in the RMS configuration file. Specify each device address with one to four characters. If you specify a device address as fewer than four characters, it is left-padded with zeros during processing.

When you specify a range of addresses, the starting device address must have a lesser numeric value than the ending device address and the starting and ending device addresses must be delimited with a dash (-), and optionally by one or more spaces. Any information following the first blank character in an RMS configuration file record is ignored.

If you respecify an address that is in the range, you will get an error; duplicate addresses are not accepted.

```
D00                      * ESCON Attach 3490E
D01                      * ESCON Attach 3490E
D02                      * ESCON Attach 3490E
D03-D08
D09-D0A
D0B-D0C
D0D-D0F
BD0-BDF
BE0-BEF
BF0-BFF
```

Figure 5. Example RMS Configuration File

File Record Requirements: The record format must be fixed. The logical record length must be no greater than 240 characters.

Record Structure: Create the RMS configuration file with a single data-record type; do not use keywords. You can start required information for data records in any column but you must delimit the positional parameters by one or more blanks.

Comments: You can start a comment anywhere on the line. However, once the comment indicator is encountered, the remainder of the line is interpreted as a comment. Comments cannot span lines. A comment stops at the end of a line; to continue a comment on the next line, start the next line with an asterisk.

Blank Lines: If you include blank lines in the RMS configuration file, they will be ignored during RMS processing.

Lowercase/Mixed-Case/Uppercase: You can use any combination of letter case for file data and comments. The RMS configuration file is not case-sensitive.

Considerations for a Multiple-Host Configuration

Because devices in a 3494 or 3495 can be connected to multiple host systems and because the configuration may expand, the RMS configuration file defines which devices can be currently specified by the requester, or used by DFSMS/VM when the requester does not specify a device.

IBM recommends that all devices be connected to all attached hosts to have maximum flexibility for device utilization by connected hosts.

Using Standard Labeling

IBM recommends that you use standard volume labels on all of the 3494 or 3495 volumes in the VM environment. The standard volume label is the first file on a volume and can be verified by the TMS machine to ensure it matches the external label on the outside of the volume. This process provides a high level of security and integrity for data by ensuring the correct volume is mounted.

Also, when you use a standard volume label convention for both VM and MVS™, a 3494 or 3495 can be more readily shared by both systems.

Use of standard volume labels is not a prerequisite for using the RMS master.

Authorizing Users

In addition to the RMS master being authorized by means of a CP directory option, further levels of 3494 or 3495 authorization checking and security for request processing, similar to those available with other DFSMS/VM services, are offered by RMS. These levels of authorization checking are:

- RACF®/VM
- DFSMS/VM authorization file
- Installation-defined authorization

The following sections provide information about each of these levels of authorization checking. For additional information about security and authorization checking for DFSMS/VM, refer to *z/VM: DFSMS/VM Customization*.

Using RACF/VM to Authorize Users

Set up the RMS master to use the RACROUTE interface. By using procedures similar to those for other DFSMS/VM virtual machines, you can connect machines (such as a TMS librarian) to the STAGADMIN group or an alternate user-defined group. With RACF/VM, you can authorize user IDs for command level entities. Table 4 on page 14 describes the additional RACF/VM entities for DFSMS/VM.

Table 4. RACF/VM Entities for 3495 Support Functions

Command	CSL Routine	RACF/VM Entity
DFSMSRM DEMOUNT	FSMRMDMT	STGADMIN.RM.DEMOUNT
DFSMSRM DISCard (user request ID)	n/a	STGADMIN.RM.DISCARD.OWN
DFSMSRM DISCard (any request ID)	n/a	STGADMIN.RM.DISCARD.ANY
DFSMSRM MOUNT	FSMRMMNT	STGADMIN.RM.MOUNT
DFSMSRM Query LIBrary	FSMRMQLB	STGADMIN.RM.QUERY.LIB
DFSMSRM Query REQuests (* or own)	n/a	STGADMIN.RM.QUERY.REQ.OWN
DFSMSRM Query REQuests (ALL, FOR, or other user's)	n/a	STGADMIN.RM.QUERY.REQ.ANY
DFSMSRM RESET DEVCAT	FSMRMRDC	STGADMIN.RM.RESET.DEV
DFSMSRM SET DEVCAT	FSMRMSDC	STGADMIN.RM.SET.DEV
DFSMSRM SET VOLCAT	FSMRMSVC	STGADMIN.RM.SET.VOL
DFSMSRM SET VOLCAT BULK	FSMRMSVB	STGADMIN.RM.SET.VOL.B
DFSMSRM STOP	n/a	STGADMIN.RM.STOP

Note: Commands that have an *n/a* designation in the "CSL Routine" column do not have CSL routine equivalents.

Using DFSMS/VM Authorization File to Authorize Users

If you do not install RACF/VM, you can use the alternate DFSMS/VM mechanism for authorization control, the DFSMS/VM authorization file. Standard DFSMS/VM authorization usage designates certain commands as end-user commands, which are available to any user who can access the product. The remaining commands are for storage administrator (or librarian) use only and require each authorized user ID to have an entry in the authorization file.

Because the RMS master is designed to work with a tape management system, the standard designation for all DFSMSRM commands is that they are issued by authorized users and are not available to end-users. You have the flexibility of making selected commands available to all users, by modifying the installation-wide exit as described in "Using Installation-Defined Authorization" on page 15.

PI

Using Installation-Defined Authorization

RMS processing calls the installation-wide exit FSMVAUTH, which allows you to apply an alternate security-checking technique instead of using the DFSMS/VM standard mechanisms. This installation-wide exit point precedes the standard processing (RACF/VM or DFSMS/VM authorization file), replacing standard processing when authorization status can be determined and deferring authorization to standard mechanisms if authority cannot be determined. This installation-wide exit can invoke a routine that checks the authorization file for a subset of DFSMSRM commands, for example, all commands except MOUNT. You can make MOUNT or any subset of the RMS commands available to end users; or you can extend MOUNT authorization to a list of end users by using FSMVAUTH.

An additional installation-wide exit, FSMRMPRE, invoked prior to actual command processing (for example, before hardware communication for a mount) can be used for supplementary authorization checking to augment, rather than replace, standard processing. For example, you can use the installation-wide exit to fulfill requirements for object-level authorization checking.

With removable media, security protocols entail object-level authorization. Both RACF/VM and authorization file processing verify the user's authority to issue the command at hand. If your tape management system does not provide object-level authorization, you can use the supplementary installation-wide exit for adding this function.

Note: When a 3495 is accessed concurrently with another operating environment such as MVS, the installation-wide exit FSMRMSHR can be used to verify that library resources (volumes and categories) specified in the request are available for use in the VM environment. Refer to "FSMRMSHR – RMS Library-Partitioning" on page 35 for further details.

Chapter 5, "Customizing Removable Media Services," on page 29 provides details on types of installation-wide exit.

PI end

Supporting Manual Tape Libraries

For volumes in a manual tape library, you can have RMS pass volume mount requests to a designated user ID or operator. Support for manual tape libraries is provided by identifying manual removable media libraries in the DFSMS/VM control file. You can use the DFSMSRM MOUNT command as the standard MOUNT command for removable media located in any library, automated or manual, that is attached to the system.

Stand-Alone Tape Support

Stand-alone tape support is not provided on 3494 or 3495 devices in automated mode in DFSMS/VM Function Level 221. For stand-alone operation, the 3494 or 3495 must be in manual mode.

Chapter 3. Using Removable Media Services

You can use RMS to control the 3494 or 3495 with DFSMSRM commands and CSL routines. By issuing these commands and routines, you can perform several important tasks that integrate 3494 or 3495 functions into your tape management environment.

This chapter provides general information about the following tasks and lists the commands and CSL routines used to perform these tasks:

- Control operation of the RMS master
- Access volumes in the 3494 or 3495
- Utilize DFSMS/VM RMS categories
- Move volumes in and out of the 3494 or 3495
- Assign volumes to scratch pools
- Request information from the 3494 or 3495 and the RMS master
- Log DFSMS/VM and 3494 or 3495 messages
- Generate accounting records

For specific information about DFSMSRM commands, refer to Chapter 6, "DFSMSRM Commands," on page 39. For specific information about CSL APIs, refer to Chapter 7, "RMS Callable Services Library (CSL) Routines," on page 59.

Note: Commands and function code words are presented in both upper and lower case format, where the upper case characters represent the minimum set of characters acceptable to RMS.

Controlling Operation of RMS Master

To control RMS processing, use the following DFSMSRM commands and CSL routine:

DISCard
allows an authorized user to cancel certain DFSMSRM requests that have been accepted by RMS for processing. For more information on the DISCard command, see "DFSMSRM DISCard" on page 42.

STOP
allows an authorized user to shutdown the RMS master. Optional parameters determine if shutdown is immediate. For more information on the STOP command, see "DFSMSRM STOP" on page 58.

FSMRMMNT CAN
allows an authorized user to cancel a specific mount request. For more information on the CSL routine FSMRMMNT, see "FSMRMMNT – Library Mount" on page 67.

Accessing Volumes in the 3494 and the 3495

RMS provides functions that not only enable you to mount (and demount) volumes onto devices, but also enable you to designate which device is used.

The following two sections give background information and explain which DFSMSRM commands and CSL routines you use to perform mount and demount volumes and to designate a target device.

Mounting and Demounting Volumes

RMS provides functions that enable you to mount and demount library volumes onto devices defined in the 3494 or 3495 configuration. Mount functions are provided by the DFSMSRM MOUNT command and the FSMRMMNT CSL routine. Demount functions are provided by the DFSMSRM DEMOUNT command and the FSMRMDMT CSL routine.

As part of processing of the DFSMSRM MOUNT command, the RMS master automatically demounts any volume mounted on the device before mounting the requested volume. As a result, you may find that you do not normally need to use the DFSMSRM DEMOUNT command. However, if you wish to mount a volume that is currently mounted on a different device, you can use the DFSMSRM DEMOUNT command to demount the volume from the device on which it is currently mounted and make it available for mounting on your device.

For information on MOUNT and DEMOUNT commands, see “DFSMSRM MOUNT” on page 43 and “DFSMSRM DEMOUNT” on page 40. For information on the CSL routine equivalents to these commands, see “FSMRMMNT – Library Mount” on page 67 and “FSMRMDMT – Library Demount” on page 65.

Designating a Target Device

MOUNT, DEMOUNT, and SET DEVCAT functions entail interaction with a specific tape unit. Library control communications is accomplished on the drive path for the designated target device.

To ensure that the selected device type is appropriate for the media involved, the request issuer can designate a target device for the MOUNT and SET DEVCAT functions. If a target device is not specified, the installation-wide exit FSMRMDEV (real device selection) can be used to locate an available (unattached) device. If the exit takes no action, RMS will attempt to find an available device.

Although QUERY LIB and SET VOLCAT requests entail communication with the 3494 or 3495 library manager rather than action directed at a library device, virtual machine communication still requires a library drive path. Any library drive can meet this need. If the requisite software is installed for enhanced library control, a free drive is not required to provide an available channel path; CP manages drive selection that is transparent to the requester.

When the requisite software is not installed, a free drive must be available. This drive is designated by the requester, supplied by FSMRMDEV exit, or located by RMS. A real drive designated by the requester may be ATTACHed to the requesting virtual machine.

The following attach protocols are observed:

- If enhanced library control is available and the request is a SET VOLCAT or QUERY LIB (other than DEVice or VOLume with AUDit), then RMS Master processes the request (no attach or detach of devices occurs).
- If the device specified in a request is currently unattached, then the RMS master:
  - Attaches the device to itself
  - Processes the request
  - Detaches the device
  - Attaches the device to the appropriate user for MOUNT requests
- If the specified device is attached to the command requester, the RMS master:
  - Detaches the device from that user
  - Attaches the device to itself
  - Processes the request
  - Detaches the device
  - Attaches the device to the command issuer for all commands except SET DEVCAT or, if specified in the request syntax, attaches the device to another user ID.
- If the specified device is attached to a user other than the command issuer, the request fails.
- If there is no device specified in the request then:
  - An installation-wide exit, FSMRMDEV, can be used to select a device. Or the RMS master default for device selection is used.

    For more information about the installation-wide exit FSMRMDEV, refer to “FSMRMDEV – Real Device Selection” on page 31.

Utilizing DFSMS/VM RMS Categories

Categories are logical groupings of volumes maintained by the 3494 or 3495 for managing the inventory. As shown in Table 5 on page 19, DFSMS/VM supports up to seven category designations: INSERT, EJECT, EJECTB, SCRATCH*x* (where *x* is 0–9 or A–F), VOLspecific, *hexvalue*, and copy_export.

Table 5. Category Designations Supported by DFSMS/VM

Category	**Description**
INSERT	New library volumes that have not yet been assigned to another category.
EJECT	Volumes designated for removal from the library by means of the convenience output station. Refer to "Moving Volumes In and Out of the 3494 or 3495" on page 19 for a description of the convenience output station.
EJECTB	Volumes designated for removal from the library by means of the high-capacity output station. Refer to "Moving Volumes In and Out of the 3494 or 3495" on page 19 for a description of the high-capacity output station.
SCRATCH*x*	Volumes assigned to scratch pools. Up to 16 scratch pools may be used; where *x* is 0–9 or A–F.
VOLspecific	Volumes that are not part of any logical grouping.
hexvalue	Hexadecimal category in the range 0000 to FFFF.
COPY_EXPORT	Initiates a Copy Export operation which allows a copy of selected logical volumes written to the TS7700 to be removed and taken offsite for disaster recovery purposes.

DFSMSRM commands and CSL routines include operands and options that enable you to specify in which category a volume is placed and from which category a volume is taken. Use a target category operand or option to specify in which category a volume is placed during request processing. Use a source category operand or option to specify from which category a volume is taken when a request is issued. Although source category designations are usually optional, they can prevent unintended category changes.

For detailed information about the INSERT, EJECT, and EJECTB categories, refer to "Moving Volumes In and Out of the 3494 or 3495" on page 19. For more information about the SCRATCH*x* category, refer to "Assigning Volumes to Scratch Pools" on page 20. For more information about copy_export, refer to Appendix E, "RSM Copy Export Support," on page 141.

Moving Volumes In and Out of the 3494 or 3495

You can add volumes to the 3494 or 3495 through the convenience input station and remove them from the 3494 or 3495 through the convenience output station. An optional high-capacity input/output station may be used to add or remove larger quantities of volumes.

Special category names are used by the 3494 or 3495 to move volumes in and out of the library through the convenience input/output stations and optional high-capacity input/output stations. The following category names are reserved for use by the 3494 or 3495 to move volumes in and out of the library:

- INSERT
- EJECT
- EJECTB

When you insert a volume into the library's input station, the 3494 or 3495 automatically assigns it to the INSERT category. Once a volume is in the INSERT category, it remains here until you assign it to another category by one of the functions capable of this task, for example, the SET VOLCAT function.

When you want to remove a volume from the 3494 or 3495, assign it to the EJECT or EJECTB category. Use the EJECT category to move a volume to the convenience output station, or use the EJECTB category to move a volume to the high-capacity output station. When the 3494 or 3495 recognizes that a volume has been assigned to the EJECT or EJECTB category, it places the volume in the appropriate output station. You can assign volumes to the EJECT or EJECTB categories with the SET VOLCAT and SET VOLCAT BULK functions.

For information on the SET VOLCAT and SET VOLCAT BULK commands, see "DFSMSRM SET VOLCAT" on page 55. For information on the set volume category CSL routine, see "FSMRMSVC – Set Volume Category" on page 80. For information on the bulk processing CSL routine, see "FSMRMSVB – Set Volume Categories Using Bulk Processing" on page 83.

Assigning Volumes to Scratch Pools

RMS provides sixteen predefined scratch categories, or pools, which can help you manage 3494 or 3495 volumes. A scratch pool has the name SCRATCH*x* where *x* is 0–9 or A–F. Several RMS functions enable you to exploit the benefits of scratch pools with the 3494 or 3495; for example, you can:

- Assign volumes to, or release volumes from, scratch pools with either the SET VOLCAT function or the bulk library processing capability.
- Request that a volume from a scratch pool be mounted with the CATegory SCRATCH*x* option of a mount request.
- Request that volumes from a specified scratch pool be assigned to a device with the SET DEVCAT function, or unassigned from a device with the RESET DEVCAT function.
- Assign a specific volume to a scratch pool, or release a volume from a scratch pool, in conjunction with a mount or demount operation by specifying the TARGETcat option.

By means of a control file parameter, RM_DEFAULT_SCRATCH_POOL, you may designate a specific scratch category as the default scratch pool for use when SCRATCH is designated (as opposed to SCRATCH*x*).

RMS does not track characteristics of the volumes assigned to categories; for example, the media type of volumes in a scratch pool. It is a TMS function to track volume attributes and to ensure that scratch pool assignment and usage differentiates among media types.

Requesting Information from the 3494 or 3495 and RMS Master

RMS provides functions that enable you to inquire about DFSMS/VM requests and 3494 or 3495 information.

To inquire about the status of DFSMSRM requests that have been accepted for processing by RMS, issue the Query REQuests command.

To inquire about 3494 or 3495 information, issue a Query LIBrary request. A Query LIBrary request will give you 3495 information which includes:

- 3494 or 3495 operational status
- Inventory or volume count for a specific category or the entire library
- Status of volumes and devices

For information on the Query LIBrary command, see "DFSMSRM Query REQuests" on page 49. For information on the Query LIBrary CSL routine, see "FSMRMQLB – Query Library Information" on page 70.

Logging DFSMS/VM and 3494 or 3495 Messages

When you use RMS to support the 3494 or 3495, a variety of messages can be logged to help you understand DFSMS/VM processing events and hardware events.

Logging DFSMS/VM Messages

RMS logs DFSMS/VM messages for events that occur during normal processing, such as errors that occur in handling requests from users or fielding interrupts from the hardware.

You can specify the destination for these messages (console or file) and the level of messages (if any) to be logged in each location with two control file keywords, RM_LOG_TO_CONSOLE and RM_LOG_TO_FILE. You can also use these keywords to apply different logging procedures for 3494 or 3495 use than for space management and minidisk functions. For example, security and audit requirements for monitoring volume-mount activity can be satisfied by logging all messages, including informational messages, to a file or the console.

For more information on logging DFSMS/VM messages and on using RM_LOG keywords, see *z/VM: DFSMS/VM Customization*.

Recording Hardware Events

RMS also logs standard VM Environmental Record Editing and Printing Program (EREP) records. EREPs include temporary and permanent outboard recorder (OBR) records as well as miscellaneous data records.

Logging Hourly Library Statistics

The 3494 and 3495 provide hourly statistics that can be logged by attached hosts and used to monitor performance characteristics.

In order to collect this record on a regular basis, the entire range of 3494 or 3495 tape units connected to z/VM should be enabled for event monitoring in the I/O domain. CP (not RMS) handles the collection of hourly statistics. Typically a performance monitor application product is used to analyse the statistical data.

Generating Accounting Records

You can generate account records by using DFSMS/VM accounting services. To specify whether or not you want to use the DFSMS/VM accounting services, use the control file option RM_ACCOUNTING.

For more information about using the RM_ACCOUNTING option see *z/VM: DFSMS/VM Customization*. For more information about accounting records, see *z/VM: DFSMS/VM Storage Administration*.

Chapter 4. Creating DFSMS/VM RMS Bulk Processing Files

Using DFSMS/VM RMS bulk processing files, you can efficiently designate target categories for large or small groups of volumes. Instead of repetitively initiating DFSMSRM commands or CSL routines, you can create a bulk processing file that enables you to complete multiple volume category designations in one request. Bulk processing files enable you to perform either *automatic-insert* bulk processing or *on-request* bulk processing.

This chapter:

- Describes automatic-insert and on-request bulk processing
- Details the attribute and syntax conventions of bulk processing files
- Explains how to create a bulk processing file
- Gives examples of bulk processing files for both automatic-insert and on-request bulk processing.

Automatic-Insert Bulk Processing

With automatic-insert bulk processing you can specify which volumes are to be automatically moved from the INSERT category to another category, once they have been newly added to the 3494 or 3495.

Automatic insert processing does not start immediately after a volume is put into the input station. Instead, the hardware assigns the volume to the insert category.

Then, automatic insert processing starts when the insert category is not empty and the following events occur:

- The RMS master machine is initially started
- The RMS machine is restarted
- A valid MOUNT command is received by the RMS master
- A valid SET DEVCAT command is received by the RMS master

Either a MOUNT or SET DEVCAT command can be used but the specific command must be valid.

The following example describes the steps in this process in detail:

1. Insert two cartridges (CART01 and CART02) into the input station
2. The hardware microcode assigns each cartridge to the INSERT category
3. Issue a MOUNT command for CART09
4. MOUNT processing checks to see if the INSERT category is empty
5. Since the INSERT category is not empty, the MOUNT processing queues up automatic insert processing and continues the MOUNT processing
6. Automatic insert processing runs and assigns CART01 and CART02 to the category specified for each in the automatic insert bulk processing file

Automatic insert processing uses a device that is selected by the RMS master. When enhanced library control is not available and there is no device available when insert processing begins, insert processing stops. Since the insert processing starts before the MOUNT or SET DEVCAT command is completed, another device must be available. If another device is not available, move volumes from the insert category to another category by using the SET VOLCAT BULK command. However, when enhanced library control is available, insert processing can proceed regardless of the availability of a free drive.

On-Request Bulk Processing

On-request bulk processing enables you to handle multiple volume category changes with the SET VOLCAT BULK function. Using the SET VOLCAT BULK function is an alternative to repetitively performing the SET VOLCAT VOL function. Once you create a bulk processing file for on-request bulk processing, you can call the SET VOLCAT BULK function to perform tasks such as:

- Moving a group of volumes to a scratch pool
- Ejecting a group of volumes from the library
- Adding a group of volumes to the library (if they are not designated in an automatic-insert bulk processing file before automatic-insert processing occurs)

When you call the SET VOLCAT BULK function, RMS reads the bulk processing file, changes the volume category as specified in each entry of the file, and sends a report to the user ID that issued the request. The bulk processing report for on-request bulk processing includes:

- Volumes and target categories set successfully
- Volumes that do not exist in the 3495

For information on the SET VOLCAT command, see "DFSMSRM SET VOLCAT" on page 55. For information on the SET VOLCAT CSL routine, see "FSMRMSVC – Set Volume Category" on page 80.

Attributes of Bulk Processing Files

To create a bulk processing file for either automatic-insert or on-request bulk processing, you must follow the attribute and syntax conventions described in this part of the chapter.

File Name, File Type, and File Placement: The name, type, and placement of a bulk processing file depend on whether the file is used for automatic-insert bulk processing or on-request bulk processing.

When you create a bulk processing file for automatic-insert bulk processing, the contents of the file must apply to the resources located in a single 3494 or 3495. For this reason, you must name such a file with the identity of the 3494 or 3495 associated with it. The naming convention for an automatic-insert bulk processing file is:

```
RMBxxxxx DATA VMSYS:DFSMS.CONTROL
```

Where xxxxx is the sequence number of the 3494 or 3495 associated with the file, DATA is the file type, and VMSYS:DFSMS.CONTROL is the SFS directory. The sequence number is an identifier unique to each library and is assigned by IBM when the 3494 or 3495 is built (it is also the same number that is used for the RM_AUTO_LIBRARY control file parameter). For more information about the library sequence number, refer to *IBM TotalStorage Enterprise Automated Tape Library (3494) Introduction and Planning Guide*, GA32-0448 or *IBM TotalStorage Enterprise Automated Tape Library (3494) Operator's Guide*, GA32-0449.

For read access by the RMS master, you must place each automatic-insert bulk processing file (one for each 3494 or 3495) in the SFS directory, VMSYS:DFSMS.CONTROL.

When you create a bulk processing file for on-request bulk processing, you can place the file in any directory to which you and the RMS Master have access. For this reason, on-request bulk processing files can have any user-selected file name and file type that abide by the naming rules of the respective directory.

General Format: To create a bulk processing file for either automatic-insert or on-request bulk processing, follow the general format:

```
vlabel  TARGET_CATEGORY  <SOURCE_CATEGORY>          * optional comment
```

or

```
fr_vlab-to_vlab  TARGET_CATEGORY  <SOURCE_CATEGORY>  * optional comment
```

Where:

`vlabel`	is the volume's label
`fr_vlab-to_vlab`	is the range of the volumes' labels
`TARGET_CATEGORY`	is the target category of the volumes to be processed
`<SOURCE_CATEGORY>`	is the source category of the volumes to be processed

File Record Requirements: The record format of the bulk processing files must be fixed, and the logical record length must be no greater than 240 characters.

All bulk processing file records are inspected and must be valid. Each record in a particular bulk library processing file must pertain to volumes and categories that are located in the same 3494 or 3495. If any of the records are invalid, then no records are processed. A message is logged for each invalid record.

Record Structure: Create a bulk processing file with a single data-record type; do not use keywords. You can start required information for data records in any column but you must delimit the positional parameters by one or more blanks.

File Updates: When you change the contents of a bulk processing file, you do not need to restart the RMS master.

Comments: You can start a comment anywhere on the line. However, once the comment indicator is encountered, the remainder of the line is interpreted as a comment. Comments cannot span lines. That is, a comment stops at the end of a line; to continue a comment on the next line, start the next line with an asterisk.

Blank Lines: If you include blank lines in a bulk processing file, they are ignored by RMS processing.

Lowercase/Mixed-Case/Uppercase: Specify volume labels with A–Z and 0–9. Mixed case is allowed for the creation of bulk processing files.

Making Entries in a Bulk Processing File

To create a bulk processing file, position the entries in the following order, and delimit each entry by one or more blanks.

1. First, specify either a volume label or a range of volume labels.

   A volume label can be for a single volume or a range of volumes. Specify each volume label with one to six characters.

   When you specify a range, the starting volume label must have a lesser numeric value than the ending volume label (the characters A–Z are considered to be "less-than" 0–9). The starting and ending volume labels must be delimited by a dash (-), either with or without intervening blanks.

   If you want to specify that a volume label is in a specified range, make the label exactly as long (excluding trailing blanks) as the other volume labels specified in that range.

   Note:

   a. When you specify a range of volume labels, the beginning volume label and the ending volume label must be equal in length. If the two volume labels do not have equal lengths, the range of volume labels is invalid.
   b. For automatic-insert bulk processing files, only the first entry matching a volume in the INSERT category is used.
   c. Be aware that when specifying a range for a bulk insert, the range will include A-Z, then 0-9. For example, if you start at 10000, RMS will process volumes in the range 10000-10009, and then try processing 1001A-1001Z before processing 10010-10019.

You may wish to specify specific ranges in the bulk processing file to exclude attempts to process unwanted tape ranges.

2. Next, specify the target category name.

   The target category name specifies to which category the volumes are assigned.

   Valid target category specifications are:

   - EJECT—the volume is ejected and placed in the convenience input/output station
   - EJECTB—the volume is ejected and placed in the high-capacity input/output station
   - SCRATCH*x*—volumes are moved to one of sixteen scratch pools
   - VOLspecific—volumes are moved to the volume-specific category
   - *hexvalue*—hexadecimal category in the range of 0000 to FFFF

3. Finally, you may specify the source category name. (This entry is optional.)

   The source category name indicates to which category the volumes are currently assigned. If this parameter is included, the specified volumes are moved to the target category only if the volumes are currently assigned to the specific source category. If you designate the source category, you can prevent unintended category changes.

   Valid source category specifications are:

   - INSERT—the volume's category is only changed if it is new to the library
   - SCRATCH*x*—volumes are moved from one of sixteen scratch pools
   - VOLspecific—volumes are moved from the volume-specific category
   - *hexvalue*—hexadecimal category in the range of 0000 to FFFF

   Note:

   a. If you do not specify the source category for on-request processing, the new target category is set, regardless of the current category.
   b. If you do not specify a source category for automatic-insert processing, the INSERT source category is assumed.
   c. If you include information after the optional third positional parameter, it is ignored, even if it is not preceded by an asterisk.

Example: Automatic-Insert Bulk Processing File

Figure 6 on page 26 shows sample entries for a bulk processing file that is used for automatic-insert bulk processing.

```
* Vol label       Target    Source

 VOL3             SCRATCH0  INSERT         * optional comment
 VOL7             SCRATCH1  INSERT
 VOL07            SCRATCH2  INSERT         * optional comment
 VOL1-VOL9        SCRATCH3  INSERT
 VOL01-VOL09      SCRATCH4  INSERT         * optional comment
```

Figure 6. Sample Automatic Library Add list

RMS searches automatic-insert bulk processing files from the top of the file to the bottom. When you create a bulk processing file, remember that sequencing volume labels in the file can be critical if a volume meets the criteria of more than one entry. If a volume's category is defined more than once in a file, its category is set by the very first entry and all other entries for that volume or range of volumes are ignored.

In this example, if the volumes VOL03, VOL07, VOL10, VOL2, VOL3, and VOL7 are in the INSERT category, they are processed as follows:

VOL3
would be set to category SCRATCH0

VOL7
would be set to category SCRATCH1

VOL07
would be set to category SCRATCH2

VOL2
would be set to category SCRATCH3

VOL03
would be set to category SCRATCH4

VOL10
would NOT be set to a new category

Volumes VOL3, VOL7, and VOL07 are only processed for their first occurrence in the list because once they are processed here, they are no longer in the INSERT category. The volume VOL10 is not processed because it is not entered in this bulk processing file.

Example: On-Request Bulk Processing File

Figure 7 on page 27 shows sample entries for a bulk processing file to be used by a SET VOLCAT BULK function for on-request bulk processing.

```
* Vol label       Target    Source

  876511          SCRATCH2            *set volumes meeting criteria to the SCRATCH2 category
  876441          SCRATCH2            *set volumes meeting criteria to the SCRATCH2 category

  323000-323010   EJECT     SCRATCH1  *eject volumes in this range from the SCRATCH1 category
```

Figure 7. Sample Bulk Processing File for On-Request Bulk Processing

When you create a bulk processing file, remember that the sequence of volume labels is critical if a volume meets the criteria of more than one entry. For example, if both a single volume entry and a range of volumes are included for a particular volume label, the SET VOLCAT function is be performed more than once for this volume. If you specify a source category for each volume label or range of volume labels, the danger of repetitive processing is minimized.

Chapter 5. Customizing Removable Media Services

PI

RMS processing includes several types of installation-wide exit that you may modify if you wish to customize support of 3494 or 3495 functions. Use of the RMS types of installation-wide exit is optional.

This chapter describes each of the installation-wide exits and the input parameters that allow you to add additional processing for your installation or override default processing provided with the RMS master.

User exits are located in FSMPSI CSLLIB on the product disk and must be replaced there if you modify them.

Summary of the RMS Installation-Wide Exits

The following types of installation-wide exit are used in RMS processing:

FSMACCNT is called to perform accounting services available to DFSMS/VM. Installation modification of the accounting record can be accomplished by calling this installation-wide exit prior to record generation. For more information about this exit, refer to *z/VM: DFSMS/VM Customization*.

FSMRMATE is called to issue commands for device attachment.

FSMRMDEV is called to select a real device if one is not specified.

FSMRMDTE is called to issue commands for device detachment.

FSMRMINC is called after RMS initialization has completed.

FSMRMINI is called before RMS initialization has started.

FSMRMPRE is called before the processing of an authorized request.

FSMRMPRO is called after the processing of an authorized request.

FSMRMSHR is called to verify that volumes and categories specified in requested commands do not violate partitioning criteria for any 3494 or 3495 accessed concurrently by other operating systems.

FSMVAUTH is called to perform authorization checking. This is a required CSL routine and a common installation-wide exit used for all DFSMS/VM functions. For more information about this installation-wide exit, refer to *z/VM: DFSMS/VM Customization*.

A more detailed description, including input and parameters, for each installation-wide exit follows. Among the input parameters for most types of installation-wide exit is a code for the request-type being processed. Table 6 on page 29 lists the subset of DFSMS/VM command codes that applies to the RMS master. See Chapter 6 for detailed information on each of the commands.

Table 6. Command Codes Passed to Installation-Wide Exit types

Command	CSL Routine	Command Code
DFSMSRM DEMOUNT	FSMRMDMT	RMDMOUNT
DFSMSRM DISCard (user’s)	n/a	RMDISCDO
DFSMSRM DISCard (any)	n/a	RMDISCDA
DFSMSRM MOUNT	FSMRMMNT	RMMOUNT
DFSMSRM Query LIBrary	FSMRMQLB	RMQUERYL
DFSMSRM Query REQuests (* or own)	n/a	RMQUERYO
DFSMSRM Query REQuests (ALL, FOR, or other user’s)	n/a	RMQUERYA

Table 6. Command Codes Passed to Installation-Wide Exit types (continued)

Command	CSL Routine	Command Code
DFSMSRM RESET DEVCAT	FSMRMRDC	RMRSETDC
DFSMSRM SET DEVCAT	FSMRMSDC	RMSETDC
DFSMSRM SET VOLCAT	FSMRMSVC	RMSETVC
DFSMSRM SET VOLCAT BULK	FSMRMSVB	RMSETVCB
DFSMSRM STOP	n/a	RMSTOP
Note: Commands with an *n/a* designation in the "CSL Routine" column do not have CSL routine equivalents.		

FSMACCNT – Accounting

The accounting installation-wide exit, FSMACCNT, is a CSL routine that provides the means for an installation to modify an accounting record just prior to generating the record or to stop a record from being generated. You may customize the output of an accounting record or suppress the generation of accounting records by replacing the CSL routine, FSMACCNT, in FSMPSI CSLLIB on the product disk. As shipped, the CSL routine returns a 0 return code (write the record). FSMACCNT is invoked at the end of the specified DFSMS/VM processing. Refer to *z/VM: DFSMS/VM Customization* for detailed information about the FSMACCNT installation-wide exit, including the exit's return codes, parameters, and usage notes.

FSMRMATE – Device Attachment

The device-attachment installation-wide exit, FSMRMATE, allows your installation to substitute its own command for attaching the requested (or selected) device:

- To the RMS master in preparation for request processing
- To a user, after processing is completed, for a MOUNT or SET DEVCAT request that specifies device attachment upon completion

This installation-wide exit allows the following options:

1. To use the product default processing that invokes the CP ATTACH command.

   FSMRMATE should be retained in the CSL library, as shipped with DFSMS/VM to maintain default processing.
2. To replace FSMRMATE with a routine that accomplishes device attachment by an alternate technique.

The FSMRMATE CSL routine is called by DFSMS/VM with the following parameter list.

FSMRMATE Parameter List:

Parameter Name and Description	***Attributes***	
Return Code is the outcome of FSMRMATE processing.	FIXED(31)	Output
Userid is the ID of the user to whom the device is attached.	CHAR(8)	Input
Real Device Address is the real device to be attached.	BIT(16)	Input
Virtual Device Address is the address at which the device is attached.	BIT(16)	Input
Assign Flag indicates if the device should be attached with the ASSIGN option "Y" or the NOASSIGN option "N".	CHAR(1)	Input

FSMRMATE Return Codes:

RC	*Meaning*
0	Device attachment has performed successfully.
4	Exit processing cannot be used for device attachment.
8	Problem has occurred during processing.

Note: When RM_USE_GIVE is defined as Y in the DGTVCNTL DATA file, this exit is not called upon completion of any request that specifies device attachment.

FSMRMDEV – Real Device Selection

The real-device-selection installation-wide exit, FSMRMDEV, can select a device when a real device address is not specified in the incoming request. As shipped with DFSMS/VM, this installation-wide exit takes no action and can be replaced with a customized device-selection routine.

When a real device address is not specified in the incoming request, this installation-wide exit performs one of three options:

1. Uses the product default processing to select the first available (unattached) library device that can be located, but fails the request if no devices are available.

   This is the option that is shipped with the product.
2. Replaces FSMRMDEV with a device-selection program.

   This alternative provides the potential for selecting an appropriate device type for the volume at hand, whereas default processing is indiscriminate about device types and may result in device-type/media-type mismatch. Note that in a typical environment, the tape management system selects a device, and processing for this installation-wide exit is not required.
3. Fails all requests in which no real device is designated by the requester. With this alternative, it is essential that the real device (RDEV) keyword is an element of the request syntax.

   You can choose this option by replacing FSMRMDEV with a routine that returns a return code of 8.

Note: This installation-wide exit is invoked when a real device address has not been specified in the request syntax for all request types. This includes Query LIBrary or SET VOLCAT, which are not sensitive to device-type considerations. Restricting the set of devices that can be chosen increases the chance of request failure due to lack of device availability. Therefore, if FSMRMDEV is replaced with a more sophisticated device-selection algorithm, use the command code to ensure that selection restrictions are applied only to the MOUNT and SET DEVCAT requests. For other request types, device selection can be deferred to the RMS master's default technique of issuing return code 4.

If you are replacing the CSL routine shipped as part of DFSMS/VM, refer to *z/VM: CMS Application Development Guide for Assembler* for information on coding CSL routines.

The FSMRMDEV CSL routine is called by DFSMS/VM with the following parameter list. Not all parameters are relevant to every request type, but the structure and sequence of the parameter list must be maintained according to standard CSL calling conventions.

FSMRMDEV Parameter List:

Parameter Name and Description	*Attributes*	
Return Code is outcome of FSMRMDEV processing.	FIXED(31)	Output
Userid is the ID of the user who issued the command.	CHAR(8)	Input
Command Code designates the command being processed (from Table 6 on page 29.)	CHAR(8)	Input
Attach-to Userid is the ID of a user (optionally specified in the MOUNT or SET DEVCAT function) to which the device may be attached instead of to the requester upon command completion.	CHAR(8)	Input

Parameter Name and Description	*Attributes*	
Volume Label is the external volume label specified for a request.	CHAR(6)	Input
Source Category Name is the name of the category that the volume is currently assigned to.	CHAR(*)	Input
Source Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Target Category Name is the name of the category to which the volume is assigned.	CHAR(*)	Input
Target Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Library Name is the up-to-32-character user-specified name of the library.	CHAR(*)	Input
Library Name Length is the length of the previous parameter.	FIXED(31)	Input
Real Device Address is the device selected for use in the request.	BIT(16)	Output

FSMRMDEV Return Codes:

RC	*Meaning*
0	Exit processing completes successfully.
4	Exit processing cannot be used for device selection.
8	Problem has occurred during processing.

FSMRMDTE – Device Detachment

The device-detachment installation-wide exit, FSMRMDTE, allows your installation to substitute its own command for detaching the requested (or selected) device:

- From the requester, prior to attachment to the RMS master for request processing, when the device is not unattached
- From the RMS master, at the conclusion of request processing.

This installation-wide exit allows the following options:

1. To use the product default processing that invokes the CP DETACH command with the LEAVE option.

   Retain FSMRMDTE in the CSL library, as shipped with DFSMS/VM, to maintain default processing.
2. To replace FSMRMDTE with a routine that accomplishes device detachment by an alternate technique.

The FSMRMDTE CSL routine is called by DFSMS/VM with the following parameter list.

FSMRMDTE Parameter List:

Parameter Name and Description	*Attributes*	
Return Code is the outcome of FSMRMDTE processing.	FIXED(31)	Output
Userid is the ID of the user from whom the device is detached.	CHAR(8)	Input
Real Device Address is the real device to be detached.	BIT(16)	Input

FSMRMDTE Return Codes:

RC	*Meaning*
0	Device detachment has performed successfully.
4	Exit processing cannot be used for device detachment.
8	Problem has occurred during processing.

Note: When RM_USE_GIVE is defined as Y in the DGTVCNTL DATA file, this exit is not called upon completion of any request that specifies device attachment.

FSMRMINC – RMS Post-Initialization

The post-initialization installation wide-exit, FSMRMINC, can be used for any installation processing that should occur after RMS initialization has completed. As shipped with DFSMS/VM, this installation-wide exit takes no action and can be replaced with a customized post-initialization routine.

The FSMRMINC CSL routine is called by DFSMS/VM (RMS) with the return code from initialization, so the exit can take appropriate action based on whether RMS initialization was successful.

The exit routine should return unaltered the return code passed from RMS, unless the exit needs to change whether RMS will continue processing.

FSMRMINC Return Codes:

RC	*Meaning*
0	Initialization completed successfully.
4	Initialization completed with warnings, but RMS will continue processing.
8	Initialization completed with errors — RMS will terminate processing.

FSMRMINI – RMS Pre-Initialization

The pre-initialization installation wide-exit, FSMRMINI, can be used for any installation processing that should occur before RMS initialization starts. As shipped with DFSMS/VM, this installation-wide exit takes no action and can be replaced with a customized pre-initialization routine.

The FSMRMINI CSL routine is called by DFSMS/VM (RMS) with no input parameter list. On return, DFSMS/VM expects a return code value of 0.

FSMRMPRE – RMS Preprocessing

The RMS preprocessing installation-wide exit, FSMRMPRE, provides flexibility for using the RMS master with tape management systems that vary in terms of functions they support. This installation-wide exit is called before an actual command request to the 3494 or 3495 hardware takes place but after the following processing steps have completed successfully:

- Standard DFSMS/VM authorization processing
- Further request-specific options are resolved, including device selection, if appropriate
- Library partition-criteria checking (FSMRMSHR)

In essence, this installation-wide exit is called only when the request meets all validity criteria.

This installation-wide exit allows the following options:

1. Do no additional processing, in which case the CSL routine, FSMRMPRE, as shipped with DFSMS/VM, remains in place.
2. Replace FSMRMPRE with a routine that performs desired functions.

One potential use for this installation-wide exit is object-level security checking, as described in “Authorizing Users” on page 14. This use is appropriate when this function is not supplied by a tape management system.

If you are replacing the CSL routine shipped as part of DFSMS/VM, refer to *z/VM: CMS Application Development Guide for Assembler* for information on coding CSL routines.

The FSMRMPRE CSL routine is called by DFSMS/VM with the following parameter list. Not all parameters are relevant to every request type, but the structure and sequence of the parameter list must be maintained according to standard CSL calling conventions.

FSMRMPRE Parameter List:

Parameter Name and Description	*Attributes*	
Return Code is outcome of FSMRMPRE processing.	FIXED(31)	Output
Userid is the ID of the user who issues the command.	CHAR(8)	Input
Command Code designates the command being processed (from Table 6 on page 29.)	CHAR(8)	Input
Attach-to Userid is the ID of a user (optionally specified in a MOUNT or SET DEVCAT command) to which the device may be attached instead of to the requester upon command completion.	CHAR(8)	Input
Volume Label is the external volume label specified for a request.	CHAR(6)	Input
Source Category Name is the name of the category that the volume is currently assigned to.	CHAR(*)	Input
Source Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Target Category Name is the name of the category to which the volume is assigned.	CHAR(*)	Input
Target Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Library Name is the up-to-32-character user-specified name of the library.	CHAR(*)	Input
Library Name Length is the length of the previous parameter.	FIXED(31)	Input
Real Device Address is the device selected for use in the request.	BIT(16)	Input

FSMRMPRE Return Codes:

RC	*Meaning*
0	Exit processing has completed successfully.
8	Problem has occurred during processing.

FSMRMPRO – RMS Postprocessing

The RMS postprocessing installation-wide exit, FSMRMPRO, provides further flexibility for using the RMS master with tape management systems. This installation-wide exit is called after actual command interaction with 3494 or 3495 hardware has completed successfully and while the device is still attached to the RMS master.

This installation-wide exit allows the following options:

1. Do no additional processing, in which case the CSL routine, FSMRMPRO, as shipped with DFSMS/VM, remains in place.
2. Replace FSMRMPRO with a routine that performs desired functions.

Volume label verification is an example of the type of optional processing that may be invoked from the postprocessing installation-wide exit.

If you are replacing the CSL routine shipped as part of DFSMS/VM, refer to *z/VM: CMS Application Development Guide for Assembler* for information on coding CSL routines.

The FSMRMPRO CSL routine is called by DFSMS/VM with the following parameter list. Not all parameters are relevant to every request type, but the structure and sequence of the parameter list must be maintained according to standard CSL calling conventions.

FSMRMPRO Parameter List:

Parameter Name and Description	*Attributes*	
Return Code is outcome of FSMRMPRO processing.	FIXED(31)	Output
Userid is the ID of the user who issued the command.	CHAR(8)	Input
Command Code designates the command being processed (from Table 6 on page 29.)	CHAR(8)	Input
Attach-to Userid is the ID of a user (optionally specified in a MOUNT or SET DEVCAT command) to which the device may be attached instead of to the requester upon command completion.	CHAR(8)	Input
Volume Label is the external volume label specified for a request.	CHAR(6)	Input
Source Category Name is the name of the category that the volume is currently assigned.	CHAR(*)	Input
Source Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Target Category Name is the name of the category to which the volume is assigned.	CHAR(*)	Input
Target Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Library Name is the up-to-32-character user-specified name of the library.	CHAR(*)	Input
Library Name Length is the length of the previous parameter.	FIXED(31)	Input
Real Device Address is the device selected for use in the request.	BIT(16)	Input

FSMRMPRO Return Codes:

RC	*Meaning*
0	Exit processing has completed successfully.
8	Problem has occurred during processing.

FSMRMSHR – RMS Library-Partitioning

You can partition 3494 or 3495 resources so that a VM system can access the 3494 or 3495 hardware concurrently with other operating environments, for example, MVS or another VM system. FSMRMSHR screens category designations and volume labels that are requested for use by the VM system. This RMS library-partitioning installation-wide exit also ensures that a volume, or category, dedicated to another operating environment is not used. However, it does not provide the ability for VM and the other operating environments to share specific volumes and categories through a central “catalog” facility. This installation-wide exit is called during the processing of MOUNT, SET DEVCAT, and SET VOLCAT requests and as part of bulk library processing.

This installation-wide exit allows the following options:

1. To use the product default processing which returns a successful return code, indicating that partitioning or sharing criteria have been met.

   Retain FSMRMSHR in the CSL library, as shipped with DFSMS/VM, to maintain default processing.
2. To replace FSMRMSHR with a routine that examines the volume and category parameters supplied as input and screens according to installation guidelines for partitioning or sharing resources.

If you are replacing the CSL routine shipped as part of DFSMS/VM, refer to *z/VM: CMS Application Development Guide for Assembler* for information on coding CSL routines.

The FSMRMSHR CSL routine is called by DFSMS/VM with the following parameter list. Not all parameters may be relevant to every request type, but the structure and sequence of the parameter list must be maintained according to standard CSL calling conventions.

FSMRMSHR Parameter List:

Parameter Name and Description	*Attributes*	
Return Code is outcome of FSMRMSHR processing.	FIXED(31)	Output
Volume Label is the external volume label specified for a request.	CHAR(6)	Input
Source Category Name is the name of the category that the volume is currently assigned.	CHAR(*)	Input
Source Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Target Category Name is the name of the category to which the volume is assigned.	CHAR(*)	Input
Target Category Name Length is the length of the previous parameter.	FIXED(31)	Input
Library Name is the up-to-32-character user-specified name of the library.	CHAR(*)	Input
Library Name Length is the length of the previous parameter.	FIXED(31)	Input

FSMRMSHR Return Codes:

RC	*Meaning*
0	Volume and categories meet criteria.
8	Volume or category does not meet criteria.

FSMVAUTH – Authorization

This installation-wide exit with which the user provides a local alternative to standard authorization checking, functions in the same way in RMS processing as in other DFSMS/VM services. Refer to "Authorizing Users" on page 14 for additional information on authorizing users. Refer to *z/VM: DFSMS/VM Customization* for detailed information about the FSMVAUTH installation-wide exit, including the exit's return codes, parameters, and usage notes.

PI end

Part 2. Reference

Chapter 6. DFSMSRM Commands

DFSMSRM commands allow you to request the automated tape library services provided by the 3494 or 3495, interactively. For most command functions there is a comparable CSL interface that allows the 3494 or 3495 function to be requested under program control.

This chapter contains reference information about DFSMSRM commands. Each part of this chapter includes a command's name, functional description, syntax diagram, operands and options, and usage notes. Below are brief descriptions of commands detailed in this chapter.

Important

Any userid issuing a DFSMSRM command (as well as issuing a DFSMS command, using the CSL interface, or using the ISMF interface), must have an `IUCV ANY` statement in its directory. See *z/VM: CP Planning and Administration* for more information on including this statement in a user directory.

DFSMSRM DEMOUNT allows you to request that a volume currently mounted on a specific device be demounted from that device. Demounting a volume makes it eligible for remount to another device.

DFSMSRM DISCard allows you to cancel a specific command that has been accepted by the RMS master.

DFSMSRM MOUNT allows you to request that a specific volume or a volume from a specific category be mounted.

DFSMSRM Query LIBrary allows you to obtain information about the state of the 3494 or 3495, the status of a volume or device, and the inventory of the library itself.

DFSMSRM Query REQuests allows you to obtain information about the status of DFSMSRM requests.

DFSMSRM RESET DEVCAT allows you to disassociate a category from the device to which it was assigned.

DFSMSRM SET DEVCAT allows you to assign a specific category to a library device.

DFSMSRM SET VOLCAT allows you to assign a specific volume to a category.

DFSMSRM STOP allows you to quiesce or halt the RMS master.

Warning

During RMS initialization, command request behavior may be unpredictable based on whether a device (if required) is available for processing the request.

DFSMSRM DEMOUNT

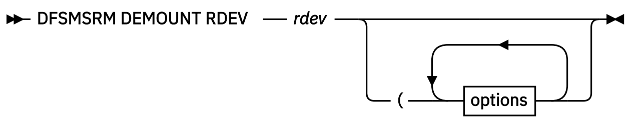
The DFSMSRM DEMOUNT command allows you to request that a volume currently mounted on a specific device be demounted from that device.

Demounting a volume makes it eligible for remount. However, DEMOUNT requests are optional since volume demounts are handled routinely during RMS master processing.

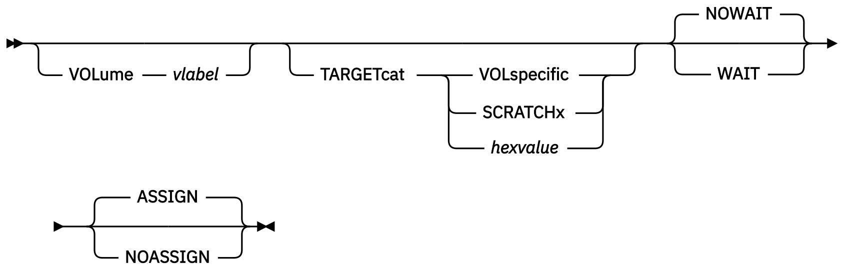
If the real device was attached to the user prior to the command being issued, it will be reattached upon completion of command processing.

The command syntax of the DFSMSRM DEMOUNT command is:

The DFSMSRM DEMOUNT Command



options



Operand

RDEV *rdev*
is the real address of the device on which the volume is mounted. This is a required parameter.

Options

VOLume *vlabel*
is the up-to-six-character, exterior label of the specific volume to be demounted from the device.

TARGETcat
allows you to determine how the selected volume is recategorized. You have three choices:

VOLspecific
classifies the selected volume as volume-specific; that is, the volume is not to be a member of a category.

SCRATCH*x*
classifies the selected volume as a member of a predefined scratch pool where *x* is 0–9 or A–F.

hexvalue
is a hexadecimal category in the range of 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM redbooks, should be performed prior to using as a category.

NOWAIT
gives control to the command issuer as soon as the DEMOUNT request is received by DFSMSRM, thereby creating asynchronous handling of the functions. Use this option when you do not want to wait for request processing to complete. This keyword is the default and cannot be used with the WAIT keyword.

WAIT
: allows the command issuer to wait until request processing has completed, thereby creating synchronous handling of the functions. Control is not returned to the command issuer until processing has been completed by DFSMSRM. This keyword cannot be used with the NOWAIT keyword.

ASSIGN
: indicates that the device will be assigned to the system as part of device attach performed by the RMS master (to itself or to the requested user ID upon request completion). This keyword cannot be used with the NOASSIGN keyword; it is the default if neither is specified and is the normal operational approach.

NOASSIGN
: indicates that attach operations performed by the RMS master must not affect device assignment. When this option is used, the requester is responsible for issuing the appropriate ASSIGN and UNASSIGN channel command words. While this capability is not part of typical library-usage practices, it provides flexibility when connecting library devices to multiple systems and sharing library resources. This keyword cannot be used with the ASSIGN keyword.

Usage Notes

1. The CSL routine FSMRMDMT provides a programming interface equivalent to the DFSMSRM DEMOUNT command.
2. If the specified volume is not currently mounted on the specified device, the command fails.
3. If, prior to issuing the mount command, the real device has been attached using the multiuser option to the user issuing the mount request, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.
4. Final response messages for successfully completed DEMOUNT requests include:
    - The volume label for the volume demounted
    - The real device address
    - The library name
5. TS7700 configurations do not support a target category change while the volume is mounted. Peer-to-Peer and TS32700 configurations do not support a target category change as part of the DFSMSRM DEMOUNT command.

DFSMSRM DISCard

The DFSMSRM DISCard command allows you to cancel certain DFSMSRM command requests that have been accepted by the RMS master for processing. If the RMS master has not yet begun to interact with the hardware, then processing for the identified request is terminated when the DISCard request is received.

DISCard can be used for these commands:

- MOUNT
- Query LIBrary (CATegory or INVentory)
- SET VOLCAT (BULK only)

DISCard cannot be used for the following commands:

- DEMOUNT
- DISCard
- Query (REQuests and LIB requests that are synchronous)
- RESET DEVCAT
- SET DEVCAT
- SET VOLCAT (VOL only)
- STOP

The command syntax of the DFSMSRM DISCard command is:

The DFSMSRM DISCard Command

```
►►─ DFSMSRM DISCard ── requestid ─►◄
```

Operand

requestid
identifies which request is discarded. This ID is returned to the requester when an asynchronous command is accepted by the RMS master. However, the requester can also obtain this ID by issuing the DFSMSRM Query REQuests command; issuing Query REQuests is the only way to get the IDs of synchronous requests.

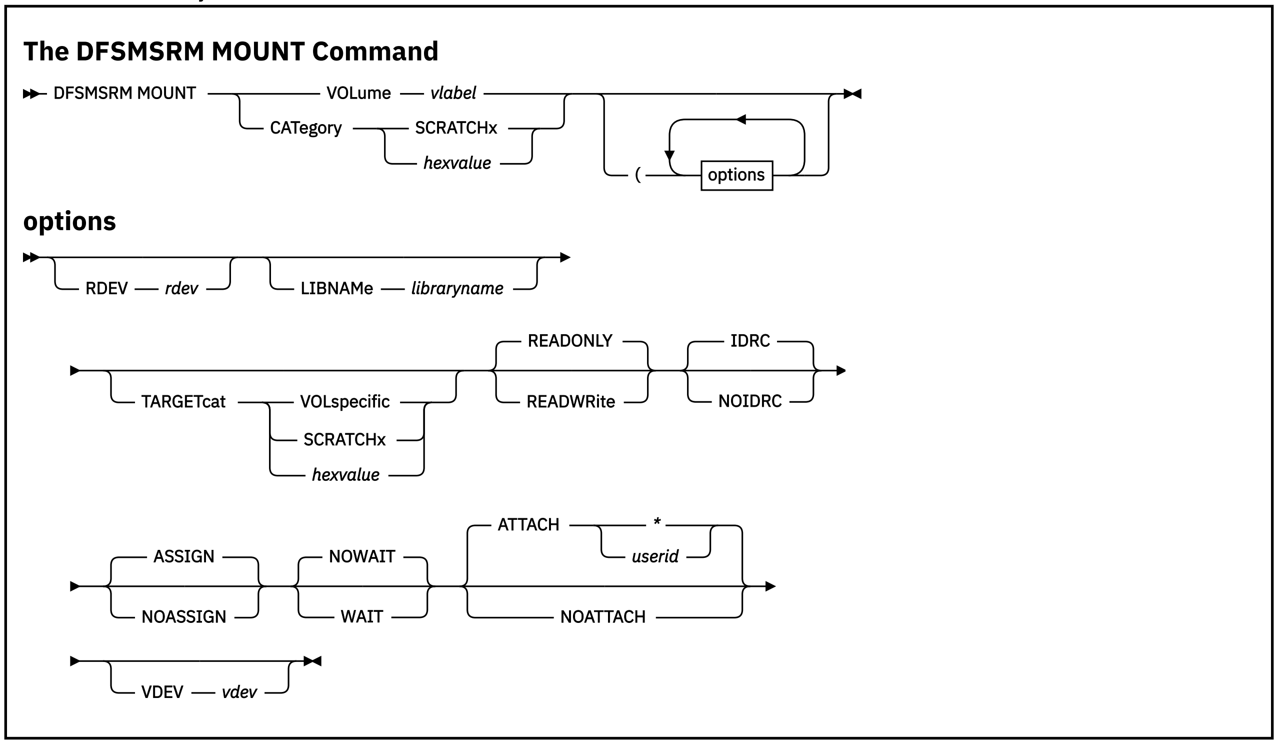
Usage Notes

1. There is no programming interface equivalent to the DFSMSRM DISCard command.
2. The DFSMSRM DISCard command is handled synchronously; that is, the command does an implicit wait until DISCard processing is complete. At that time, a message describing the success of the operation is returned to the requester's console.
3. In the case of a SET VOLCAT BULK request, processing is discontinued even if the request has only partially completed. Once the processing is discontinued, the Bulk Library Processing Report is returned to the requester; actions completed prior to the DISCard request are noted in the report.

DFSMSRM MOUNT

The DFSMSRM MOUNT command allows you to request that a specific volume or a volume from a specific category be mounted to a device.

The command syntax of the DFSMSRM MOUNT command is:



Operands

VOLume *vlabel*
: is the up-to-six-character, exterior label of the specific volume mounted to the device.

CATegory
: allows you to determine how the selected volume is recategorized. You have two choices:

SCRATCH*x*
: is the name of the scratch pool from which the next volume is mounted. Typically, this scratch pool has been previously assigned to the device, and the stacker has been indexed at mount time. The scratch pool does not have to be assigned to a device when you request that the next sequential volume from the scratch pool be mounted on it.

hexvalue
: is a hexadecimal category in the range 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM redbooks, should be performed prior to using as a category.

Options

RDEV *rdev*
: is the real address of the device on which the volume is mounted. If this keyword is omitted, any available device is used.

LIBNAME *libraryname*
: specifies the library name. This option is required if a mount request is associated with a manual device. This option is not required if there is an automated library, or if the RDEV keyword is specified.

TARGETcat
allows you to determine how the selected volume is recategorized once it is mounted. When this option is not specified, the volume's category remains unchanged. When you use this option, you have three choices:

VOLspecific
classifies the selected volume as volume-specific; that is, the volume is not a member of a category.

SCRATCH*x*
classifies the selected volume as a member of a scratch pool, where *x* is a value from 0–9 or A–F.

hexvalue
is a hexadecimal category in the range 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM redbooks, should be performed prior to using as a category.

READONLY
sets the logical write protect feature on the device, prior to attaching it to the requester. This keyword is ignored if the NOATTACH option is specified. This keyword is the default and cannot be used with the READWRite keyword. This default may be overridden by means of a control-file parameter.

READWRite
deactivates the logical write protection on the device, prior to attaching it to the requester. This keyword cannot be used with the READONLY keyword.

IDRC
indicates that an IDRC-capable device is required. This option is provided for mount requests targeted for manual tape libraries where the characteristics of a device must be communicated. (All library devices are IDRC-capable.) This keyword is the default and cannot be used with the NOIDRC keyword.

NOIDRC
indicates that an IDRC-capable device is not required. This option is provided for mount requests targeted for manual tape libraries where the characteristics of a device must be communicated. This keyword cannot be used with the IDRC keyword.

ASSIGN
indicates that the device will be assigned to the system as part of device attach performed by the RMS master (to itself or to the requested user ID upon request completion). This keyword cannot be used with the NOASSIGN keyword; it is the default if neither is specified and is the normal operational approach.

NOASSIGN
indicates that attach operations performed by the RMS master must not affect device assignment to the system. When this option is used, the requester is responsible for issuing the appropriate ASSIGN and UNASSIGN channel command words. While this capability is not part of typical library-usage practices, it provides flexibility when connecting library devices to multiple systems and sharing library resources. This keyword cannot be used with the ASSIGN keyword.

NOWAIT
gives control to the command issuer as soon as the request is received by DFSMSRM, thereby creating asynchronous handling of the functions. Use this option when you do not want to wait for request processing to complete. This keyword is the default and cannot be used with the WAIT keyword.

WAIT
allows the command issuer to wait until request processing has completed, thereby CREATING synchronous handling of the functions. Control is not returned to the command issuer until processing has been completed by DFSMSRM. This keyword cannot be used with the NOWAIT keyword.

ATTACH **** or userid***
indicates that the device is attached to the requester (*) or a specified user ID once the mount operation is complete. The requester (*) is the default if a user ID is not specified. This keyword is the default and cannot be used with the NOATTACH keyword.

NOATTACH
indicates that the device is not attached to a user, once the mount operation is complete. This option should be used judiciously if the device selection is performed by the RMS master, because a free device is eligible for the incoming mount requests. This keyword cannot be used with the ATTACH keyword.

VDEV *vdev*
attaches the device to the requester (or alternative user ID) at the specified virtual address. This keyword is valid only when the ATTACH option is explicitly or implicitly specified. For information on what happens if this option is not specified, refer to "Usage Notes" on page 45.

Usage Notes

1. The CSL routine FSMRMMNT provides a programming interface equivalent to the DFSMSRM MOUNT command.
2. You can send a MOUNT request to a designated user ID for manual mount if a manual library name is specified in the request.
3. If the VDEV option is not specified, the virtual address used when the device is attached to the requestor or alternative user ID is determined as follows.
   - If the device was originally attached to the requestor, the device's virtual address is preserved.
   - If the device was *free*, the device's real address is used as the virtual address.
4. If, prior to issuing the mount command, the real device has been attached using the multiuser option to either the user issuing the mount request or to the user identified with the attach option, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.
5. For MOUNT requests that are successfully completed, the following information is included in the final response:
   - The volume label for the volume mounted
   - The real device address
   - The library name
   - The name of the category to which the mounted volume belongs or is assigned
   - The virtual address attached to the requester or requested user ID (unless NOATTACH is specified)

   The target category is included in the response message only if you explicitly included the corresponding keywords in the command.
6. When the 3494 or the 3495 is shared by multiple operating environments, the installation-wide exit FSMRMSHR (RMS-library partitioning) provides an additional level of control to ensure that only volumes within the control of the VM system partition are mounted by VM requesters. Refer to "Considerations for a Multiple-Host Configuration" on page 13 for further details.

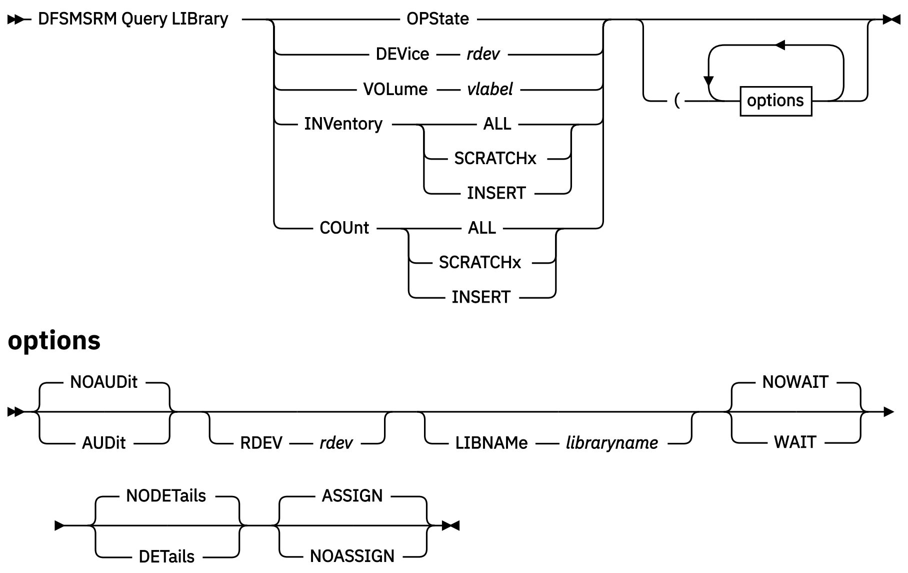
DFSMSRM Query LIBrary

The DFSMSRM Query LIBrary command allows you to:

- Query the operational state of the 3494 or 3495
- Request information on a specific real device in the library
- Request information on a specific volume
- Inventory one or more volumes in a specific scratch pool or library
- Count the number of volumes in a specific scratch pool or library

The command syntax of the DFSMSRM Query LIBrary command is:

The DFSMSRM Query LIBrary Command



Operands

OPState
: queries the operational state of the 3494 or 3495. If neither LIBNAME nor RDEV is specified when multiple libraries are connected, then the status of the first automated 3494 or 3495 listed in the control file is reported.

VOLume *vlabel*
: requests information on a specific volume, as specified by the six-character exterior volume label. If there are multiple libraries attached, then all 3494s or 3495s are checked for the volume until found. If a library name is supplied on a multiple library environment, this library will be checked first followed by all libraries.

DEVice *rdev*
: requests information on a specific real device in the library, as specified by the four-character real device address.

INVentory
: allows you to request an inventory list of volumes. You have three choices:

  ALL
  : requests an inventory list of the total volumes in the library.

  SCRATCH*x*
  : requests an inventory list of volumes in one of sixteen possible scratch pools where *x* is a value from 0–9 or A–F.

INSERT
: requests an inventory list of volumes that have entered the library, but are not yet assigned to a target category.

COUnt
: allows you to request the number of volumes.

ALL
: requests a count of the total volumes in the library.

SCRATCH*x*
: requests a count of the volumes in one of sixteen possible scratch pools where *x* is a value from 0–9 or A–F.

INSERT
: requests a count of volumes that have entered the library, but are not yet assigned to a target category.

Options

NOAUDit
: indicates that the specified volume should not be physically audited by the 3494 or 3495 robotics vision system. This option applies only to the Query LIBrary VOLume requests and is invalid if included for any other type of query. This option is the default for Query LIBrary VOLume requests and cannot be used with the AUDit option.

AUDit
: indicates that the specified volume is to be physically audited by the 3494 or 3495 robotics vision system. This option applies only to the Query LIBrary VOLume requests and is invalid if included for any other type of Query. This option cannot be used with the NOAUDit option.

RDEV *rdev*
: is the real address of a device in the library of interest. This designation provides a real device path for use by the RMS master in communicating with the library manager to obtain Query LIBrary data. This real device address must be available for attachment by the RMS master. If this parameter is not provided, the RMS master attempts to find an unattached device (in the library specified with the LIBNAME parameter or in the default library when there are multiple 3494s or 3495s) to use as a communication path for the query request. The request fails if such a device is not found.

If RDEV is specified when enhanced library control is in use, the drive specified must be a valid address, although this drive is not necessarily used to satisfy the request. A drive control path is required for the Query Volume with Audit option.

If the requisite software is not installed, a free tape drive is required for all communications.

Do not use this option when a DEVice query is requested, because the device that is the object of the query is used for communication with the 3494 or 3495.

LIBNAME *libraryname*
: specifies the library of interest for the Query LIBrary request. This parameter is not required if there is a single 3494 or 3495, or if the real address of a library device has been specified with the RDEV keyword.

NOWAIT
: gives control to the command issuer as soon as the Query LIBrary request is received by DFSMSRM, thereby creating asynchronous handling of the functions. Use this option when you do not want to wait for request processing to complete. This keyword is the default and cannot be used with the WAIT keyword.

WAIT
: allows the command issuer to wait until request processing has been completed, thereby creating synchronous handling of the functions. Control is not returned to the command issuer until processing has been completed by DFSMSRM. This keyword cannot be used with the NOWAIT keyword.

NODETails
indicates that details need not be provided in connection with this OPState, VOLume, or DEVice query. This keyword is the default and cannot be used with the DETails keyword.

DETails
requests that details be provided with this query. Such details are sent to the requester in a reader file and augment the standard synchronous message that is sent in response to the request. This option is available only for OPState, VOLume, and DEVice queries, and if requested in connection with a CATegory, INVentory, or COUnt query, it is ignored. This keyword can be used with the neither the AUDIT nor NODETails keyword.

ASSIGN
indicates that the device will be assigned to the system as part of device attach performed by the RMS master (to itself or to the requested user ID upon request completion.) This keyword cannot be used with the NOASSIGN keyword; it is the default if neither is specified and is the normal operational approach.

This operand is ignored when enhanced library control is available, except for Query Library Device and Query Library Volume with Audit option.

NOASSIGN
indicates that attach operations performed by the RMS master must not affect device assignment to the system. When this option is used, the requester is responsible for issuing the appropriate ASSIGN and UNASSIGN channel control words. While this capability is not part of typical library-usage practices, it provides flexibility when connecting library devices to multiple systems and sharing library resources. This keyword cannot be used with the ASSIGN keyword.

This operand is ignored when enhanced library control is available, except for Query Library Device and Query Library Volume with Audit option.

Usage Notes

1. The CSL routine FSMRMQLB provides a programming interface for querying basic library information. This routine does not support detailed queries, library inventory, or request-ID queries.
2. When using this command to query either a library device or volume with the AUDIT option specified, note that if, prior to issuing the mount command, the real device has been attached using the multiuser option to the user issuing the mount request, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.
3. When using this command to query a volume, the following media type values may be returned:
   - 160M for standard 3480 (MEDIA1)
   - 320M for extended 3480 (MEDIA2)
   - 10GB for standard 3590 (MEDIA3)
   - 20GB for extended 3590 (MEDIA4)
   - 300GB for 3592 Enterprise Tape Generation 1 (MEDIA5)
   - 300W for 3592 WORM Enterprise Tape Generation 1 (MEDIA6)
   - 60GB for 3592 Short Enterprise Tape Generation 1 (MEDIA7)
   - 60W for 3592 Short WORM Enterprise Tape Generation 1 (MEDIA8)
   - 700GB for 3592 Extended Enterprise Tape Generation 2 (MEDIA9)
   - 700W for 3592 Extended WORM Enterprise Tape Generation 2 (MEDIA10)
   - 880M for 3592 Enterprise Tape Generation 3 (MEDIA11)
   - 880W for 3592 WORM Enterprise Tape Generation 3 (MEDIA12)
   - 211M for 3592 Short Enterprise Tape Generation 3 (MEDIA13)

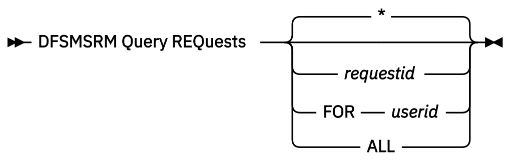
DFSMSRM Query REQuests

The DFSMSRM Query REQuests command allows you to:

- Query the status of this particular request
- Query all requests submitted by a specific user (except for STOP requests)
- Query the status of all DFSMSRM requests (except for STOP requests)

The command syntax of the DFSMSRM Query REQuests command is:

The DFSMSRM Query REQuests Command



Operands

*****
All requests for the command issuer are to be queried. This is the default for Query REQuests if no options are specified.

requestid
allows you to query the status of this particular request.

FOR *userid*
allows you to query all of the requests submitted by a specific user. If you do not have the authority to query the requests of this user, the command does not pass authorization checking. To learn how to acquire authorization to query the requests of a specific user, refer to "Authorizing Users" on page 14

ALL
allows you to query the status of all DFSMSRM requests that you are authorized to check.

Usage Notes

1. The DFSMSRM Query REQuests command is handled synchronously; that is, the command does an implicit wait until processing is complete, at which time a response message is issued to the console of the requester. Figure 8 on page 49 shows sample response messages.

```
FSMBCL3268I  REQUEST  ISSUED BY    TIME      STATUS OF REQUEST  COMMAND TEXT
FSMBCP3269I      303    OPERATOR 05:24:55              ACTIVE   RM DEMOUNT RDEV ED0
FSMBCP3269I      315    OPERATOR 05:29:44              ACTIVE   RM MOUNT VOL 001440
FSMBCP3269I      318    MAINT    05:33:54              ACTIVE   RM QUERY LIB OPS VMAUTO1
FSMBCP3269I      325    VMUSER   05:38:03              ACTIVE   RM QUERY LIB DEVICE D0F
FfMBCP3269I      337    MAINT    05:40:32              ACTIVE   RM QUERY LIB VOL 001445
FSMBCP3269I      436    OPERATOR 14:07:11              ACTIVE   RM MOUNT CAT SCRATCH1
FSMBCP3269I      439    OPERATOR 14:08:12              ACTIVE   RM QUERY LIB COUNT VMAUTO1
FSMBCL3276W No request found for user VMUSER1 or issuer not authorized
FSMBCL3277W No request found or issuer not authorized
FSMBCL3271W QUERY REQUEST 00000003 not found
```

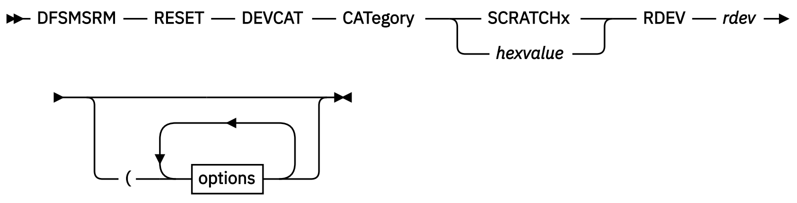
Figure 8. Sample Response Messages for Query REQuests

DFSMSRM RESET DEVCAT

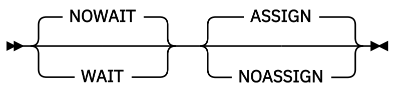
The DFSMSRM RESET DEVCAT command allows you to disassociate a category that has been assigned to a library device.

The command syntax of the DFSMSRM RESET DEVCAT command is:

DFSMSRM RESET DEVCAT Command



options



Operands

CATegory
allows you to determine how the selected volume is recategorized. You have two choices:

SCRATCH*x*
specifies which of the sixteen scratch pools becomes disassociated with the specified real device. If this scratch pool is not currently assigned to the specified device, the request fails.

hexvalue
is a hexadecimal category in the range 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM redbooks, should be performed prior to using as a category.

RDEV *rdev*
is the real address of the device that is disassociated from the specified category. If the specified category is not currently assigned to the specified device, the request fails.

Options

NOWAIT
gives control to the command issuer as soon as the request is received by DFSMSRM, thereby creating asynchronous handling of the functions. Use this option when you do not want to wait for request processing to complete. This keyword is the default and cannot be used with the WAIT keyword.

WAIT
allows the command issuer to wait until request processing has been completed, thereby creating synchronous handling of the functions. Control is not returned to the command issuer until processing has been completed by DFSMSRM. This keyword cannot be used with the NOWAIT keyword.

ASSIGN
indicates that the device will be assigned to the system as part of device attach performed by the RMS master (to itself or to the requested user ID upon request completion). This keyword cannot be used with the NOASSIGN keyword; it is the default if neither is specified and is the normal operational approach.

NOASSIGN
indicates that attach operations performed by the RMS master must not affect device assignment to the system. When this option is used, the requester is responsible for issuing the appropriate ASSIGN and UNASSIGN channel control words. While this capability is not part of typical library-usage

practices, it provides flexibility when connecting library devices to multiple systems and sharing library resources. This keyword cannot be used with the ASSIGN keyword.

Usage Notes

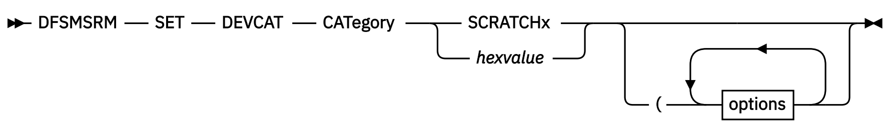
1. The CSL routine FSMRMRDC provides a programming interface equivalent to the DFSMSRM RESET DEVCAT command.
2. If, prior to issuing the mount command, the real device has been attached using the multiuser option to the user issuing the mount request, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.
3. For all RESET DEVCAT requests that are successfully completed, the following information is included in the final response:
   - The name of the scratch pool disassociated with a device
   - The real address of the device disassociated with a scratch pool
   - The library name

DFSMSRM SET DEVCAT

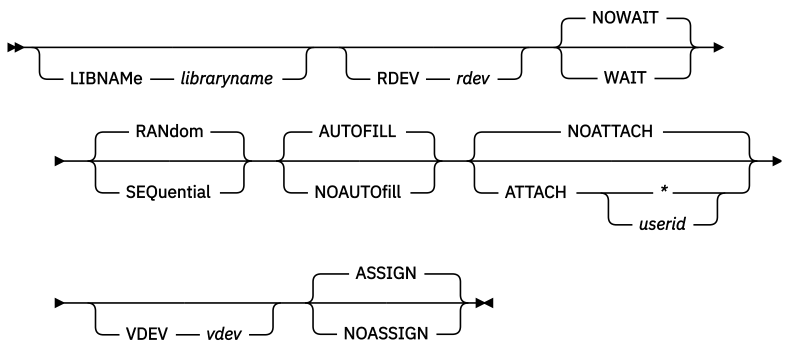
The DFSMSRM SET DEVCAT command allows you to assign a specified category to a library device. The device can either be specified by the requester or randomly selected by RMS.

The command syntax of the DFSMSRM SET DEVCAT command is:

DFSMSRM SET DEVCAT Command



options



Operand

CATegory

allows you to determine how the selected volume is recategorized. You have two choices:

SCRATCH*x*

is the name of the scratch pool to be assigned to a library device, where *x* is a value from 0–9 or A–F.

hexvalue

is a hexadecimal category in the range 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM redbooks, should be performed prior to using as a category.

Options

LIBNAME *libraryname*

specifies the library of interest when multiple libraries are attached to the system. If there is a single 3494 or 3495 or if a real address of a library device is specified with the RDEV keyword, then this parameter is not required.

RDEV *rdev*

is the real address of a device in the library of interest. This address must be available for attachment to the RMS master. This designation not only clarifies the library of interest when there are multiple 3494s or 3495s connected, but it also provides a real device path for use by the RMS master in communicating with the library manager for the category change operation. If this parameter is not provided, the RMS master finds an unattached device (in the library specified with the LIBNAME parameter or, when there are multiple 3494s or 3495s, in the default library) to use as a communication path for the SET request. If such a device is not found, the request fails.

NOWAIT

gives control to the command issuer as soon as the request is received by DFSMSRM, thereby, creating asynchronous handling of the functions. Use this option when you do not want to wait for request processing to complete. This keyword is the default and cannot be used with the WAIT keyword.

WAIT
allows the command issuer to wait until request processing has completed, thereby, creating synchronous handling of the functions. Control is not returned to the command issuer until processing has been completed by DFSMSRM. This keyword cannot be used with the NOWAIT keyword.

RANdom
indicates that volumes are loaded in random order in the integrated cartridge loader on the device. A particular category, for example, a scratch pool, may be set on multiple devices when random order is used. This keyword is the default and cannot be used with the SEQuential keyword.

SEQuential
indicates that volumes are loaded in category sequence order when the volume loader on the device is filled. A category can be set on only one device if category sequence order is used. This keyword cannot be used with the RANdom keyword.

AUTOFILL
automatically keeps the index stack of the loader filled with volumes from the specified category. This keyword is the default and cannot be used with the NOAUTOfill keyword.

When a device has been assigned a category and the AUTOFILL option is used, any subsequent MOUNT or DEMOUNT commands issued to or for the device will be rejected by the Automated Tape Library — as this is inconsistent with having tapes mounted automatically. Use the DFSMSRM RESET DEVCAT command to turn off the AUTOFILL option and restore normal MOUNT/DEMOUNT processing.

NOAUTOfill
does *not* automatically keep the index stack of the loader filled with volumes from the specified category. This keyword cannot be used with the AUTOFILL keyword.

NOATTACH
indicates that the device is not attached to a user once the operation is complete. This keyword is the default and cannot be used with the ATTACH keyword.

ATTACH ** or userid*
indicates that the device is attached to the requester (or specified user ID) once the SET DEVCAT operation is complete. The requester (*) is the default if a user ID is not specified. This keyword cannot be used with the NOATTACH keyword.

VDEV *vdev*
attaches the device to the requester (or alternative user ID) at the specified virtual address. This keyword is valid only when the ATTACH option is explicitly or implicitly specified. For information on what happens if this option is not specified, refer to "Usage Notes" on page 53.

ASSIGN
indicates that the device will be assigned to the system as part of device attach performed by the RMS master (to itself or to the requested user ID upon request completion.) This keyword cannot be used with the NOASSIGN keyword; it is the default if neither is specified and is the normal operational approach.

NOASSIGN
indicates that attach operations performed by the RMS master must not affect device assignment to the system. When this option is used, the requester is responsible for issuing the appropriate ASSIGN and UNASSIGN channel control words. While this capability is not part of typical library-usage practices, it provides flexibility in connecting library devices to multiple systems and sharing library resources. This keyword cannot be used with the ASSIGN keyword.

Usage Notes

1. The CSL routine FSMRMSDC provides a programming interface equivalent to the DFSMSRM SET DEVCAT command.
2. If, prior to issuing the mount command, the real device has been attached using the multiuser option to either the user issuing the mount request or to the user identified with the attach option, then

RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.

3. If the VDEV option is not specified, the virtual address used when the device is attached to the requestor or alternative user ID is determined as follows.
   - If the device was originally attached to the requestor, the device's virtual address is preserved.
   - If the device was *free*, the device's real address is used as the virtual address.

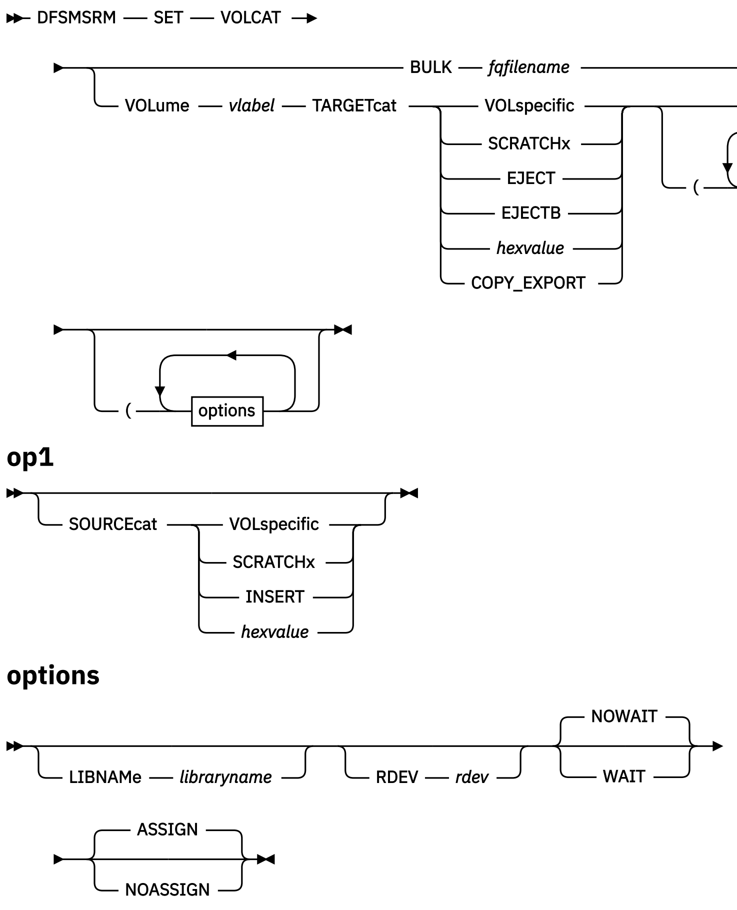
DFSMSRM SET VOLCAT

The DFSMSRM SET VOLCAT command allows you to assign a specified volume to a category. The SET VOLCAT command can also be used to initiate a Copy Export operation which allows a copy of selected logical volumes written to the TS7700 to be removed and taken offsite for disaster recovery purposes.

The SET VOLCAT BULK form of this command allows you to establish a list of volume-category relationships by means of a single command invocation. (The list must be in a preedited file and must subscribe to the required format as described in Chapter 4, “Creating DFSMS/VM RMS Bulk Processing Files,” on page 23.)

The command syntax for the DFSMSRM SET VOLCAT command is:

DFSMSRM SET VOLCAT Command



Operands

fqfilename
is the fully-qualified file name (including directory name unless the RM_WORK_DIRECTORY is to be used) for the Bulk Library Processing List.

VOLume *vlabel*
is the up-to-six-character, exterior label of the specific volume to undergo a category change.

TARGETcat
allows you to determine how the selected volume is recategorized. The target category is the category in which the volume is placed. You have six choices:

VOLspecific
classifies the selected volume as volume-specific; that is, the volume is not to be a member of a category.

SCRATCH*x*
selects one of sixteen possible scratch pools, where *x* is a value from 0–9 or A–F.

EJECT
removes the specified volume from the 3494 or 3495 by means of the convenience input/output station.

EJECTB
removes the specified volume from the 3494 or 3495 by means of the high-capacity input/output station.

hexvalue
is a hexadecimal category in the range 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM Redbooks®, should be performed prior to using as a category.

COPY_EXPORT
initiates a Copy Export operation which allows a copy of selected logical volumes written to the TS7700 to be removed and taken offsite for disaster recovery purposes.

Note: copy_export can only be used as a target category. When used as a target category, a source category must not be specified.

Options

SOURCEcat
allows you to specify the source category for the volume to receive a category change. The source category is the category from which the volume is retrieved. You have four choices:

VOLspecific
indicates that the specified volume is currently categorized as volume-specific; that is, it is not a member of a logical group of volumes.

SCRATCH*x*
indicates that the volume currently belongs to this scratch pool, where *x* is a value from 0–9 or A–F.

INSERT
indicates that the volume category is to be changed only if it is currently in the INSERT category (new to the library.)

Note: INSERT as a source category is not supported by a PtP VTS.

hexvalue
is a hexadecimal category in the range 0000 to FFFF. Passing a hexadecimal category should be done with extreme caution. Review of hexadecimal categories, as documented in IBM redbooks, should be performed prior to using as a category.

LIBNAME *libraryname*
specifies the library of interest when multiple libraries are attached to the system. If there is a single 3494 or 3495 or if a real address of a library device has been specified with the RDEV keyword, this parameter is not required. (Processing for more than one 3494 or 3495 cannot be intermixed in a single Bulk Library Processing List.)

RDEV *rdev*
is the real address of a device in the library of interest. This address must be available for attachment to the RMS master. This designation not only clarifies the library of interest when there are multiple 3494s or 3495s connected, but it also provides a real device path for use by the RMS master in communicating with the library manager for the category change operation. If this parameter is not provided, the RMS master attempts to find an unattached device (in the library specified with the LIBNAME parameter or, when there are multiple 3494s or 3495s, in the default library) to use as a communication path for the SET request. If such a device is not found, the request fails.

When the enhanced library control is being used, no device is needed to satisfy a SET VOLCAT request.

If RDEV is provided when enhanced library control is in use, the drive specified must be a valid address, although this drive is not necessarily used to satisfy the request.

NOWAIT
gives control to the command issuer as soon as the request is received by DFSMSRM, thereby creating asynchronous handling of the functions. Use this option when you do not want to wait for request processing to complete. This keyword is the default and cannot be used with the WAIT keyword.

WAIT
allows the command issuer to wait until request processing has completed, thereby creating synchronous handling of the functions. Control is not returned to the command issuer until processing has been completed by DFSMSRM. This keyword cannot be used with the NOWAIT keyword.

ASSIGN
indicates that the device will be assigned to the system as part of device attach performed by the RMS master (to itself or to the requested user ID upon request completion.) This keyword cannot be used with the NOASSIGN keyword; it is the default if neither is specified and is the normal operational approach.

This operand is ignored when enhanced library control is available.

NOASSIGN
indicates that attach operations performed by the RMS master must not affect device assignment to the system. The requester thus has the capability to control assignment of the device. When this option is used, the requester is responsible for issuing the appropriate ASSIGN and UNASSIGN channel command words. While this capability is not part of typical library-usage practices, it provides flexibility when connecting library devices to multiple systems and sharing library resources. This keyword cannot be used with the ASSIGN keyword.

This operand is ignored when enhanced library control is available.

Usage Notes®

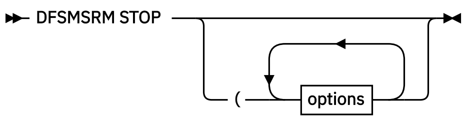
1. The CSL routine FSMRMSVC provides a programming interface equivalent for the DFSMSRM SET VOLCAT VOLume command.
2. The CSL routine FSMRMSVB provides a programming interface equivalent for the DFSMSRM SET VOLCAT BULK command.
3. TS7700 configurations do not support a target category change while the volume is mounted.
4. The COPY_EXPORT operand cannot be specified as a source category.

DFSMSRM STOP

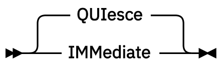
The DFSMSRM STOP command provides a comparable function to the DFSMS STOP command, supporting both immediate and quiesce termination scenarios for RMS master processing.

The command syntax of the DFSMSRM STOP command is:

The DFSMSRM STOP Command



options



Options

QUIesce
: initiates an orderly shutdown of library control operations. All in-process requests for MOUNT, DEMOUNT and SET operations are completed, but new requests for these operations will be rejected. DISCard, Query, and STOP commands are accepted until in-process library control commands are completed. The RMS master terminates processing and logs off five minutes after the STOP request is received.

IMMediate
: initiates an immediate shutdown of library control operations. All currently in-process library control activity is cancelled and requesters with in-process requests are notified of the cancellation. The RMS master terminates processing and logs off immediately.

Usage Notes

1. There is no CSL-routine programming interface equivalent for the DFSMSRM STOP command.
2. This command behaves like the DFSMS STOP command.
3. The messages issued by this command are similar to those messages for the DFSMS STOP command. Processing steps are simplified in comparison, because the RMS master does not dispatch work to other service machines.

Chapter 7. RMS Callable Services Library (CSL) Routines

PI

This chapter gives general information on callable services library (CSL) routines that are supplied with DFSMS/VM's RMS. These routines include:

- Programs that invoke CSL routines
- Synchronous and asynchronous handling of CSL requests
- Character parameters and compound variables
- Device attachment
- Return codes and reason codes

Important

Any userid using the CSL interface (as well as issuing a DFSMS or DFSMSRM command, or using the ISMF interface), must have an `IUCV ANY` statement in its directory. See *z/VM: CP Planning and Administration* for more information on including this statement in a user directory.

Thorough descriptions of each RMS CSL routine are also included in this chapter; each description includes the CSL routine's format, parameters, and reason codes. Below are brief descriptions of routines described in this chapter.

FSMRMDMT requests that the volume currently mounted on a specific device be demounted.

FSMRMMNT requests that a specific volume or the next volume in a designated category be mounted on a library device. A variety of mount options are provided for tailoring mount requests to meet specific needs.

FSMRMQLB requests information about specific 3494 or 3495 resources, including volumes, devices, categories, and overall inventory.

FSMRMRDC allows you to disassociate a device from a category.

FSMRMSDC establishes a relationship between a category and a device.

FSMRMSVC assigns a specified volume to a new or already defined category.

FSMRMSVB performs a list of volume assignments to specified categories.

If the real device was attached to the user prior to CSL routine issuance, it will be reattached upon completion of the request.

Warning

During RMS initialization, command request behavior may be unpredictable based on whether a device (if required) is available for processing the request.

Programs that Invoke CSL Routines

You can call CSL routines from a program that is written in any of these programming languages:

- Assembler
- C
- COBOL (IBM COBOL II and OS/VS COBOL Program Products)
- VS FORTRAN

- VS Pascal
- PL/I
- REXX

Appendix C, "Invoking CSL Routines," on page 105 shows some sample programs for invoking RMS CSL Routines.

Refer to *z/VM: CMS Callable Services Reference* for detailed instructions on invoking CSL routines from your application programs. Remember that the CSL library needs to be loaded on the virtual machine invoking the CSL routines. You can invoke the CSL routines by entering the following command:

```
'RTNLOAD * (FROM FSMPPSI'
```

You can add this command to the PROFILE EXEC of a virtual machine that uses RMS CSL routines.

Synchronous and Asynchronous Handling of CSL Requests

CSL requests for RMS functions can be handled either synchronously or asynchronously. Because RMS requests involve interaction with library components and, in some cases, require delayed responses from the hardware, wait times inherent in synchronous handling may be unacceptable for a caller in a multitasking environment. The RMS CSL calls include a *reqtoken* parameter that specifies whether a request is handled synchronously or asynchronously, and thus control processing is detailed as follows. **For synchronous processing**

The caller sets *reqtoken* to a binary 0 as input to the CSL routine. Processing is handled synchronously, and the results from processing are returned (as output parameters) on completion of a single call to the CSL. **For asynchronous processing**

The caller sets *reqtoken* to a binary 1 as input to the CSL to indicate that asynchronous processing is requested. The CSL returns a request token as output in the *reqtoken* parameter.

This token is a unique identifier for this request and can be used as input by the caller in making check-back (subsequent) calls by means of the same CSL. The caller, using a parameter list similar to the original but containing the request token, makes additional calls to the CSL until the response data is available (as indicated by return codes.) On check-back calls, only the request token must be specified; all other fields are ignored as input on check-back calls but are filled in with response data from the RMS master when the request is complete.

A caller making asynchronous RMS requests and checking back in a processing loop can make use of the CSL fast-path processing as described in *z/VM: CMS Application Development Guide for Assembler*.

Character Parameters and Compound Variables

The parameter lists for calling many of the RMS CSL routines include variable-length character parameters. Some parameters do not have a mandatory length or are specified in groups where options can be omitted—if defaults are desired or if there is one or more blanks before or after any of the individual members. In such cases, an integer-type length parameter follows the character parameter. For variable-length character parameters designated as optional in the CSL routine syntax descriptions, it is essential to also provide associated length parameter. In other words, the length parameter is not optional when the preceding character parameter is used.

Character parameters specified in a group are referred to as "compound variables." Typically, you are asked to select one parameter from one or more groups of options. These separate fields are combined into a compound character parameter for inclusion in the parameter list of the CSL routine. The combined length of these fields, including any leading, trailing, or intervening blanks, is specified as the length parameter that follows the compound character variable.

Device Attachment

Because library host control is integrated with 3490 control unit function, performing RMS functions that involve interaction with the 3495 requires attachment of a library device as part of the communication protocol.

For more information on device attachment and attach protocols, refer to "Designating a Target Device" on page 18.

Return Codes and Reason Codes

The RMS programming interface defined for RMS CSL routines includes two sets of return and reason codes:

1. Return and reason codes are defined as variables named *retcode* and *reascode* in each CSL parameter list and they describe the overall outcome of CSL processing. The information returned to the caller in these variables describes problems encountered during execution of the CSL routine and alerts the caller of any need for examining the second return and reason codes.

   A problem encountered by DMSCSL (for example, an incorrect parameter list) results in a return code with a negative value in the *retcode* field. These return codes are documented in *z/VM: CMS Callable Services Reference*. CSL calls made from a REXX program can generate additional return codes (with negative values); these are documented in *z/VM: REXX/VM Reference*.
2. The additional return-reason code pair is defined as *XXXXXrc* and *XXXXXrsn*, where *XXXXX* is a request-type identifier. These describe the status of the RMS operation processing performed by the RMS master during handling of the request. When abnormal conditions occur in handling an RMS operation, the precise reason is detailed. When a problem in CSL processing prevents the request from being transmitted to the RMS master, or when a requested asynchronous operation is still pending (as indicated by the CSL *retcode* and *reascode*), the *XXXXXrc* and *XXXXXrsn* are not set by CSL processing and should not be examined by the caller.

The overall processing steps for all RMS CSL routines is similar. Thus, the handling and specification of CSL return codes (*retcode*) and reason codes (*reascode*) are common among all RMS CSL routines, and are described in this section.

While there are many common processing steps for handling requests during processing in the RMS master, there are unique conditions based on request type. Thus, the status and reason codes associated with the RMS operation processing (*XXXXXrc* and *XXXXXrsn*) are listed individually for each CSL routine in the later parts of this chapter. Additionally, Appendix A, "Return Codes and Reason Codes for CSL Routines," on page 85 contains explanations of all possible *XXXXXrc* and *XXXXXrsn*.

Return Code Protocol

If the CSL routine encounters a problem, it reports an error return code. The conventions that apply in setting the CSL-routine return code variable, *retcode*, are listed in Table 7 on page 61.

Table 7. Return Code (retcode) Protocol

Return Code	**Explanation**
retcode = 0	Indicates that the RMS operation has completed successfully and output parameters are set for use by the caller. The caller does not need to perform any additional return or reason code checking.

Table 7. Return Code (retcode) Protocol (continued)

Return Code	**Explanation**
retcode = 2	Indicates that a request for asynchronous handling of an RMS operation has been successfully transmitted to the RMS master but processing is not yet complete. This return code constitutes a successful return from the initial invocation of an asynchronous request or, for a check-back call, indicates that the request is still in-process. No further return or reason code checking is required. The only output parameter that has been set for use by the calling routine is the request token (*reqtoken*) in asynchronous calls; this is used for subsequent check-back calls.
retcode = 4	Indicates that the RMS operation is complete but a warning condition has occurred during processing. The caller must check the function reason code (*XXXXXrsn*) for further details on the warning condition. Output parameters have been set for use by the calling routine.
retcode = 8	Indicates that an error condition has occurred, either during processing of the CSL routine or during RMS processing by the RMS master. If the variable *reascode* is nonzero, then the error has occurred in CSL processing and the reason code should be one of those listed in Table 10 on page 63. If the variable *reascode* is zero, then the error has occurred in RMS processing and the function reason code (*XXXXXrsn*, described for each CSL routine in Appendix A, "Return Codes and Reason Codes for CSL Routines," on page 85) should be examined by the caller. Output parameters have not been set for use by the calling routine.
retcode = 12	Indicates that a critical error condition has occurred during processing of the CSL routine and the variable *reascode* should be examined by the caller. Possible *reascode* values are listed in Table 10 on page 63. Output parameters may not have been set for use by the calling routine.

The status of the RMS operation processing performed by the RMS master during the handling of the request are reported through the *XXXXXrc*. The conventions that apply in setting the return code, *XXXXXrc*, are listed in Table 8 on page 63.

Table 8. Return Code (XXXXXrc) Protocol

Return Code	Explanation
XXXXXrc = 0	Good.
XXXXXrc = 4	Warning. There is an error, but the request has completed.
XXXXXrc = 8	Error; the request has failed.

Reason Code Protocol

Reason codes for the CSL routines range from 0000 to 4010. For ease of comprehension, these codes are divided, by their numbers, into seven groups, as illustrated in Table 9 on page 63.

Table 9. Reason Code Protocol

Reason Code	Explanation
XXXXXrsn = 0000–2999	Reports errors encountered in accepting the request for processing. These errors are detected by the RMS master.
XXXXXrsn = 3000–3099	Reports errors in the way the request was specified. These errors are detected by the RMS master.
XXXXXrsn = 3100–3199	Reports errors experienced by the RMS master. These errors prevent processing of the request.
XXXXXrsn = 3200–3299	Reports errors experienced after primary processing of the request has occurred. For example, the volume is successfully mounted, but Logical Write Protect cannot be set.
XXXXXrsn = 3300–3499	Reports completion status.
XXXXXrsn = 3500–3899	Reports reason codes for input/output errors reported by the hardware. Reason codes 3500–3599 are status codes that denote configuration-related problems; for example, a mount is issued to a device that is not in a library.
reascode = 3900–3999	Reports errors detected by the CSL itself. These errors are detected before the connection is made to the RMS master and are common to all CSL routines; they are not routine-specific. Possible *reascode* are listed and described in Table 10 on page 63.
reascode = 4000–4010	Reports bulk file processing errors.

Reascodes Common to All CSL Routines

Table 10 on page 63 describes the reason codes that may be returned in the variable called *reascode*. These reason codes, which describe the overall outcome of CSL processing, are common to all CSL routines; they are not routine-specific.

Table 10. Reason Code (reascode) Protocol

Return Code (*retcode*)	Reason Code (*reascode*)	Explanation
12	3900	Environment error—An error has occurred in the environment.
8	3904	Syntax error
8	3908	Invalid request token—The request token specified a negative value.

Table 10. Reason Code (reascode) Protocol (continued)

Return Code (*retcode*)	**Reason Code (*reascode*)**	**Explanation**
12	3912	Functional error—The CSL processor has experienced an internal error.
8	3916	Invalid volume label
8	3920	Invalid keyword
8	3924	Invalid library name
12	3928	Storage error
12	3932	Communication error
12	3936	CMS procedure error
8	3940	RMS master unavailable
8	3941	Check-back call failed
12	3944	READ_TERMINAL procedure error
12	3948	Corrupted master response
8	3952	Invalid target category name
8	3956	Invalid source category name
8	3960	Invalid user ID
8	3964	Invalid real device number
8	3968	Invalid virtual device number
8	3972	Invalid bulk list file identifier

FSMRMDMT – Library Demount

The FSMRMDMT CSL routine provides a programming interface equivalent to the DFSMSRM DEMOUNT command. With this routine, you can request that the volume currently mounted on a specific device be demounted.

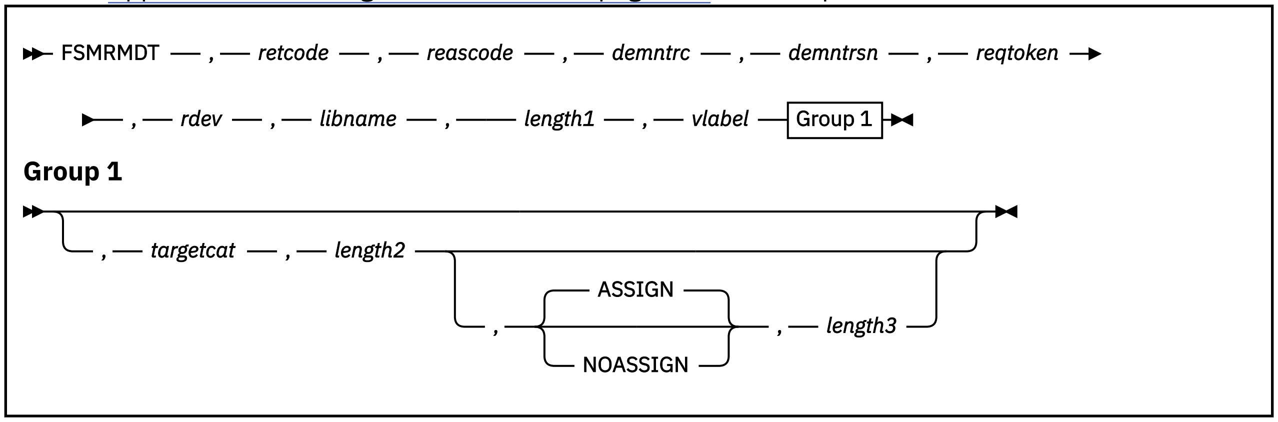
Normally, user-initiated demount operations are not required because volume demounts are handled during RMS processing. This routine, however, provides a means of "freeing-up" a volume that is currently unavailable for a requested operation because, for example, it may still be mounted from a prior request.

The volume label can be specified, in which case the demount operation is performed only if that volume is mounted on the specified device.

Call Format

The format for calling a CSL routine is language dependent. FSMRMDMT is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.



Parameters

FSMRMDMT
(input, CHAR, 8) is the name of the CSL routine being invoked. The value of FSMRMDMT can be passed either directly or in a variable.

retcode
(output, INT, 4) is a variable for the return code from FSMRMDMT CSL processing. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
(output, INT, 4) is a variable for the reason code from FSMRMDMT CSL processing. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

demntrc
(output, INT, 4) is a return code variable that describes the success of the demount operation. Refer to "Return Codes and Reason Codes" on page 61 for more information on these return codes.

demntrsn
(output, INT, 4) is a reason code variable that describes the success of the demount operation. Return codes that are applicable to the FSMRMDMT CSL routine are listed at the end of this CSL-routine description.

reqtoken
(input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. This output request token is to be used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the call to this routine is interpreted as a check-back call on an asynchronous request, and the contents of this variable is the request identifier.

rdev
(input, CHAR, 4) specifies the real address of the device from which a volume is demounted. This parameter is required.

libname
(input/output, CHAR, 1–32) specifies the name of the library in which the operation is requested. This parameter can be blank on input when a real device (*rdev*)h is provided, or when there is only one 3495 attached and the real address is blank. If this field is not provided as input, it is returned as output.

length1
(input, INT, 4) specifies the length of the preceding character parameter, library name. When the library name is returned as output, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated.

vlabel
(input/output, CHAR, 6) specifies the label of the volume to be demounted. If this parameter is specified, the request is processed only if the specified volume is mounted on the specified device; no other volume will be demounted. If this parameter is blank on input, the exterior label of the volume demounted is returned as output.

targetcat
(input, CHAR, 1–32) specifies the name of a target category to which the volume is set. The following special category names can be specified:

- SCRATCH*x*
- VOLspecific
- *hexvalue*

length2
(input, INT, 4) specifies the length of the preceding character parameter, target category.

`ASSIGN`
(input, CHAR, 6) indicates that the device attach operations that take place as part of processing include device assignment to the system.

`NOASSIGN`
(input, CHAR, 8) indicates that the device attach operations that take place as part of processing must not affect the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.)

length3
(input, INT, 4) specifies the length of the preceding character parameter, ASSIGN or NOASSIGN.

Usage Notes

1. If, prior to issuing the mount command, the real device has been attached using the multiuser option to the user issuing the mount request, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, “Return Codes and Reason Codes for CSL Routines,” on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

FSMRMMNT – Library Mount

The FSMRMMNT CSL routine provides a programming interface equivalent to the DFSMSRM MOUNT command. With this routine, you can request that a specific volume or the next volume in a designated category be mounted.

Mount options are available for tailoring mount requests to specific needs. For example:

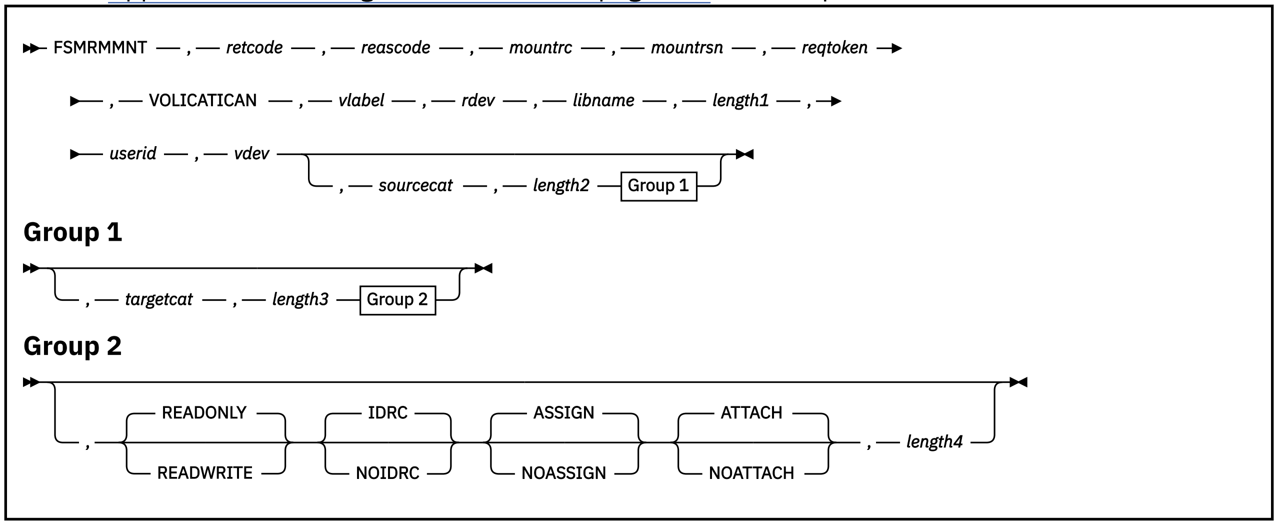
- The volume category can be set to a new category as part of the mount operation.
- The requester can indicate that the device is attached to a specified user ID (other than the requester) when the mount is complete.
- A volume can be mounted for read-only or for read/write.
- A virtual address for the device attach can be provided if the default is not desired.

A MOUNT request issued to the RMS master can be sent to a designated user ID if a manual library name is specified in the request.

Call Format

The format for calling a CSL routine is language dependent. FSMRMMNT is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.



Parameters

FSMRMMNT
(input, CHAR, 8) is the name of the CSL routine being invoked. The value FSMRMMNT can be passed either directly or in a variable.

retcode
(output, INT, 4) is a variable for the return code from FSMRMMNT. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
(output, INT, 4) is a variable for the reason code from FSMRMMNT. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

mountrc
(output, INT, 4) is a return code variable that describes the success of the mount operation. Refer to "Return Codes and Reason Codes" on page 61 for explanations of these return codes.

mountrsn
(output, INT, 4) is a reason code variable that describes the success of the mount operation. Return codes that are applicable to the FSMRMMNT CSL routine are listed at the end of this CSL-routine description.

reqtoken
(input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. The output request token is to be used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the call to this routine is interpreted as either a check-back or a cancel call on an asynchronous request, and the contents of this variable is the request identifier.

VOL
(input, CHAR, 3) indicates that the mount request is for a designated volume.

CAT
(input, CHAR, 3) indicates that the mount request is for the next volume in a designated category.

CAN
(input, CHAR, 3) indicates that the mount request identified by the request identifier in the reqtoken is to be cancelled.

vlabel
(input/output, CHAR, 6) specifies the label of the volume to be mounted, and is expected as input when VOLume is specified. When CAT is specified, the label of the mounted volume is provided here as output.

rdev
(input/output, CHAR, 4) specifies the real device address of the device to be mounted This parameter is an optional input parameter. If it is omitted as input, device selection is handled by the RMS master and this field is returned as output.

libname
(input/output, CHAR, 1–32) specifies the library name in which the mount operation is requested. This parameter is not required when a real device (*rdev*) is provided, nor is it required when there is only one 3495 attached and the real device address is not specified. If this field is not provided as input, it is returned as output.

length1
(input, INT, 4) specifies the length of the preceding character parameter, library name. When the library name is returned as an output field, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated.

userid
(input, CHAR, 8) specifies an alternate user ID to which the real device is attached if ATTACH is requested. If NOATTACH is specified, this parameter must be blank on input.

vdev
(input/output, CHAR, 4) specifies a virtual address to which the device is attached when the mount is complete. This parameter is applicable only when ATTACH is requested. If this parameter is blank on input, it is provided as output, with the same default attach protocols as those used for the MOUNT command.

sourcecat
(input, CHAR, 1–32) specifies the name of a source categroy from which the next volume is to be mounted when CAT is specified. The following special category names can be specified:

- SCRATCH*x*
- *hexvalue*

length2
(input, INT, 4) specifies the length of the preceding character parameter, source category.

targetcat
(input, CHAR, 1–32) specifies the name of a target category to which the volume is set. The following special category names can be specified:

- SCRATCH*x*
- VOLspecific

- *hexvalue*

length3
(input, INT, 4) specifies the length of the preceding character parameter, target category.

`READONLY`
(input, CHAR, 8) turns on the logical write protection for the device as part of the mount operation; the volume is to be mounted for read only. This default can be overridden by means of a control file parameter. This keyword is ignored if either the CAN or NOATTACH option is specified.

`READWRITE`
(input, CHAR, 9) turns off the logical write protection for the device as part of the mount operation; the volume is mounted for read and write. This keyword is ignored if the CAN request is specified.

`IDRC`
(input, CHAR, 4) turns on the IDRC capability of the device as part of the mount operation. This keyword is ignored if the CAN request is specified.

`NOIDRC`
(input, CHAR, 6) turns off the IDRC capability of the device as part of the mount operation. This NOIDRC option cannot be used with automated libraries. This keyword is ignored if the CAN request is specified.

`ASSIGN`
(input, CHAR, 6) indicates that the device attach operations that take place as part of processing include device assignment to the system. This keyword is ignored if the CAN request is specified.

`NOASSIGN`
(input, CHAR, 8) indicates that the device attach operations that take place as part of processing must not affect the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.) This keyword is ignored if the CAN request is specified.

`ATTACH`
(input, CHAR, 6) indicates that the device is attached to the requester's virtual machine, or to an optionally specified user ID, once the mount operation is complete. This keyword is ignored if the CAN request is specified.

`NOATTACH`
(input, CHAR, 8) indicates that the device not be attached to a virtual machine once the mount operation is complete. For a CAN request, the device specified or selected in the original mount request is not attached to a virtual machine.

length4
(input, INT, 4) specifies the length of the preceding compound character parameter (READONLY or READWRITE, IDRC or NOIDRC, ASSIGN or NOASSIGN, and ATTACH or NOATTACH). See "Character Parameters and Compound Variables" on page 60 for related information on compound variables.

Usage Notes

1. If, prior to issuing the mount command, the real device has been attached using the multiuser option to either the user issuing the mount request or to the user identified with the attach option, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, "Return Codes and Reason Codes for CSL Routines," on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

FSMRMQLB – Query Library Information

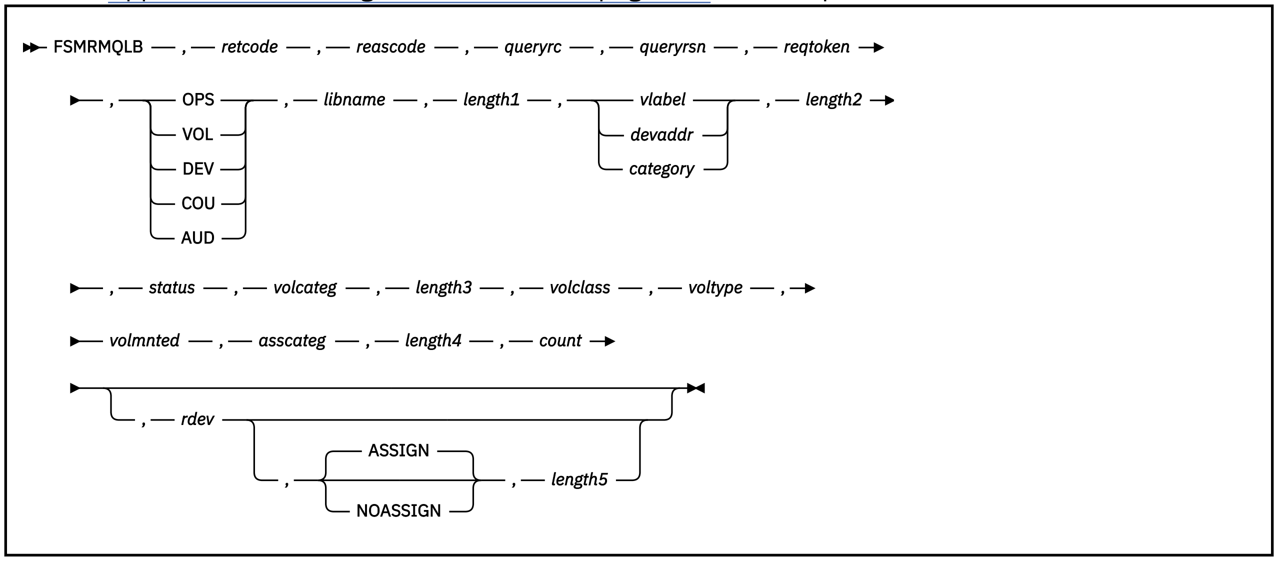
The FSMRMQLB CSL routine provides a programming interface similar to the DFSMSRM Query LIBrary command. With this routine, you can request information about 3494 or 3495 status and its resources. However, detailed information requested by means of the command interface and returned to the requester as a reader file is **not** available by means of the programming interface. The programming interface can return:

- Information about the 3494 or 3495 operational state
- Volume information (category, class, type, and status)
- Device information (volume mounted, category of volume mounted, category assigned, status, class, and type)
- Count of specified category or entire inventory
- Volume audit status (optical verification of physical location)
- Cache statistics (for a disk-only environment)

Call Format

The format for calling a CSL routine is language dependent. FSMRMQLB is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.



Parameters

FSMRMQLB
: (input, CHAR, 8) is the name of the CSL routine being invoked. The value FSMRMQLB can be passed either directly or in a variable.

retcode
: (output, INT, 4) is a variable for the return code from FSMRMQLB. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
: (output, INT, 4) is a variable for the reason code from FSMRMQLB. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

queryrc
: (output, INT, 4) is a return code variable describing the success of the query operation. Refer to "Return Codes and Reason Codes" on page 61 for more information on these return codes.

queryrsn
: (output, INT, 4) is a reason code variable describing the success of the query operation. Return codes that are applicable to the FSMRMQLB CSL routine are listed at the end of this CSL-routine description.

reqtoken
: (input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. This output request token is used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the call to this routine is interpreted as a check-back call on an asynchronous request, and the contents of this variable is the request identifier.

OPS
: (input, CHAR, 3) indicates that the 3494 or 3495 operational state is queried.

VOL
: (input, CHAR, 3) indicates that a specific volume is queried. RMS will search for the volume in all defined libraries until found, starting with either the library specified on the CSL request or the first library defined to RMS.

DEV
: (input, CHAR, 3) indicates that a specific real device is queried.

COU
: (input, CHAR, 3) requests a volume count of a category (or complete inventory.)

AUD
: (input, CHAR, 3) indicates that a specific volume be physically audited (other volume status and descriptive data is not provided.)

libname
: (input/output, CHAR, 1–32) specifies the library name in which the operation is requested. This parameter is not required when a real device (*rdev*) is provided, nor is it required when there is only one 3495 attached and the real address is not specified. The library name returned will be the library in which the volume was found to reside, which may be different from the library name provided as input. If the request is successful (Return Code=0 and Reason Code=0), then programs using this CSL to determine the library in which a volume resides should check this field before proceeding with further requests.

length1
: (input, INT, 4) specifies the length of the preceding character parameter, library name. When a library name is returned as an output field, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated.

vlabel* or *devaddr* or *category
: (input, CHAR, *length2*) specifies either the label of the volume to be queried when VOL or AUD is requested, the real device address of the DEVice being queried, or the name of the category for which a COUnt is desired. When the query type is COUnt and an entire library inventory count is desired, omit the category name by specifying a length of 0 in the *length2* parameter. Otherwise, you can specify a count for the INSERT or SCRATCH*x* category.

length2
: (input, INT, 4) specifies the length of the preceding character parameter, either a volume label, real device address, or category name. When the query type is COUnt and an entire library inventory count is desired, omit the category name by specifying a length of 0 in the *length2* parameter.

status
: (output, BIT, 16) returns the status of the queried object. Table 11 on page 73 shows the status conditions returned in this field for the various query types.

volcateg
: (output, CHAR, *length3*) returns a category name. If a volume is queried, the volume's category name is returned. If a device is queried the category of the mounted volume is returned.

length3
: (input, INT, 4) specifies the length of the preceding character parameter, volume category. When the volume-category name is returned as an output field, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated. Specifying a length of 0 has the effect of omitting this parameter for calls in which the preceding parameter is not used.

volclass
: (output, CHAR, 4) returns the name of either the volume class (if querying a volume using the VOL option) or the device class (if querying a specific real device using the DEV option), as maintained by the library manager. This field is blank upon return if an object other than a volume or device is being queried.

voltype
: (output, CHAR, 4) returns the name of either the volume type (if querying a volume using the VOL option) or the device type (if querying a specific real device using the DEV option), as maintained by the library manager. This field is blank upon return if an object other than a volume or device is being queried.

volmnted
: (output, CHAR, 6) returns the label of the volume mounted on a device being queried. This field is blank upon return if the queried object is not a device, or if there are no mounted volumes on the device being queried.

asscateg
: (output, CHAR, *length4*) returns the name of the category that is assigned to a device being queried.

length4
: (input, INT, 4) specifies the length of the preceding character parameter, assigned category. When the assigned-category name is returned as an output field, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated. Specifying a length of 0 has the effect of omitting this parameter for calls in which the preceding parameter is not used.

count
: (output, INT, 4) is the variable for returning either the number of volumes in a specified category or complete inventory (if querying a volume count using the COU option) or the percentage of cache storage in use for a disk-only environment (if querying the operational state using the OPS option). In the latter case, a zero will be returned when querying the operational state if the library is *not* a disk-only environment.

rdev
: (input, CHAR, 4) specifies the real device address used to communicate with the library in processing the request. This parameter is an optional input parameter. If this option is omitted, any available device is used. When you query a device, do not specify this parameter because the device address specified as the object of the query is used for library communication. This operand is not required when the enhanced library control is available. If the operand is specified, the address must be valid.

`ASSIGN`
: (input, CHAR, 6) indicates that the device attach operations that take place as part of processing include device assignment to the system. Except for Query Library Device and Query Library Volume with the Audit option, this parameter is ignored when enhanced library control is available.

`NOASSIGN`
: (input, CHAR, 8) indicates that the device attach operations that take place as part of processing must not affect the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.) Except for Query Library Device and Query Library Volume with the Audit option, this parameter is ignored when enhanced library control is available.

length5
: (iput, INT, 4) specifies the length of the preceding character parameter, ASSIGN or NOASSIGN.

Output

Table 11 on page 73 details the output returned by the various types of queries.

Table 11. Output Returned by Query Type

Type	Status Conditions	Other Output
OPState	X'0000'—Automated mode X'0100'—Paused mode X'0200'—Manual mode	Cache statistics (Disk only)
VOLume	X'0000'—No Special Conditions X'8000'—Inaccessible X'4000'—Mounted X'2000'—Queued For Mount X'1000'—Being Mounted X'0800'—Queued For Demount X'0400'—Being Demounted X'0200'—Queued For Eject X'0100'—Being Ejected X'0080'—Queued For Audit X'0040'—Being Audited X'0020'—Misplaced X'0010'—Missing or Damaged Label X'0008'—Used in Manual Mode X'0004'—Manually Ejected X'0002'—Assigned with the Fast Ready Attribute Set	Category Class Type
DEVice	X'0000'—Installed and available X'0100'—Not installed or unavailable	Label, mounted volume Category, mounted volume Category, assigned to device Class Type
COUnt	Not applicable	Count
AUDit	Not applicable	Status passed in *queryrsn*

Usage Notes

1. When using this routine to query either a library device or volume with the AUDIT option specified, note that if, prior to issuing the mount command, the real device has been attached using the multiuser option to the user issuing the mount request, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.
2. When using this routine to query a volume, the following media type values may be returned:
   - 160M for standard 3480 (MEDIA1)
   - 320M for extended 3480 (MEDIA2)
   - 10GB for standard 3590 (MEDIA3)
   - 20GB for extended 3590 (MEDIA4)

- 300GB for 3592 Enterprise Tape Generation 1 (MEDIA5)
- 300W for 3592 WORM Enterprise Tape Generation 1 (MEDIA6)
- 60GB for 3592 Short Enterprise Tape Generation 1 (MEDIA7)
- 60W for 3592 Short WORM Enterprise Tape Generation 1 (MEDIA8)
- 700GB for 3592 Extended Enterprise Tape Generation 2 (MEDIA9)
- 700W for 3592 Extended WORM Enterprise Tape Generation 2 (MEDIA10)
- 880M for 3592 Enterprise Tape Generation 3 (MEDIA11)
- 880W for 3592 WORM Enterprise Tape Generation 3 (MEDIA12)
- 211M for 3592 Short Enterprise Tape Generation 3 (MEDIA13)

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, “Return Codes and Reason Codes for CSL Routines,” on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

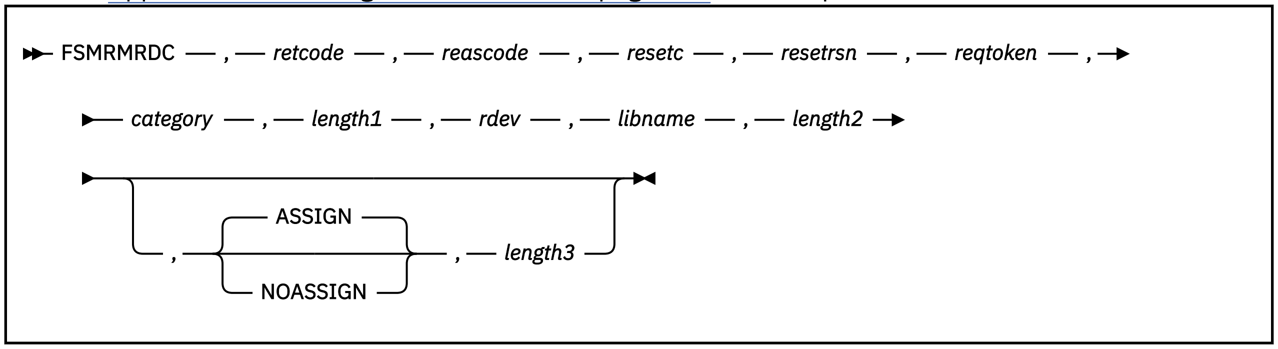
FSMRMRDC – Reset Device Category

The FSMRMRDC CSL routine provides a programming interface equivalent to the DFSMSRM RESET DEVCAT command. With this routine, you can request that the category currently assigned to a library device (by means of the SET DEVCAT function) is now disassociated from that device. The automatic volume loader on the library device is no longer filled with volumes from the specified category.

Call Format

The format for calling a CSL routine is language dependent. FSMRMRDC is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.



Parameters

FSMRMRDC
: (input, CHAR, 8) is the name of the CSL routine being invoked. The value FSMRMRDC can be passed either directly or in a variable.

retcode
: (output, INT, 4) is a variable for the return code from FSMRMRDC. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
: (output, INT, 4) is a variable for the reason code from FSMRMRDC. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

resetrc
: (output, INT, 4) is a return code variable describing the success of the reset operation. Refer to "Return Codes and Reason Codes" on page 61 for more information on these return codes.

resetrsn
: (output, INT, 4) is a reason code variable describing the success of the reset operation. Return codes that are applicable to the FSMRMRDC CSL routine are listed at the end of this CSL-routine description.

reqtoken
: (input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. The output request token is used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the call to this routine is interpreted as a check-back call on an asynchronous request, and the contents of this variable is the request identifier.

category
: (input, CHAR, 1–32) specifies the name of the category that is currently assigned to the device but becomes disassociated from the device as a result of this request. The following category names can be specified:

- SCRATCH*x*
- *hexvalue*

length1
: (input, INT, 4) specifies the length of the preceding character parameter, category name.

rdev
: (input, CHAR, 4) specifies the real device address of the device to be disassociated with the assigned category.

libname
: (input/output, CHAR, 1–32) specifies the library name in which the operation is requested. This parameter is not required as input because the real device (*rdev*) is also required and pinpoints the library. However, the library name in which the operation is performed is provided as output, so that this field must be included.

length2
: (input, INT, 4) specifies the length of the preceding character parameter, library name. Because the library name is returned as an output field, it is important that the length be adequate; use of the maximum length (32 bytes) prevents returned information from being truncated.

ASSIGN
: (input, CHAR, 6) indicates that the device attach operations that take place as part of processing should include device assignment to the system.

NOASSIGN
: (input, CHAR, 8) indicates that the device attach operations that take place as part of processing must not have any affect on the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.)

length3
: (input, INT, 4) specifies the length of the preceding character parameter, ASSIGN or NOASSIGN.

Usage Notes

1. If, prior to issuing the mount command, the real device has been attached using the multiuser option to the user issuing the mount request, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, “Return Codes and Reason Codes for CSL Routines,” on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

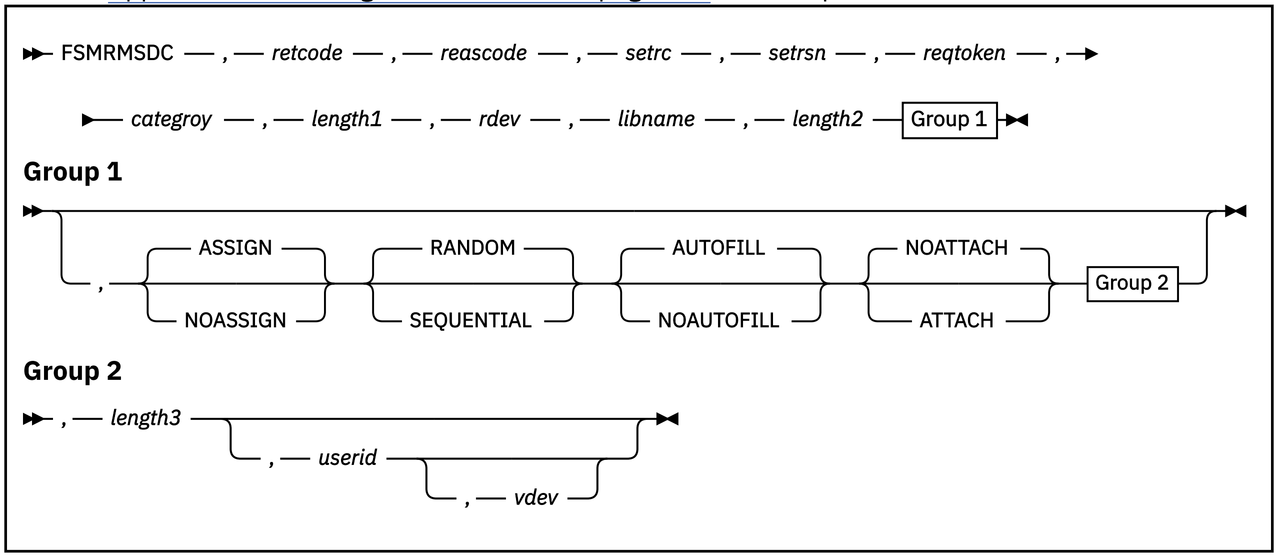
FSMRMSDC – Set Device Category

The FSMRMSDC CSL routine provides a programming interface equivalent to the DFSMSRM SET DEVCAT command. With this routine, you can assign a specific category to a library device. The device can either be specified by the requester or selected during RMS processing. The library device's automatic volume loader is filled with volumes from the specified category.

Call Format

The format for calling a CSL routine is language dependent. FSMRMSDC is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.


FSMRMSDC , retcode , reascode , setrc , setrsn , reqtoken ,
categroy , length1 , rdev , libname , length2 Group 1
Group 1
, ASSIGN NOASSIGN RANDOM SEQUENTIAL AUTOFILL NOAUTOFILL NOATTACH ATTACH Group 2
Group 2
, length3 , userid , vdev

Parameters

FSMRMSDC
: (input, CHAR, 8) is the name of the CSL routine being invoked. The value FSMRMSDC can be passed either directly or in a variable.

retcode
: (output, INT, 4) is a variable for the return code from FSMRMSDC. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
: (output, INT, 4) is a variable for the reason code from FSMRMSDC. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

setrc
: (output, INT, 4) is a return code variable describing the success of the operation. Refer to "Return Codes and Reason Codes" on page 61 for more information on these return codes.

setrsn
: (output, INT, 4) is a reason code variable describing the success of the operation. Return codes that are applicable to the FSMRMSDC CSL routine are listed at the end of this CSL-routine description.

reqtoken
: (input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. The output request token is to be used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the call to this routine is interpreted as a check-back call on an asynchronous request, and the contents of this variable is the request identifier.

category
(input, CHAR, 1–32) specifies the name of the category assigned to the device. The following can be specified for category:

- SCRATCH*x*
- *hexvalue*

length1
(input, INT, 4) specifies the length of the preceding character parameter, category.

rdev
(input/output, CHAR, 4) specifies the real device address of the device associated with the specified category. If this parameter is omitted as input, device selection is handled by the RMS master and this field is returned as output.

libname
(input/output, CHAR, 1–32) specifies the library name in which the operation is requested. This parameter can be blank when a real device (*rdev*) is provided, or when there is only one 3495 attached and the real address is not specified. If this field is not provided as input, it is returned as output.

length2
(input, INT, 4) specifies the length of the preceding character parameter, library name. When library name is returned as an output field, it is important that the length be adequate; use of the maximum length (32 bytes) prevents returned information from being truncated.

ASSIGN
(input, CHAR, 6) indicates that device attach operations that take place as part of processing include device assignment to the system. (This is the standard attachment technique.)

NOASSIGN
(input, CHAR, 8) indicates that device attach operations that take place as part of processing must not affect the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.)

RANDOM
(input, CHAR, 6) indicates that volumes are to be loaded into the automatic volume loader in random order. A particular category can be assigned to multiple devices when random order is specified.

SEQUENTIAL
(input, CHAR, 10) indicates that volumes are to be loaded into the automatic volume loader in sequence order. A particular category can be assigned to only one device when sequence order is used.

AUTOFILL
(input, CHAR, 8) indicates that the automatic volume loader be automatically filled with volumes from the specified category.

NOAUTOFILL
(input, CHAR, 10) indicates that the automatic volume loader not be automatically filled with volumes from the specified category. A particular volume is not reloaded after it cycles through the stacker and mounts.

NOATTACH
(input, CHAR, 8) indicates that the device not be attached to a virtual machine after the operation is complete.

ATTACH
(input, CHAR, 6) indicates that the device be attached to the requester's virtual machine, or to an optionally specified user ID, after the operation is complete.

length3
(input, INT, 4) specifies the length of the preceding compound character parameters (ASSIGN or NOASSIGN, RANDOM or SEQUENTIAL, AUTOFILL or NOAUTOFILL, and NOATTACH or ATTACH). See "Character Parameters and Compound Variables" on page 60 for information on compound variables.

userid
: (input, CHAR, 8) specifies an alternate user ID to which the real device be attached upon completion of the operation. If this parameter is omitted or blank on input when ATTACH is specified, the user ID defaults to the requester's user ID. This parameter is valid only if the ATTACH option is requested.

vdev
: (input/output, CHAR, 4) specifies a virtual address for attachment of the real device after the operation is complete. This parameter is valid only if the ATTACH option has been requested. If this parameter is omitted or is blank on input, the device is attached at a virtual address consistent with default processing described for the SET DEVCAT command. If included in the parameter list, but left blank on input, the virtual address is returned as output.

Usage Notes

1. If, prior to issuing the mount command, the real device has been attached using the multiuser option to either the user issuing the mount request or to the user identified with the attach option, then RMS will also attach the device using the multiuser option. This allows the device to be attached simultaneously to both the RMS master and to another userid. Be aware, though, that while the RMS master has the device attached, other userids should not attempt to use the device.

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, "Return Codes and Reason Codes for CSL Routines," on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

FSMRMSVC – Set Volume Category

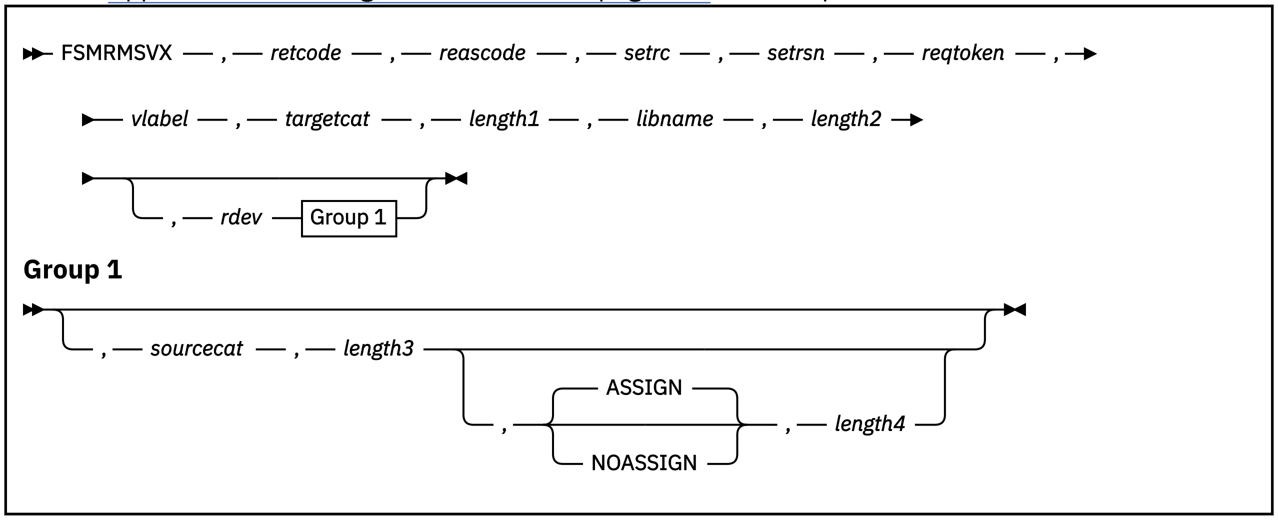
The FSMRMSVC CSL routine provides a programming interface equivalent to the DFSMSRM SET VOLCAT command. With this routine, you can request that:

- A specific volume be assigned to a designated category.
- A specific volume be removed from the library enclosure.
- A new library volume now in the INSERT category be assigned to a another category
- Initiate® a Copy Export operation which allows a copy of selected logical volumes written to the TS7700 to be removed and taken offsite for disaster recovery purposes.

Call Format

The format for calling a CSL routine is language dependent. FSMRMSVC is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.



Parameters

FSMRMSVC
: (input, CHAR, 8) is the name of the CSL routine being invoked. The value FSMRMSVC can be passed either directly or in a variable.

retcode
: (output, INT, 4) is a variable for the return code from FSMRMSVC. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
: (output, INT, 4) is a variable for the reason code from FSMRMSVC. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

setrc
: (output, INT, 4) is a return code variable describing the success of the set operation. Refer to "Return Codes and Reason Codes" on page 61 for more information on these return codes.

setrsn
: (output, INT, 4) is a reason code variable describing the success of the set operation. Return codes that are applicable to the FSMRMSVC CSL routine are listed at the end of this CSL-routine description.

reqtoken
: (input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. The output request token is used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the

call to this routine is interpreted as a check-back call on an asynchronous request, and the contents of this variable is the request identifier.

vlabel
(input, CHAR, 6) indicates that the external volume with this specified label have its category changed.

targetcat
(input, CHAR, 1–32) specifies the name of a target category to which the volume is set. The following special category names can be specified:

- EJECT
- EJECTB
- SCRATCH*x*
- VOLspecific
- *hexvalue*
- copy_export

Note:

1. Copy_Export is not supported as a source category.
2. A source category must not be specified when the target category is copy_export.

length1
(input, INT, 4) specifies the length of the preceding character parameter, target category.

libname
(input/output, CHAR, 1–32) specifies the library name in which the operation is requested. This parameter is not required when a real device (*rdev*) is provided, nor is it required when there is only one 3495 attached and the real address is not specified. If this field is not provided as input, it is returned as output.

length2
(input, INT, 4) specifies the length of the preceding character parameter, library name. When library name is returned as an output field, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated.

rdev
(input, CHAR, 4) specifies the real device address used for communicating with the library to process the request. This parameter is an optional input parameter. Any available device is used if this parameter is omitted. This parameter is ignored when enhanced library control is available. If this parameter is specified, the address must be valid.

sourcecat
(input, CHAR, 1–32) specifies the name of a current source category of the volume. The following special category names can be specified:

- INSERT
- SCRATCH*x*
- VOLspecific
- *hexvalue*

Note:

1. INSERT as a source category is not supported by a PtP VTS.

length3
(input, INT, 4) specifies the length of the preceding character parameter, source category name. When this parameter is specified, the set operation can succeed only if the volume is assigned to the specified source category when the request is made.

ASSIGN
: (input, CHAR, 6) indicates that the device attach operations that take place as part of processing include device assignment to the system. This parameter is ignored when enhanced library control is available.

NOASSIGN
: (input, CHAR, 8) indicates that the device attach operations that take place as part of processing not affect the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.) This parameter is ignored when enhanced library control is available.

length4
: (input, INT, 4) specifies the length of the preceding character parameter, ASSIGN or NOASSIGN.

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, "Return Codes and Reason Codes for CSL Routines," on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

FSMRMSVB – Set Volume Categories Using Bulk Processing

The FSMRMSVB CSL routine provides a programming interface equivalent to the DFSMSRM SET VOLCAT BULK command. With this routine, you can request that a list of various volume-category assignments be made. Such volume-category assignments include:

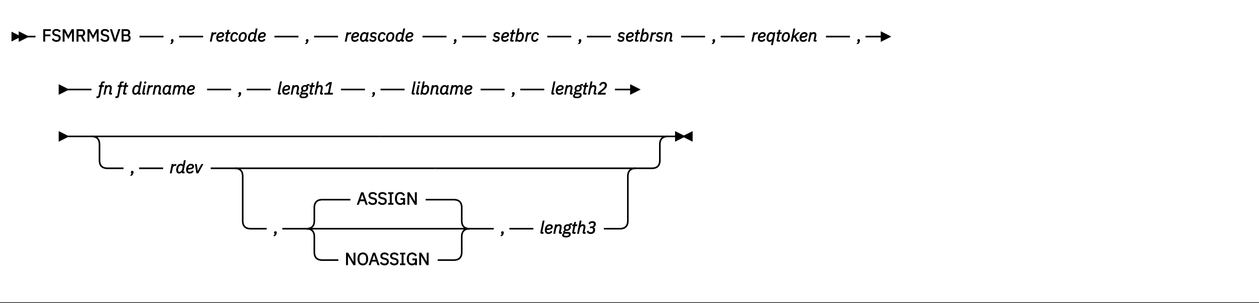
- Reassigning a volume from one category to another.
- Removing a volume from a library enclosure by specifying the EJECT category as the target category.
- Assigning a new library volume now in the INSERT category to a specified category.

Bulk category assignment are especially useful for initial migration of the volume inventory to the 3494 or 3495. (A list of category assignments must be in a preedited file and subscribe to the required format as described in Chapter 4, "Creating DFSMS/VM RMS Bulk Processing Files," on page 23.)

Call Format

The format for calling a CSL routine is language dependent. FSMRMSVB is called only through DMSCSL.

Refer to Appendix C, "Invoking CSL Routines," on page 105 for examples.



Parameters

FSMRMSVB
: (input, CHAR, 8) is the name of the CSL routine being invoked. The value FSMRMSVB can be passed either directly or in a variable.

retcode
: (output, INT, 4) is a variable for the return code from FSMRMSVB. Refer to "Return Codes and Reason Codes" on page 61 for information on these return codes.

reascode
: (output, INT, 4) is a variable for the reason code from FSMRMSVB. Refer to "Return Codes and Reason Codes" on page 61 for a list of reason codes.

setbrc
: (output, INT, 4) is a return code variable describing the success of the set-bulk request. Refer to "Return Codes and Reason Codes" on page 61 for more information on these return codes.

setbrsn
: (output, INT, 4) is a reason code variable describing the success of the set-bulk request. Return codes that are applicable to the FSMSVB CSL routine are listed at the end of this CSL-routine description.

reqtoken
: (input/output, INT, 4) identifies a specific asynchronous request. If it contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is an asynchronous call, and a request token is provided as output. The output request token is used on subsequent calls when checking back on the asynchronous response. If neither binary 0 nor binary 1 is specified as input, the call to this routine is interpreted as a check-back call on an asynchronous request, and the contents of this variable is the request identifier.

fn ft dirname
(input, CHAR, 5–150) specifies the file name, file type, and directory name of the bulk library processing file. The three elements of this fully qualified file name must be separated by at least a blank character.

length1
(input, INT, 4) specifies the length of the preceding character parameter, file name, file type, and directory name.

libname
(input/output, CHAR, 1–32) specifies the library name in which the operation is requested. This parameter is not required when a real device (*rdev*) is provided, nor is it required when there is only one 3495 attached and the real address is not specified. If this field is not provided as input, it is returned as output.

length2
(input, INT, 4) specifies the length of the preceding character parameter, library name. When library name is returned as an output field, it is important that the length be adequate; specifying the maximum length (32 bytes) prevents returned information from being truncated.

rdev
(input, CHAR, 4) specifies the real device address used for communicating with the library to process the request. This parameter is an optional input parameter. If this parameter is omitted, any available device is used. This parameter is ignored when enhanced library control is available. If this parameter is specified, the address must be valid.

ASSIGN
(input, CHAR, 6) indicates that the device attach operations that take place as part of processing include device assignment to the system. This parameter is ignored when enhanced library control is available.

NOASSIGN
(input, CHAR, 8) indicates that the device attach operations that take place as part of processing must not affect the current state of device assignment to the system. (This option is for unusual circumstances and should be used judiciously.) This parameter is ignored when enhanced library control is available.

length3
(input, INT, 4) specifies the length of the preceding character parameter, ASSIGN or NOASSIGN.

Reason Codes

Reason codes (*XXXXXrsn*), return codes (*XXXXXrc*), and explanations of these codes for all CSL routines are listed in Appendix A, “Return Codes and Reason Codes for CSL Routines,” on page 85. Additionally, reason codes (*reascode*) and return codes (*retcode*), which report the overall status of CSL processing, are listed (with their explanations) in Table 10 on page 63.

Appendix A. Return Codes and Reason Codes for CSL Routines

Reason codes (*XXXXXrsn*) and return codes (*XXXXXrc*) that are associated with the RMS operation processing for individual CSL routines are listed in Table 12 on page 85. Also included in the table are explanations of the codes and CSL routines they are applicable to.

Note: For explanations of return codes (*retcode*) and reason codes 3900–3999 (*reascode*), which report errors detected by the CSL itself, refer to Table 10 on page 63.

Table 12. Return Codes and Reason Codes for CSL Routines

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	0000	ALL	Internal processing error occurred.
8	0012	ALL	Requestor not authorized to issue this command.
8	0020	ALL	Invalid Opcode—Request type unknown.
8	0024	ALL	Master not accepting commands.
4	0036	FSMRMMNT-C	Request not found for MOUNT CANcel.
4	0048	ALL	The check-back call cannot be satisfied because the request could not be found.
8	0064	FSMRMMNT-C	Invalid request for mount CANcel.
8	0076	ALL	Request could not be started.
8	2030	FMSRMMNT-C	Requestor does not have the authority to cancel.
8	2040	FSMRMMNT-C	Request was already cancelled.
8	3000	ALL	Library/Device Mismatch—The specified device does not reside in the specified library.
8	3004	FSMRMMNT, FSMRMQLB, FSMRMRDC, FSMRMSDC, FSMRMSVC, FSMRMSVB	Source Category Undefined—The source category specified in a MOUNT, SET VOLCAT, or SET DEVCAT command is undefined.
8	3008	ALL	Undefined Device—The specified device is not part of the RMS library configuration. It was not genned in the RMCONFIG DATA file.
8	3012	FSMRMQLB, FSMRMRDC, FSMRMSDC, FSMRMSVC, FSMRMSVB	Invalid Manual Request—A request other than a mount or demount request is received and the specified library is a manual library.

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	3016	FSMRMMNT	NOIDRC option invalid for Automated library—The user has specified the NOIDRC option on a request that is going to an automated library. Since all automated library devices are IDRC-capable, this option is not valid. IDRC/NOIDRC can only be specified if the request is bound for a manual library.
8	3020	ALL	Invalid Library Name—The request specifies a library that is not defined in the control file, DGTMCNTL DATA.
8	3024	FSMRMMNT, FSMRMSVC	Target category not defined for input library, or for library in which specified device resides.
8	3028	FSMRMMNT	Target or source category defined for mount issued to manual library.
8	3032	FSMRMRDC	Specified category not assigned to specified device.
8	3034	FSMRMSDC, FSMRMSVB	Volume not currently assigned to the source category.
8	3100	ALL	FSMRMSHR Error—The return code from the installation exit FSMRMSHR indicates processing should not be allowed to continue.
8	3104	ALL	FSMRMDTE Error—The return code from the installation exit FSMRMDTE indicates the device could not be detached.
8	3108	ALL	FSMRMATE Error—The return code from the installation exit FSMRMATE indicates the device could not be attached.
8	3112	ALL	FSMRMDEV Error—The return code from the installation exit FSMRMDEV indicates that a suitable device could not be found.
8	3116	ALL	Preprocessing Error—The return code from the installation exit FSMRMPRE indicates that processing should not be allowed to continue.
8	3120	FSMRMDMT, FSMRMMNT	Library is in paused mode and the control file parameter RM_REQUEST_QUEUEING indicates that requests should be rejected when the library is in paused mode.
8	3124	FSMRMMNT, FSMRMSDC	Attach-to-User not logged on—The user that the device is supposed to be attached to when the command completes is not logged on.
8	3128	ALL	Device Cannot Be Opened—An internal processing error has occurred attempting to use the specified or selected device.
8	3132	ALL	Execute CCW Failed—Internal error attempting to issue I/O.
8	3136	ALL	No Device Available—No device is available to process this request.

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	3140	ALL	Specified Device Not Available—The device requested is not available to attach, or has become unavailable during processing.
8	3144	ALL	Attach or Detach Error—An error has occurred while attempting to attach or detach the device.
8	3148	ALL	Fail forwarding—Manual (check if this is valid)
8	3152	ALL	Unable to acquire I/O resource
8	3156	FSMRMSVC FSMRMSVB FSMRMQRL	Diagnose 254 I/O failed during enhanced library control
8	3160	FSMRMMNT	Failed Logical Volume Mount—A MOUNT request failed with the library hardware indicating that a logical volume mount has failed.
4	3200	ALL	Postprocessing Error—The return code from the installation-wide exit FSMRMPRO indicates that an error occurred.
4	3204	FSMRMMNT	Access Controls Not Set—The MOUNT command specified or defaulted to READOnly, but the attempt to set Logical Write Protect after the mount completed, fails.
4	3208	ALL	Attach or Detach Error After Completion—The primary function of the command has completed, but a subsequent attempt to attach or detach the device is unsuccessful.
4	3212	ALL	FSMRMDTE Error After Completion—The return code from the installation-wide exit FSMRMDTE indicates the device cannot be detached from the RMS master. This has occurred after the primary function of the command is completed.
4	3216	ALL	FSMRMATE Error After Completion—The return code from the installation-wide exit FSMRMATE indicates that the device cannot be attached to the requestor or the attach-to-user. This has occurred after the primary function of the command is completed.
8	3220	FSMRMDMT	Category Not Changed—This reason code applies to the Library Change Category command which is not supported in this release.
8	3300	FSMRMDMT, FSMRMMNT, FSMRMSVC, FSMRMSDC, FSMRMRDC	Completion Status Unknown
4	3304	FSMRMDMT, FSMRMMNT, FSMRMSVC, FSMRMSVB	External Label Unreadable—The mount, demount, or set volume category request completes, but the volume's external label cannot be verified because it is unreadable or missing.

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	Reason Code (*xxxxxrsn*)	Interfaces	Explanation
8	3304	FSMRMQLB-A	External Label Unreadable—Applies to FSMRMQLB AUDit function only. The volume's external label cannot be verified because it is unreadable or missing.
8	3308	FSMRMMNT, FSMRMQLB-A, FSMRMSVC, FSMRMSVB	Cancelled by Request—The operation is cancelled because a DISCard command has been received specifying its transaction ID.
8	3312	FSMRMDMT, FSMRMMNT, FSMRMSVC, FSMRMSVB	Cancelled Due to Order Sequence—A pending operation is cancelled by another operation; for example, a pending mount operation has been cancelled because a demount is issued to the device.
8	3316	FSMRMQLB-A	Cancelled Due to Manual Mode—The Query LIBrary VOLume AUDIT command is cancelled because the library has entered manual mode.
8	3320	FSMRMDMT, FSMRMMNT, FSMRMSVC, FSMRMSVB	Hardware Malfunction—The request fails because of a hardware failure condition at the time the request is attempted.
8	3324	FSMRMDMT, FSMRMMNT, FSMRMSVC, FSMRMSVB	The requested volume is in a position that is inaccessible to the library.
8	3328	FSMRMMNT	Source Category Empty—The mount request failed because the specified source category is empty.
8	3332	FSMRMDMT, FSMRMMNT, FSMRMQLB-A, FSMRMSVC, FSMRMSVB	Volume Not In Inventory—The requested volume has been deleted from the inventory.
8	3336	FSMRMMNT, FSMRMSVC, FSMRMSVB	Volume In Use—The requested volume is already mounted or mount pending.
8	3340	FSMRMDMT, FSMRMMNT, FSMRMQLB, FSMRMSVC, FSMRMSVB	Volume Not in Library—The requested volume is not known to the library manager.
8	3344	FSMRMMNT	Library catalog empty
8	3348	FSMRMDMT, FSMRMMNT, FSMRMQLB-A, FSMRMSVC, FSMRMSVB	Library Volume Misplaced—The specified volume is misplaced.
8	3352	ALL	Volume Exported—The requested volume has been exported.

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	3356	FSMRMDMT, FSMRMMNT, FSMRMQLB-A, FSMRMSVC, FSMRMSVB	Volume Manually Ejected—The requested volume has been manually ejected from the library.
8	3360	FSMRMSDC	Category in Use—The requested category for a SET DEVCAT command is already in use.
8	3364	FSMRMMNT	Mount In Progress—A mount is already in progress on the requested device.
8	3368	FSMRMMNT	Mount Already Pending—A mount is already pending on the requested device.
8	3372	FSMRMDMT	Demount Already Pending—A demount is already pending on the requested device.
8	3376	FSMRMDMT	No Volume Mounted—No volume is mounted on the device for which this demount is requested.
0	3380	FSMRMQLB	Request Still Active
4	3384	FSMRMDMT	Request Completed During Restart
4	3388	FSMRMMNT	Request cannot be completed due to restart.
8	3392	RSMRMMNT	Unrecoverable load failure
8	3396	FSMRMMNT	Damaged cartridge ejected
8	3500	ALL	Library Attachment Check
8	3504	ALL	Library Manager Off-line
8	3508	ALL	Control Unit and Library Manager Error
4	3512	FSMRMDMT, FSMRMMNT, FSMRMSVC	Library Vision Failure—The robot's vision system has failed. This error applies to FSMRMSVC only when the target category is EJECT or EJECTB and the library is in manual mode.
8	3512	FSMRMQLB-A, FSMRMSVC	Library Vision Failure—The robot's vision system has failed. This error applies to FSMRMQLB only for the AUDit function. This error applies to FSMRMSVC only when the target category is EJECT or EJECTB and the library is in manual mode.
8	3516	ALL	Library Not Capable
8	3520	FSMRMMNT	Demount Signaled—On a MOUNT command, the MOUNT completed, but it has subsequently been DEMOUNTed before command completion was returned to RMSMASTR.
8	3524	FSMRMMNT	Cancelled Library Operator—The 3494 operator cancelled the library mount request.
8	3528	FSMRMSVC	A copy export request completed, but with an indication that an exception occurred during processing in the tape library.

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	3532	FSMRMSVC	A copy export request was rejected by the tape library because no volume was found to export.
8	3536	FSMRMSVC	A copy export request was rejected by the tape library because the export list volume could not be processed.
8	3540	FSMRMSVC	RMS rejected a copy export request because there already is a request being processed by the tape library. Only one request per library can be active.
8	3600	ALL	Manual Rewind/Unload
8	3604	ALL	Degraded Mode
8	3608	ALL	Device Not Online
8	3612	ALL	Bus-Out Parity Check
8	3616	ALL	Channel Interface Permanent Error
8	3620	ALL	Channel Protocol Error
8	3624	ALL	Unrecognized I/O Error
8	3628	ALL	Data Streaming Error—This error is received when actually writing to the tape, which the RMS master does not do.
8	3632	ALL	Path Equipment Check
8	3636	ALL	Read Data Check
8	3640	ALL	Load Display Check
8	3644	ALL	Write Data Check
8	3648	ALL	Read Opposite
8	3652	ALL	Write ID Mark Check
8	3656	ALL	Unsolicited Environment Data
8	3660	ALL	Environmental Data Present
8	3664	ALL	Permanent Equipment Check
8	3668	ALL	Data Security Erase Failure
8	3672	ALL	RMS Library Not Capable
8	3676	ALL	Write Protected
8	3680	ALL	Tape Void
8	3684	ALL	Tension Loss
8	3688	ALL	Load Failure
8	3692	ALL	Unload Failure
8	3696	ALL	Drive Equipment Check
8	3700	ALL	End of Data
8	3704	ALL	Tape Length Error

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	3708	ALL	Backward at Beginning of Tape
8	3712	ALL	Drive Switched Not Ready
8	3716	ALL	Overrun
8	3720	ALL	Record Sequence Error
8	3724	ALL	Drive Not Ready
8	3732	ALL	Locate Block Unsuccessful
8	3736	ALL	Drive Assigned Elsewhere
8	3740	ALL	Unsolicited Sense
8	3744	ALL	Control Unit ERP Failed
8	3748	ALL	CU and Drive Incompatible
8	3752	ALL	Max Block Size Exceeded
8	3756	ALL	Read Buffered Log Overflow
8	3760	ALL	Read Buffered Log EOV
8	3764	ALL	Tape Length Incompatible—Select a different tape drive with a compatible device type, or contact the Library Administrator.
8	3768	ALL	Format 3480XF Incompatible
8	3772	ALL	Format 34802 XF Incompatible
8	3776	ALL	Tape Length Violation
8	3780	ALL	Physical End of Volume
8	3784	ALL	Recovered Check One Failure
8	3788	ALL	Global Command Intercept
8	3792	ALL	Compaction Algorithm Error
8	3796	ALL	Volume Fenced
8	3800	ALL	Command Reject—I/O that the RMS master has issued to the library has been rejected because of errors in the channel program.
8	3804	ALL	Function Incompatible
8	3808	ALL	Library Volume Reserved
4	3812	FSMRMDMT, FSMRMMNT	Library Vision Failure—The mount or demount request completes, but the volume's external label cannot be verified because the library vision system is not operational.
8	3820	FSMRMDMT	Demount Volser Mismatch
8	3824	ALL	Lost sense
8	3826	ALL	Allegiance reset

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	Reason Code (*xxxxxrsn*)	Interfaces	Explanation
8	3828	ALL	Configuration error
8	3830	ALL	Protection exception
8	3832	ALL	Write length error
8	3834	ALL	Read-only format
8	3836	ALL	Beginning of partition
8	3838	ALL	End of partition
8	3840	ALL	Device intervention
8	3842	ALL	Loader intervention
8	3844	ALL	Library intervention
8	3846	ALL	Write error
8	3848	ALL	Erase error
8	3850	ALL	Formatting error
8	3852	ALL	Read error
8	3854	ALL	Unsupported format
8	3856	ALL	No formatting
8	3858	ALL	Positioning lost
8	3860	ALL	Read length error
8	3862	ALL	Unsupported medium
8	3864	ALL	Medium removed
8	3866	ALL	Halt signal
8	3868	ALL	Device fenced
8	3870	ALL	Device path fenced
8	3872	ALL	Volume in input
8	3874	ALL	Volume ejected
8	3876	ALL	Duplicate volume
8	3878	ALL	Library output station full
8	3880	ALL	Library manager equipment check
8	3882	ALL	Library equipment check
8	3884	ALL	All library cells full
8	3886	ALL	No cleaner volumes in library
8	3888	ALL	I/O station door open
8	3890	ALL	Subsystem environmental alert
8	3892	ALL	Unrecognized message code
8	3894	ALL	File Not Found—The file specified could not be found.

Table 12. Return Codes and Reason Codes for CSL Routines (continued)

Return Code (*xxxxxrc*)	**Reason Code (*xxxxxrsn*)**	**Interfaces**	**Explanation**
8	3896	ALL	Partition Not Found—The partition specified could not be found.
8	3898	ALL	Physical Index Not Found—The physical index could not be found.
8, 12	39xx	For explanations of return codes (*retcode*) and reason codes 3900–3999 (*reascode*), which report errors detected by the CSL itself, refer to Table 10 on page 63.	
8	4000	FSMRMSVB	Bulk file does not exist
8	4001	FSMRMSVB	Bulk file invalid record length
8	4002	FSMRMSVB	Bulk file invalid records found
8	4003	FSMRMSVB	Bulk processing internal error
8	4004	FSMRMSVB	Bulk file has no records

Note: Interfaces applicable to each return-code/reason-code pair are listed in the column labeled “Interfaces.” FSMRMMNT-C refers to only those mount requests involving the CANcel option. FSMRMQLB-A refers to only those query-library requests involving the AUDit option.

PI end

Appendix B. 3494 and 3495 Control for Foreign Processors

This appendix describes how DFSMS/VM Removable Media Services (RMS) provides automated access to 3494 or 3495 cartridge data for "foreign" processors, when the 3494 or 3495 devices are connected to a z/VM system as well as the foreign host. A foreign host in this context, is a TCP/IP-capable processor or host environment that does not provide native control for the 3494 or 3495 but does support the IBM 3490/3490E tape devices to which it is connected.

This appendix provides:

- An overview of foreign-host support
- A list of software requirements for foreign-host RMS support
- A processing overview for the foreign host and the foreign-host server
- Information on user exits for customizing foreign-host processing
- Information on defining a server machine for foreign hosts
- A list of request interface formats and response formats for foreign hosts

Foreign-Host Support Overview

DFSMS/VM provides a C module to manage TCP/IP socket connections, a primary REXX EXEC to handle request processing, and several REXX user exits. You can run these programs on a dedicated service virtual machine that you define as your foreign-host server. This service machine uses the RMS programming interface to request 3494 or 3495 functions for the foreign-host client. Support includes:

- Mounting a specified volume or a volume from a 3494 or 3495 scratch pool
- Demounting a volume
- Cancelling an in-process mount request
- Querying the status of an in-process or completed mount request

The relationships between the service machines and processors are described in Figure 9 on page 96.

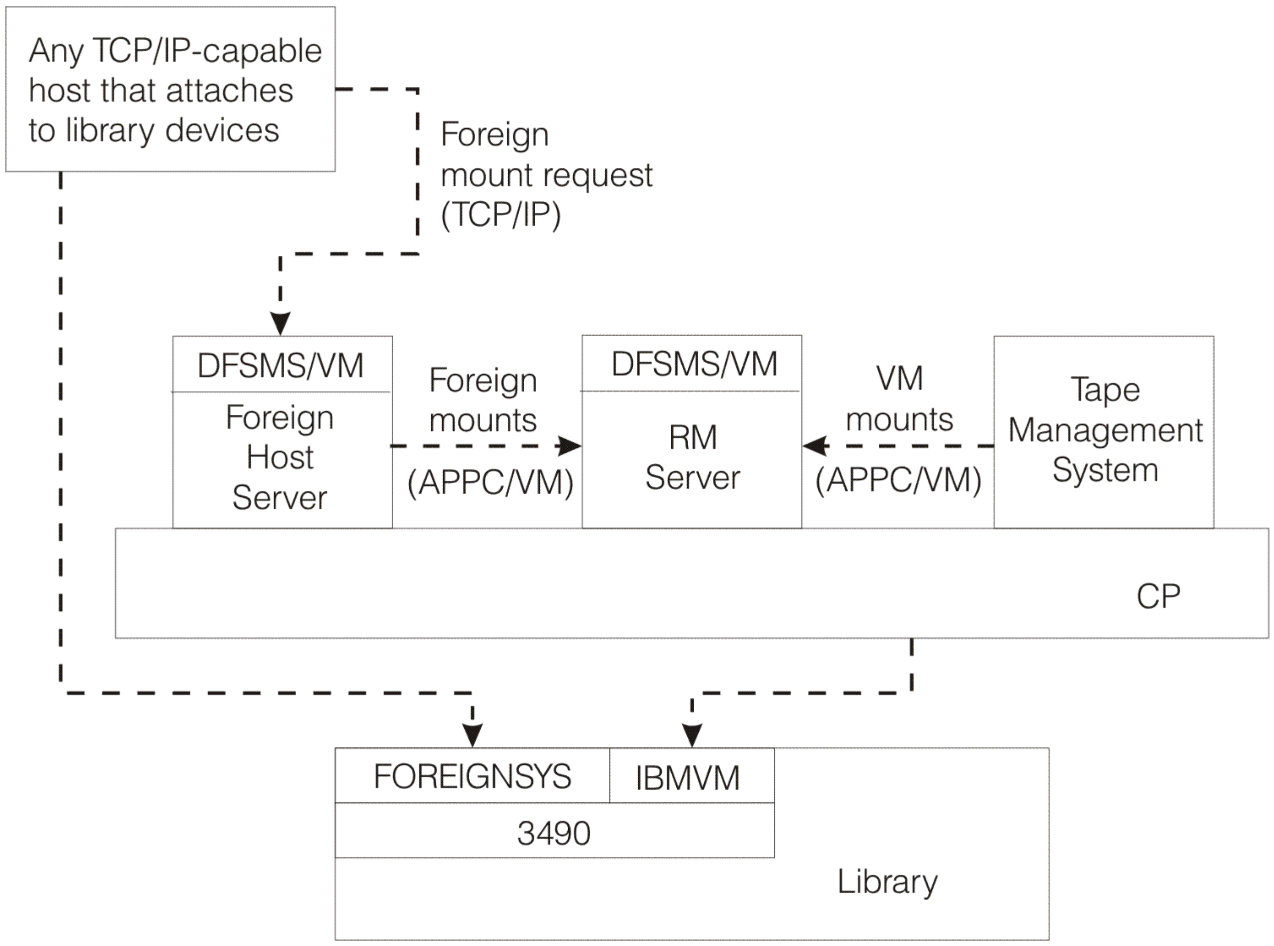


Figure 9. Foreign-Host Server Machine Interaction

To avoid potential bottlenecks, the recommended configuration is one foreign-host server for each foreign processor.

Software Requirements

RMS provides automated access for “foreign” processors only if the following software is installed:

DFSMS/VM: DFSMS/VM function level 221 is required for foreign-host support. The foreign-host server cannot function unless the RMS master is operational; however, other DFSMS/VM service machines need not be operational.

TCP/IP for VM: TCP/IP function level 620 for VM or higher at 3494 or 3495 hardware availability is required for foreign-host support. Using TCP socket support, the foreign-host server plays a passive-server role.

SAA AD/Cycle Language Environment: The foreign-host support software that handles TCP/IP socket support requires the C run-time library.

Tape Management Software: DFSMS/VM support for the 3494 or 3495 provides a high-level interface to automated library function and is designed to work in concert with the customer’s tape management system. When foreign-host support is used, the foreign host is expected to provide tape management functions such as user authorization, device selection, external-to-internal cartridge label verification, and management of the enterprise tape inventory/catalog (Although the REXX user exits provided for client authorization and for device address translation could be used to augment certain tape management functions, this is not their intended purpose.)

Processing Overview

Critical tasks performed by the foreign host (client) and the foreign-host server are summarized in this part of the appendix.

Foreign-Host Processing Overview

The foreign host is responsible for:

- Selecting the device

  The foreign host must designate the device used in the 3494 or 3495 operation.
- Supplying the required request data

  Using TCP/IP sockets, the foreign system sends a request message that complies with a defined interface format. The information in the request message is used by the foreign-host server to build RMS requests for 3494 or 3495 functions. The supported message formats are described in Table 14 on page 100 and Table 15 on page 101.

Foreign-Host Server Processing Overview

The foreign-host server handles:

- Authenticating the client

  Foreign-host server processing invokes a user exit, FSMRMFHA EXEC, for screening clients, to enable rejection of requests originated by unauthorized communicators.
- Validating information supplied in the request

  The foreign-host server validates the information in the request and rejects the request if invalid input is detected. The validation includes translation of the device designation to a real address known to the VM system, in the event that foreign-host and VM-system address specifications differ. Translation is handled in a user exit, FSMRMFHD EXEC.
- Issuing the DFSMS/VM RMS request

  The foreign host processing exploits the DFSMS/VM RMS CSL interface to issue RMS requests. Because the attach operation is requested without device assignment, the RMS master uses special device assignment handling.
- Notifying the client of request status

  The foreign host sends response messages to the client, indicating initial status and optionally, final outcome of the request. A unique identifier, or token, is returned for use by the client in making a query about the request, if it is preferable that foreign-host processing not wait for final responses. The token can also be used to cancel a mount request.

Usage Notes

Because the device is not assigned to the VM system during the handling of foreign-host requests, the RMS master cannot perform some tasks that it ordinarily performs when working with devices assigned to the VM system. For example:

- The foreign host must perform rewind/unload operations. Requests directed to a device on which a mounted volume is not unloaded will fail. The RMS master cannot perform this operation.
- The foreign host must set logical write protection as required. The RMS master cannot set logical write protection when the NOASSIGN option is requested. (NOASSIGN is always used by the foreign-host server.)
- The RMS master does not read buffered log data while handling requests on devices associated with foreign hosts, and so does not log the corresponding MDR records.

User Exits

Three user exits are provided to customize foreign-host processing for meeting specific installation needs.

Customizing the Environment

The user exit, FSMRMFHC EXEC, provides a means for customizing foreign-host processing and specifying installation variables. The modifiable characteristics include highly critical resource identification, performance-tuning variables, and processing options.

The control fields to be provided are described in Table 13 on page 98 along with the default value for each.

Note: It is essential to maintain the stated sequence for these variables.

Table 13. Control Fields for Customizing a Foreign-Host Environment

Control field	**Description**	**Default value**
PORT	Specifies which port address is to be assigned to the server.	Because the shipped value of “1234” may already be used, it is essential that you contact your TCP/IP administrator to obtain an available port address.
BACKLOG	Defines the maximum length of the queue for pending connections. This value is externalized as a tuning factor for controlling server through-put.	The default value of 10 is used unless changed in this exit.
TELL_USERID	Designates which VM user ID will receive messages related to questionable server events, if such notification is requested with a further option. A *tell_userid* on another system may be designated by using the conventions *userid/nodeid*, with no intervening blanks. A nickname can be used to specify a list of user IDs if you create a NAMES file on the foreign-host server machine’s A-disk.	There is no default user ID specified initially. Until you modify this option to reflect a user ID at your installation, or if the *tell_userid* field is set to blanks, the server sends messages to itself if the TELL control field indicators are set to “Y” (Yes).
TELL_ON_ERROR	Indicates whether the notify user ID receives a message when a non-fatal error occurs.	A default value of “Y” (Yes) is used unless changed in this exit.
TELL_ON_FATAL	Indicates whether the notify user ID receives a message when a fatal error occurs.	A default value of “Y” (Yes) is used unless changed in this exit.

Table 13. Control Fields for Customizing a Foreign-Host Environment (continued)

Control field	Description	Default value
RESULTS_AGE_LIMIT	Specifies the age criteria for retaining status information on completed mount and demount requests. The server maintains a file of results for mount and demount functions for which a final response is not requested. Record of a specific request is removed when a query is processed for that request. This parameter enables the server to prune requests information that has been on file for more than a specified number of days.	The default value of "1" (one day) is used unless changed in this exit.

Authorizing Clients

As shipped, the user exit FSMRMFHA EXEC authorizes connections by returning an authorization code of 0, which indicates that the server maintains the connection and processes the request. Any nonzero value will cause the server to close the attempted connection, after informing the communicator of authorization failure. Unauthorized attempts to connect are reported and logged on the console. The customer can screen the client, based on the information that is passed to this exit—internet address (domain, port, and ip address) and request header fields ("magic number" and user identification)—and then set the authorization return code.

Translating Device Addresses

The user exit FHSRMFHD EXEC provides a facility for translating a device address used by the client to the real device address that is known to the VM system. As shipped, this exit assumes that the device address designations used by the foreign host are the same as those by which the VM system knows devices; thus, default processing returns the same device designation passed to the exit. The customer must provide REXX code to interpret foreign-host addresses in terms of VM addresses if default processing is not used.

Server Machine Definition

The recommended configuration for foreign-host support is to define and customize a unique service machine for each foreign host. The foreign-host server user IDs are selected by the installation. If desired, the server names can be specified in the RM_FOREIGN_SERVER_VM keywords of the DFSMS/VM control file so that the RMS master can autolog the foreign-host servers.

Foreign-host support is initiated by invoking FSMRMFH1. If a foreign-host server is to be autologged, its PROFILE EXEC should call this module.

Minidisk Requirements

A foreign-host server requires a 191 disk on which it maintains files of requests that it is processing. A minidisk size equivalent to at least five 3380 cylinders is recommended. Additional space may be required if all of the following are true:

- Foreign-host processing does not request that final responses be returned
- Request querying is part of the standard foreign-host request protocol
- The RESULTS_AGE_LIMIT parameter in the customization exit is more than one day

Product Access

The foreign-host server machines must have access to the DFSMS/VM product disk and to the TCPMAINT 592 disk. Also, ensure that the start-up procedure for the foreign-host server issues appropriate link or global commands for accessing your C run-time environment.

Request Interface Formats

The foreign-host support expects request messages to conform to a specified format for the function requested, and likewise, issues response messages in prescribed formats. The formats are specified in Table 14 on page 100 and Table 15 on page 101.

Table 14. Format of Requests for 3494 or 3495 Foreign-Host Functions. Format of the message sent by the foreign host to the DFSMS/VM foreign host server.

Byte Pos (Size)	**Data Element**	**Description—all request types**
Request Header for All Request Types		
01–04 (4)	"Magic number"	A fixed, customer-designated value identifying the message to follow as a 3494 or 3495 request. This value is passed to a FHS Authorization Exit, along with client identification data, for use in request authentication.
05–18 (14)	Identification data	Space for "loginid, username, acctname" or other information desired by the customer for authenticating clients.
19–20 (2)	Request type	The type of function requested: • "MV"—mount a specified volume • "MC"—mount the next volume in a specified category • "DM"—demount a volume from a specified device • "CM"—cancel a previously requested mount • "QR"—query a previously requested mount or demount
Request Data for Mount (MV and MC) and Demount (DM)		
01–04 (4)	Tape drive	Device address designation of tape drive to use. This field is required for all mount and demount requests.
05–10 (6)	Volume label	Exterior label of cartridge, expressed as up to six characters, left justified and right-padded with character blanks. This field is required for type MV; it is ignored on input but returned as output for type MC; and it is optional for type DM and returned as output if blank as input.
11–11 (1)	Response flag	An indicator to identify if a final request message is to be returned by the foreign-host server upon completion of the 3494 or 3495 operation: • "Y"—a final response is requested • "N"—a final response is not requested When a final response is not requested, the server closes the socket after the initial response is returned to the client, whereas the socket remains open until the request completes and the final response is sent when the final message is requested. (When final message is not desired, the Query (QR) request provides an alternate means for determining request results if it is preferable for the client not to check the socket open while awaiting a final response message.)

Table 14. Format of Requests for 3494 or 3495 Foreign-Host Functions. Format of the message sent by the foreign host to the DFSMS/VM foreign host server. *(continued)*

Byte Pos (Size)	**Data Element**	**Description—all request types**
12–43 (32)	Source category	Name of the scratch-pool category from which the next volume is to be mounted for request type MC. Category name must be left-justified and right-padded with character blanks. Imbedded blanks are not allowed; the first blank encountered marks the end of the category name. This field must be nonblank for type MC; it is ignore for types MV and DM.
44–75 (32)	Target category	Name of the category to which a volume is assigned as part of request processing. Valid values are scratch category names and “VOLspecific”, per normal VM request conventions. Syntax rules for source category above apply here. Use of this field is optional for all request types. It is ignored if left blank.
Request Data for Cancel Mount (CM) and Query Request (QR)		
01–04 (4)	Request ID	The target request ID to be cancelled or queried.

Table 15. Format of Responses for 3494 or 3495 Foreign-Host Functions. Format of the response message returned by the DFSMS/VM foreign host server.

Byte Pos (Size)	**Data Element**	**Mount (MV & MC) and Demount (DM)**	**Cancel Mount (CM)**	**Query Request (QR)**
Response Header				
01–04 (4)	Request Identifier	The unique nonnegative integer (binary) value assigned to identify this request. If the request is not accepted for processing, this field is returned as binary 0.	The target request identifier that is the object of the cancel operation.	The target request identifier that is the object of the query.
05–08 (4)	Reason Code (documented with RMS CSL routines)	A binary reason code describing the error associated with status code X'20' on initial (I) responses or with X'04.' or X'08' on final (F) responses.	A binary reason code describing a CSL error or RMS master processing error associated with status code X'20' or X'08', respectively.	A binary reason code describing the RMS master processing error associated with status code X'04' or X'08'.

Table 15. Format of Responses for 3494 or 3495 Foreign-Host Functions. Format of the response message returned by the DFSMS/VM foreign host server. *(continued)*

Byte Pos (Size)	**Data Element**	**Mount (MV & MC) and Demount (DM)**	**Cancel Mount (CM)**	**Query Request (QR)**
09–09 (1)	Status Code	When a final response was requested, and no errors are detected by foreign-host server: • X'00'—request is accepted, initial (I) response (final response to follow.) Whether a final response was requested or not and an error is detected in foreign-host server processing, status codes returned as final (F) response (no additional responses): • X'11'—client not authorized • X'12'—syntax error in the incoming request data • X'13'—device address not known to VM • X'15'—internal error (examine server console) • X'20'—Problem detected in RMS CSL routine processing Final (F) response status if request accepted and completed: • X'00'—request completed with no abnormal conditions • X'04'—request completed with abnormal condition • X'08'—request completed unsuccessfully	Final response only (synchronous): • X'00'—request completed with no abnormal conditions • X'04'—request cancelled by RMS master but not found for purge from foreign-host server work in-process queue • X'08'—request completed unsuccessfully • X'11'—client not authorized • X'12'—syntax error in the incoming request data • X'15'—internal error (examine server's console) • X'16'—no in-process requests known to foreign-host server • X'20'—problem detected in RMS CSL routine processing	Final response only (synchronous): • X'00'—request completed with no abnormal conditions • X'02'—request still in-process • X'04'—request completed with abnormal condition • X'08'—request completed unsuccessfully • X'11'—client not authorized • X'12'—syntax error in the incoming request data • X'15'—internal error (examine server's console) • X'16'—request not found, unable to report status
10–10 (1)	Response Type	Type of response: • I—initial • F—final	Always F (final)	Always F (final)

Table 15. Format of Responses for 3494 or 3495 Foreign-Host Functions. Format of the response message returned by the DFSMS/VM foreign host server. *(continued)*

Byte Pos (Size)	Data Element	Mount (MV & MC) and Demount (DM)	Cancel Mount (CM)	Query Request (QR)
11–12 (2)	Request Type	Type of request with which this response is associated: • MV—mount volume • MC—mount category • DM—demount	CM	QR
Response Data				
		Original request data returned. Volume information updated for MC and DM requests.	No response data	No response data

Appendix C. Invoking CSL Routines

PI

This appendix contains the following sample invocations for CSL routines:

- REXX
  - Sample program to demount a volume from a device (see Figure 10 on page 105)
  - Sample program to mount a volume onto a device (see Figure 11 on page 106)
  - Sample program to query a volume (see Figure 12 on page 107)
- C
  - Sample program to query the library operational state (see Figure 13 on page 108 and Figure 14 on page 109)
- Assembler
  - Sample program to mount a volume using the z/VM CSL fastpath method (see Figure 15 on page 110 through Figure 17 on page 112).

Refer to *z/VM: CMS Callable Services Reference* for detailed instructions on invoking CSL routines from your application programs.

```
/*******************************************************************/
/* Program to demount a volume from the device ED3                 */
/*******************************************************************/

/* Set program parameters                                          */
CSLrc     = 0                  /* CSL return code                  */
CSLrs     = 0                  /* CSL reason code                  */
FCTrc     = 0                  /* Function return code             */
FCTrs     = 0                  /* Function reason code             */
reqtoken  = 0                  /* 0=WAIT,1=NOWAIT,other=CHECK-BACK */
rdev      = '0ED3'             /* real device address              */
libname   = '                                ' /* library name     */
len1      = 32                                 /* library name len */
vlabel    = '      '           /* volume external label            */
targetcat = ''
targetcat = ''                 /* target category                  */
len2      = 0                  /* len of target category           */
options   = ''                 /* ASSIGN|NOASSIGN                  */
len3      = 0                  /* len of options                   */

/* Call CSL to perform the request                                 */
call CSL 'FSMRMDMT CSLrc CSLrs FCTrc FCTrs',
         'reqtoken rdev libname len1 vlabel',
         'targetcat len2 options len3'

/* Display return and reason codes                                 */
say 'CSL      return code = 'CSLrc
say 'CSL      reason code = 'CSLrs
say 'Function return code = 'FCTrc
say 'Function reason code = 'FCTrs
```

Figure 10. Sample REXX Program to Demount a Volume from the Device ED3

```
/*******************************************************************/
/* Program to mount volume 001454 on device ED3                    */
/*******************************************************************/

/* Set program parameters                                          */
CSLrc     = 0                 /* CSL return code                   */
CSLrs     = 0                 /* CSL reason code                   */
FCTrc     = 0                 /* Function return code              */
FCTrs     = 0                 /* Function reason code              */
reqtoken  = 0                 /* 0=WAIT,1=NOWAIT,other=CHECK-BACK */
mounttype = 'VOL'             /* volume mount=VOL,category..=CAT  */
vlabel    = '001454'          /* volume external label             */
rdev      = '0ED3'            /* real device address               */
libname   = '                                ' /* library name     */
len1      = 32                                  /* library name len */
userid    = '        '        /* "attach-to" userid                */
vdev      = '    '            /* virtual device address            */
sourcecat = ''                /* source category                   */
len2      = 0                 /* len of source category            */
targetcat = ''                /* target category                   */
len3      = 0                 /* len of target category            */
options   = ''                /* READONLY|READWRITE                */
                              /* IDRC|NOIDRC                       */
                              /* ASSIGN|NOASSIGN                   */
                              /* ATTACH|NOATTACH                   */
len4      = 0                 /* len of options                    */

/* Call CSL to perform the request                                 */
call CSL 'FSMRMMNT CSLrc CSLrs FCTrc FCTrs',
         'reqtoken mounttype vlabel rdev libname len1 userid vdev',
         'sourcecat len2 targetcat len3 options len4'

/* Display return and reason codes                                 */
say 'CSL      return code = 'CSLrc
say 'CSL      reason code = 'CSLrs
say 'Function return code = 'FCTrc
say 'Function reason code = 'FCTrs
```

Figure 11. Sample REXX Program to Mount Volume 001454 on Device ED3

```
/*****************************************************************/
/* Program to query volume 001454                                */
/*****************************************************************/

CSLrc     = 0                  /* CSL return code                */
CSLrs     = 0                  /* CSL reason code                */
FCTrc     = 0                  /* Function return code           */
FCTrs     = 0                  /* Function reason code           */
reqtoken  = 0                  /* 0=WAIT,1=NOWAIT,other=CHECK-BACK */
querytype = 'VOL'              /* OPS | VOL | DEV | COU | AUD    */
libname   = '                              ' /* library name     */
len1      = 32                              /* library name len */
vldvca    = '001454'           /* vlabel|vdev|category           */
len2      = 6                  /* len of the above parameter     */
status    = 0                  /* status of the queried object   */
volcateg  = ''                 /* volume category                */
len3      = 0                  /* len of the above parameter     */
volclass  = '    '             /* volume class                   */
voltype   = '    '             /* volume type                    */
volmnted  = '      '           /* volume mounted                 */
asscateg  = ''                 /* category assigned to queried dev */
len4      = 0                  /* len of the above parameter     */
count     = 0                  /* number of volume               */
rdev      = '    '             /* real device address            */
options   = ''                 /* ASSIGN | NOASSIGN              */
len5      = 0                  /* len of the above parameter     */

call CSL 'FSMRMQLB CSLrc CSLrs FCTrc FCTrs',
         'reqtoken querytype libname len1 vldvca len2 status volcateg',
         'len3 volclass voltype volmnted asscateg len4 count rdev',
         'options len5'

/* Display return and reason codes                               */
say 'CSL      return code = 'CSLrc
say 'CSL      reason code = 'CSLrs
say 'Function return code = 'FCTrc
say 'Function reason code = 'FCTrs
```

Figure 12. Sample REXX Program to Query Volume 001454

```
/***********************************************************/
/* Program to query the library operational status        */
/***********************************************************/
#include <stdio.h>
#include <string.h>

#pragma linkage(DMSCSL,OS)
extern int DMSCSL(const char *RTNNAME, int *RC,...);

main()
{
   /* Declare and initialize call parameters */
   int    retcode  = 0;        /* CSL return code
   */
   int    reascode = 0;        /* CSL reason code
   */
   int    queryrc  = 0;        /* Query return code
   */
   int    queryrsn = 0;        /* Query reason code
   */
   int    reqtoken = 0;        /* Synchronous request
   */
   char   querytype[3] = "OPS";/* Type of query
   */
   char   libname[32] = "                                "; /* 32
sp */
   int    length1 = 32;        /* Length of the previous parameter
  */
   char   queryobject = '\0';  /* N/A for query OPS
   */
   int    length2 = 0;         /* N/A for query OPS
   */
   unsigned short  status;     /* Status of the query
   */
   char   volcateg = '\0';     /* N/A for query OPS
   */
   int    length3  = 0;        /* N/A for query OPS
   */
   char   volclass = '\0';     /* N/A for query OPS
   */
   char   voltype  = '\0';     /* N/A for query OPS
   */
   char   volmnted = '\0';     /* N/A for query OPS
   */
   char   asscateg = '\0';     /* N/A for query OPS
   */
   int    length4  = 0;        /* N/A for query OPS
   */
   int    count    = 0;        /* N/A for query OPS
   */
   char   rdev[4] = "    ";    /* Have RM Master to select a device
 */
   char   options = '\0';      /* Use default options
   */
   int    length5 = 0;         /* Use default options
   */
   int    i;                   /* Loop count
   */
```

Figure 13. Sample C Program to Query the Library Operational State Part 1 of 2

```
   /* Call CSL Query function*/
   DMSCSL ("FSMRMQLB",&retcode,&reascode,&queryrc,amper.&queryrsn,&reqtoken,
\
            querytype,libname,length1,queryobject,length2,
     \
            &status,volcateg,length3,volclass,voltype,volmnted,
       \
            asscateg,length4,&count,rdev,options,length5);
   /* Report result of the call */
   if (retcode == 0 || retcode == 4) {     /* Success or warning cond.*/
      printf ("Library ");
      i = 0;                               /* Initialize loop count */
      while (libname[i] != ' ' && i < 32)  /* Print non-blank char. */
         printf ("%c",libname[i++]);
      printf (" is in ");
      switch (status) {                    /* Print status */
         case 0x0000 :
            printf ("auto mode.\n"); break;
         case 0x0100 :
            printf ("paused mode.\n"); break;
         case 0x0200 :
            printf ("manual mode.\n"); break;
         default:
            printf ("an unknown mode.\n"); break;
      }
   } else {                                 /* Error condition */
      if (retcode > 0 && reascode == 0) {   /* Error in RMS processing*/
         printf ("Return code is %d\n",queryrc);
         printf ("Reason code is %d\n",queryrsn);
      } else {                              /* Error in CSL processing*/
         printf ("Return code is %d\n",retcode);
         printf ("Reason code is %d\n",reascode);
      }
      printf ("See RMS User Guide and Reference for meaning ");
      printf ("of a non-0 reason code.\n");
      printf ("See CMS Application Development Reference for meaning ");
      printf ("of a negative return code.\n");
   }
}
```

Figure 14. Sample C Program to Query the Library Operational State Part 2 of 2

```
**********************************************************************
* This program calls CSL to mount "001454" using the Fast path.      *
**********************************************************************
MNT    CSECT
       BALR    R12,0
       USING   *,R12
*
* Save return address
*
       ST      R14,RETADR
*
* Initialize the fastpath area
*
      CSLFPI   TYPE=INIT,AREA=FP1,SERVICE=FSMRMMNT,                    *
               PARMS=((P3,FCTRC),(P4,FCTRS),(P5,REQTOKEN),             *
               (P6,MOUNTTYPE),(P7,VLABEL),(P8,RDEV),(P9,LIBNAME),      *
               (P10,LEN1),(P11,USERID),(P12,VDEV),(P13,SOURCECAT),     *
               (P14,LEN2),(P15,TARGETCAT),(P16,LEN3),(P17,OPTIONS),    *
               (P18,LEN4))
*
* Set values
*
      CSLFPI   TYPE=SET,AREA=FP1,                                      *
               PARMS=((P3,FCTRC),(P4,FCTRS),(P5,REQTOKEN),             *
               (P6,MOUNTTYPE),(P7,VLABEL),(P8,RDEV),(P9,LIBNAME),      *
               (P10,LEN1),(P11,USERID),(P12,VDEV),(P13,SOURCECAT),     *
               (P14,LEN2),(P15,TARGETCAT),(P16,LEN3),(P17,OPTIONS),    *
               (P18,LEN4))
*
* Do it
*
      CSLFPI   TYPE=CALL,AREA=FP1
*
* Show result
*
      BAL      R11,RESULT
*
* Return to control program
*
EXIT   DS      0H
       L       R14,RETADR        Get saved address
       BR      R14               Return
       EJECT                     End of main
```

Figure 15. Sample Assembler Program to Mount a Volume Using the MVS/ESA™ CSL fastpath method Part 1 of 3

```
**********************************************************************
* First level subroutine RESULT  (called by main)
*
* PURPOSE:  Display all parameters of the CSL
* ENTRY CONDITIONS: None
* EXIT CONDITIONS:  None
**********************************************************************
RESULT  DS  0H
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='REQTOKEN= &&1',              *
               SUB=(HEXA,(REQTOKEN,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='CSL RETCODE= &&1',           *
               SUB=(DECA,(CSLRC,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='CSL REACODE= &&1',           *
               SUB=(DECA,(CSLRS,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='MNT RETCODE= &&1',           *
               SUB=(DECA,(FCTRC,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='MNT REACODE= &&1',           *
               SUB=(DECA,(FCTRS,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='VLABEL   = &&1',             *
               SUB=(CHARA,(VLABEL,6))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='RDEV     = &&1',             *
               SUB=(CHARA,(RDEV,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='USERID   = &&1',             *
               SUB=(CHARA,(USERID,8))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='VDEV     = &&1',             *
               SUB=(CHARA,(VDEV,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='LIBNAME  = &&1',             *
               SUB=(CHARA,(LIBNAME,32))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='LEN1     = &&1',             *
               SUB=(DECA,(LEN1,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='SOURCECAT= &&1',             *
               SUB=(CHARA,(SOURCECAT,32))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='LEN2     = &&1',             *
               SUB=(DECA,(LEN2,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='TARGETCAT= &&1',             *
               SUB=(CHARA,(TARGETCAT,32))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='LEN3     = &&1',             *
               SUB=(DECA,(LEN3,4))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='OPTIONS  = &&1',             *
               SUB=(CHARA,(OPTIONS,32))
       APPLMSG APPLID=CMS,HEADER=NO,TEXT='LEN4     = &&1',             *
               SUB=(DECA,(LEN4,4))
       BR      R11
```

Figure 16. Sample Assembler Program to Mount a Volume Using the MVS/ESA™ CSL fastpath method Part 2 of 3

```
**********************************************************************  MNT00880
* Data section                                                          MNT00890
**********************************************************************  MNT00900
*                                                                       MNT00910
RETADR    DS      F                 Return address                      MNT00920
*                                                                       MNT00930
* Fastpath area                                                         MNT00940
*                                                                       MNT00950
FP1     CSLFPI TYPE=AREA,                                              *MNT00960
               PARMS=((RETR,CSLRC),(REAS,CSLRS),P3,P4,                 *MNT00970
               P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,P17,P18)      MNT00980
*                                                                       MNT00990
CSLRC      DS      F                CSL return code                     MNT01000
CSLRS      DS      F                CSL reason code                     MNT01010
FCTRC      DS      F                MOUNT return code                   MNT01020
FCTRS      DS      F                MOUNT reason code                   MNT01030
REQTOKEN   DC      F'0'                                                 MNT01040
MOUNTTYPE  DC      C'VOL'           VOL or CAT                          MNT01050
VLABEL     DC      CL6'001454'                                          MNT01060
RDEV       DC      CL4' '                                               MNT01070
LIBNAME    DC      CL32' '                                              MNT01080
LEN1       DC      F'32'                                                MNT01090
USERID     DC      CL8' '                                               MNT01100
VDEV       DC      CL4' '                                               MNT01110
SOURCECAT  DC      CL32' '                                              MNT01120
LEN2       DC      F'0'                                                 MNT01130
TARGETCAT  DC      CL32' '                                              MNT01140
LEN3       DC      F'0'                                                 MNT01150
OPTIONS    DC      CL32' '                                              MNT01160
LEN4       DC      F'0'                                                 MNT01170
           REGEQU                                                       MNT01180
           END     MNT                                                  MNT01190
```

Figure 17. Sample Assembler Program to Mount a Volume Using the MVS/ESA™ CSL fastpath method Part 3 of 3

PI end

Appendix D. Using VSE Guest Support

This appendix describes how to use a VSE Guest Server (VGS) to handle 3494 control requests for one or more VSE guests of z/VM.

This appendix includes:

- An overview of VSE guest support
- How to install and startup VSE guest support
- A description of supported library control functions
- A description of inventory data formats
- A description of customization options
- A table of reason codes generated by VGS

VSE Guest Support Overview

The VSE Guest Server (VGS), a CMS service machine, handles 3494 control requests for one or more VSE guests of z/VM. The intermachine interface between VGS and the VSE guest is APPC/VM. VGS uses Common Programming Interface Communications (CPIC) to handle its APPC/VM functions.

The VSE/ESA guest of z/VM may request library control functions for an IBM 3494 Tape Library Dataserver by making requests to VGS by means of the VSE/ESA LBSERV macro, an application programming interface (API) provided in VSE/ESA 1.3.5 and higher. The library control API uses VSE's cross-partition communication (XPCC) capability for invoking APPC/VM to interact with VGS.

Note: The LBSERV macro of VSE/ESA supports only the IBM 3494 Tape Library Dataserver; it does not support the IBM 3495 Tape Library Dataserver.

The VGS machine supports library control functions for the VSE guest by:

- Managing relationships between virtual devices requested by the VSE system and real devices used by z/VM to fulfill tape mount requests.
- Selecting a real drive of the appropriate type for tape mounts when:
  - More than one device type (3490E, 3590, or 3592) are present in the library.
  - A tape management system running on the VSE guest system is not selecting a real drive.
  - The FSMRMVGC EXEC customization parameters are set up to enable device selection by VGS.

  For details, refer to "Mount a Volume" on page 123 and "Mount from Category" on page 123 and to "DRIVE_LIST_3490" on page 131, "DRIVE_LIST_3590" on page 131, and "DRIVE_LIST_3592" on page 131.
- Requesting library control functions by means of standard DFSMS/VM Removable Media Services (RMS) interfaces.

Figure 18 on page 114 shows the intermachine communication entailed in library-control support for VSE guests and hi-level architecture for VSE guest usage of the 3494 control API. Although only a single VSE Guest machine is shown on the illustration, multiple VSE guests may request library control functions from one VGS machine.

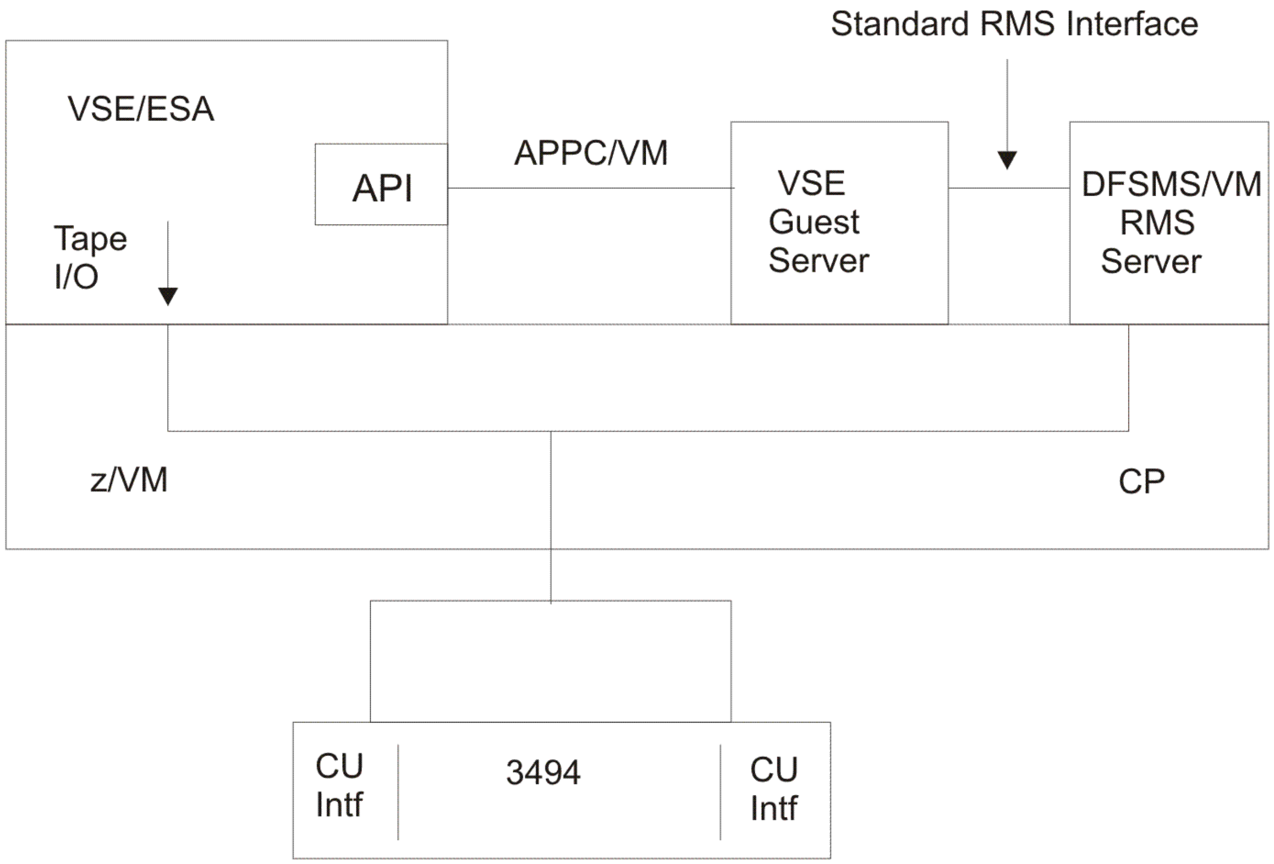
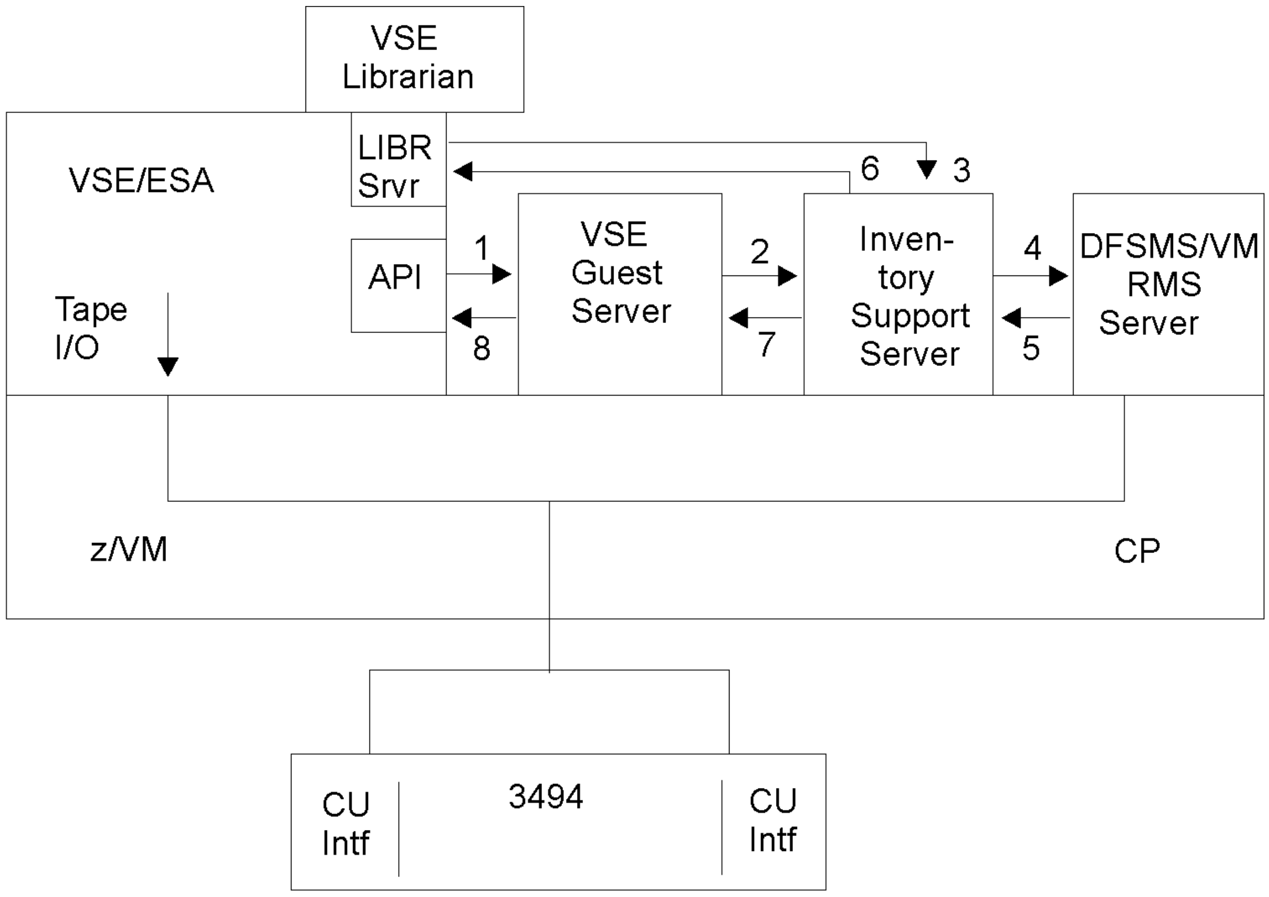


Figure 18. VSE Guest 3494 Support Overview with LBSERV API

VGS supports a full set of library functions, including inventory functions, which entail reading and/or updating inventory lists that reside on VSE/ESA as Librarian members. Because interactions required for processing the inventory functions are complex and may be long-running, a secondary VGS service machine for Inventory Support is required to exploit these functions. In addition, a Librarian Server runs in a VSE/ESA partition. The anatomy of an inventory request is depicted in Figure 19 on page 115. Refer to the following sections of this document for additional details on support for inventory requests:

- "Secondary Inventory Support Server" on page 118
- "Librarian Server for CMS Users" on page 120
- "Query Inventory" on page 126
- "Manage Inventory" on page 128
- "Inventory Data Formats" on page 129
- "LIBRCMS Server List - LIBRCMS SRVNAMES" on page 136



1 - Inventory request sent via LBSERV from VSE guest to VGS

2 - VGS presents inventory request to INVentory machine

3 - INV machine requests Librarian file locks via LIBRCMS Server, reads file Manage request

4 - INV machine sends request(s) to RMS

5 - RMS returns results (inventory list for query, result for manage)

6 - INV writes new copy of Librarian file via LIBRCMS Server

7 - INV notifies VGS of processing complete

8 - VGS replies to LBSERV

Figure 19. VSE Guest Support Detail for Inventory Support

Intermachine Communication Considerations

Inter-machine communication between the VSE guest and the VGS CMS machine involves a simple protocol using APPC/VM. VGS identifies itself as an APPC/VM resource during its initialization processing and is then ready to accept connections. Conversations can then be initiated by the VSE guest machine. Each library request sent to VGS connects, uses, and then severs a discrete intermachine path. The conversation model is shown in Figure 20 on page 116. Note that these conversations are never initiated by VGS.

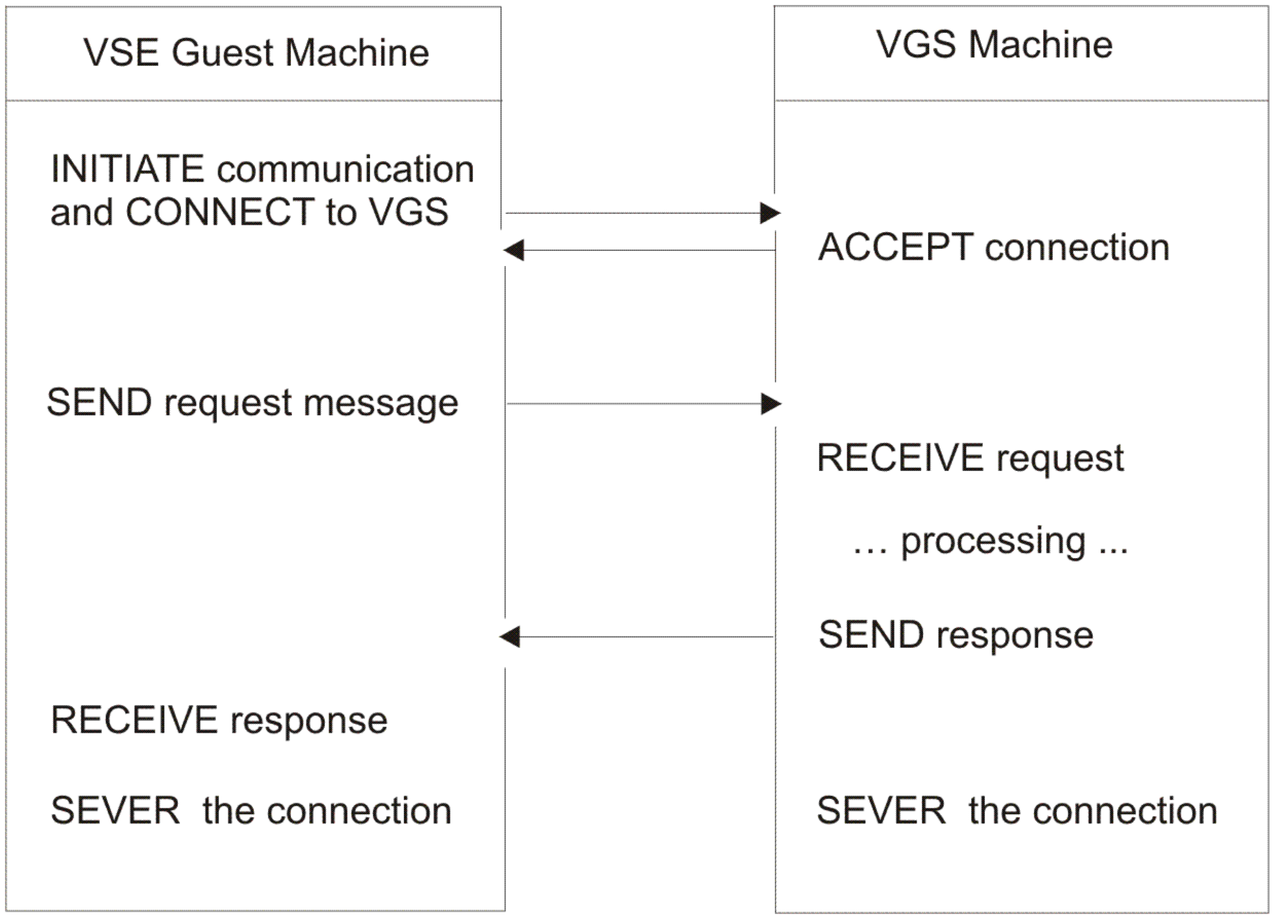


Figure 20. Conversation Model for a VSE Request to VGS

APPC Resource Identification

There is no need for the VGS machine to be a global APPC resource because the VGS machine cannot manipulate drive attachment for VSE guests on other VM nodes in the network. One of the VGS customization options, as described in "Customization Exit - FSMRMVGC" on page 130 allows the user to direct VGS to identify itself as either a LOCAL or a PRIVATE resource during initialization processing.

Note: For communication with a VSE guest, it is essential that VGS manage a LOCAL resource and that proper CP directory authorization is given to VGS to allow it to identify itself as such. See "VGS Machine" on page 116 for further details.

When VGS interacts with a VSE guest through the LBSERV API, VGS **must** manage a resource named "VGLIBSRV.".

Installation and Startup

Instructions are provided in this section for installing and starting-up:

- The VGS machine
- Secondary inventory support machine
- VSE/ESA Librarian Server for CMS users

VGS Machine

The VGS machine is required for all LBSERV-initiated 3494 control functions.

Machine Definition

A sample CP directory entry for VGS is shown in Figure 21 on page 118 and a PROFILE EXEC in Figure 22 on page 118. Local conventions and installed security and directory products will influence your implementation for both. However, the following items are of special interest in defining the VGS machine in the CP directory and/or creating its PROFILE EXEC:

1. Special privilege class

   VGS performs operations such as querying tape drive attachments for VSE guests and attaching/detaching tape drives. Privilege class B is required.

2. Intermachine communication

   To interact with the VSE API, VGS identifies itself as the manager of a Local APPC/VM resource named "VGLIBSRV". The VGS virtual machine requires IUCV statements in its directory entry:

   ```
   IUCV ALLOW
   IUCV *IDENT RESANY LOCAL
   ```

   For its conversations with the VSE machine, VGS uses Common Programming Interface (CPI) Communications. The following statements in the VGS PROFILE EXEC allow it to function as a server:

   ```
   SET SERVER ON
   SET FULLSCREEN OFF
   SET AUTOREAD OFF
   ```

3. Access to DFSMS product code

   VGS uses the CSL interfaces provided in DFSMS/VM to request library functions from the Removable Media Services (RMS) machine of DFSMS/VM. The VGS machine must be authorized to request DFSMS/VM RMS functions and have access to DFSMS/VM product disk. Typically, routines in CSL library FSMPPSI are loaded by the VGS machine's PROFILE EXEC.

4. R/W 191 minidisk

   VGS maintains CMS files with in-process and completed work on a CMS minidisk. The size of these completed-work file, and in turn, the number of cylinders required for this minidisk, can be controlled through a customization option that specifies the amount of time request history data should be kept on file. Other small, temporary files are kept on this disk. VGS accesses it 191 disk as filemode A.

5. Read-access to secondary server's 191 disk

   VGS and the secondary server required for inventory functions need to access each other's 191 disk for read. Each links the other server's 191 as 292 and accesses it as filemode C inside the application code; thus, the link and access statements are not reflected in the sample PROFILE EXEC. This authorization may be accomplished through a security or directory product.

6. $SERVER$ NAMES file on 191 disk

   The file $SERVER$ NAMES is needed on the VGS 191 disk.

   The following entry in the file is sufficient for a LOCAL resource:

   ```
   :nick.VGLIBSRV
   ```

Machine Startup

The VGS machine can be started by any of the following techniques:

- Standard operational protocols used by the installation for autologging service machines
- The DFSMS/VM capability for autologging a foreign host server. See *z/VM: DFSMS/VM Customization* for details.
- Logging on manually, starting the VGS main exec FSMRMVGS, and then disconnecting the machine (#CP DISC).

```
USER VGLIBSRV XXXXXXX 32M 64M BG
*-----------------------------------------------------------*
IPL CMS
IUCV ALLOW
IUCV *IDENT RESANY LOCAL
CONSOLE 01F 3215
SPOOL 00C 2540 READER B
SPOOL 00D 2540 PUNCH B
SPOOL 00E 1403 A
MDISK 0191 3380 2620 5 MDISKA MR
```

Figure 21. Sample CP Directory Entry for VGS

```
/***********************************************************/
TERM MORE 1 1 HOLD OFF
'CP SPOOL READER HOLD'
SET EMSG ON
SET IMSG ON
SET SMSG ON
/* Needed for CPIC server     */
'SET SERVER ON'
'SET FULLSCREEN OFF'
'SET AUTOREAD OFF'
/* Make DFSMS CSLLIB routines available */
'CP LINK DFSMS 1B5 1B5 RR'
'ACCESS 1B5 B'
'RTNLOAD * (FROM FSMPPSI'
'FSMRMVGS'
Exit
```

Figure 22. Sample VGS PROFILE EXEC

Secondary Inventory Support Server

The secondary inventory support server is required for exploiting inventory functions through LBSERV-VGS interfaces.

Machine Definition

A sample CP directory entry for the secondary inventory support server is shown in Figure 23 on page 119 and a PROFILE EXEC in Figure 24 on page 119. Local conventions and installed security and directory products will influence your implementation for both. However, the following items are of special interest in defining the machine in the CP directory and/or creating its PROFILE EXEC:

1. Intermachine communication

   This machine communicates with the VSE/ESA Librarian Server for CMS Users to obtain and replace inventory files. The inventory support server, thus, needs the following IUCV statements in its directory entry:

   IUCV ALLOW
   IUCV *IDENT RESANY GLOBAL

2. Access to DFSMS product code

   This machine uses the command and CSL interfaces provided in DFSMS/VM to request library functions from the Removable Media Services (RMS) machine of DFSMS/VM. Thus, the inventory support machine must be authorized to request DFSMS/VM RMS functions, and it needs access to DFSMS/VM product disk. Typically, routines in CSL library FSMPPSI are loaded by the machine's PROFILE EXEC.

3. R/W 191 minidisk

   The inventory support server keeps interim files for requests in process on its 191 disk.

4. Read-access to VGS's 191 disk

   VGS and the secondary server required for inventory functions need to access each other's 191 disk for read. Each links the other server's 191 as 292 and accesses it as filemode C inside the application code; thus, the link and access statements are not reflected in the sample PROFILE EXEC. This authorization may be accomplished through a security or directory product.

5. XAUTOLOG by VGS

   The default technique for starting the secondary machine is that VGS XAUTOLOGs it during initialization. If local conventions permit this XAUTOLOG procedure, the XAUTOLOG statement can be used in this machine's directory entry to authorize VGS to autolog it.

Machine Startup

The secondary inventory support machine can be started by any of the following techniques:

- Allowing the VGS machine to AUTOLOG it during VGS initialization
- Standard operational protocols used by the installation for autologging service machines
- Logging on manually, starting the main exec FSMRMVGC, and then disconnecting the machine (#CP DISC).

Orderly sequencing down of this machine with VGS is important. This machine needs to restart whenever VGS restarts. (The converse is not true; VGS does not need to restart if the secondary machine restarts.) To ensure that restart is properly serialized, the following measures are taken:

1. VGS attempts to AUTOLOG the inventory support machine during VGS startup; if the secondary machine is already logged on, VGS sends the secondary machine a "restart" order. This insures that the machines are synchronized in terms of in-process requests.
2. As part of its orderly termination processing, VGS sends the secondary machine a "shutdown" request. The secondary machine ends its processing EXEC and logs off.

```
USER VGINVHLP XXXXXXX 32M 64M G
XAUTOLOG VGLIBSRV
*---------------------------------------------------------------*
IPL CMS
IUCV ALLOW
IUCV *IDENT RESANY GLOBAL
CONSOLE 01F 3215
SPOOL 00C 2540 READER B
SPOOL 00D 2540 PUNCH B
SPOOL 00E 1403 A
MDISK 0191 3380 636 5 MDISK6 MR
```

Figure 23. Sample CP Directory Entry for Inventory Support Machine

```
/*********************************************************/
TERM MORE 1 1 HOLD OFF
'CP SPOOL READER HOLD'
SET EMSG ON
SET IMSG ON
SET SMSG ON
*/  Server characteristics  */
'SET SERVER ON'
'SET FULLSCREEN OFF'
'SET AUTOREAD OFF'
/* Make DFSMS CSLLIB routines available */
'CP LINK DFSMS 1B5 1B5 RR'
'ACCESS 1B5 B'
'RTNLOAD * (FROM FSMPPSI'
'FSMRMVGH'
Exit
```

Figure 24. Sample Inventory Support Machine PROFILE EXEC

Librarian Server for CMS Users

To enable VSE guest exploitation of inventory support functions through LBSERV-VGS interfaces, the following parts are required from the LIBRCMS PACKAGE available from the DFSMS/VM website, IBM z/VM and VM-related licensed products and features (https://www.ibm.com/vm/related).

- LIBRCMS MODULE - CMS support code to request VSE Librarian functions for a CMS user.
- LIBUME TXTAMENG - CMS message repository
- LIBRCMSI JCL - Job stream to catalog the VSE/ESA Librarian server on VSE/ESA.
- LIBRCMS JCL - Sample JCL to start the Librarian Server on VSE/ESA.

Install Instructions

After downloading the LIBRCMS package, it will be in VMARC format. See z/VM Downloads (https://www.vm.ibm.com/download/) for instructions on extracting the LIBRCMS parts.

LIBRCMS MODULE

Install this part on the DFSMS/VM product disk along with other parts related to RMS and VGS. The LIBRCMS functions are thereby accessible to the VGS secondary inventory support machine. No further activity is required for the CMS side.

LIBUME TXTAMENG

Also install this part on the DFSMS/VM product disk.

LIBRCMSI JCL

Send this job to each VSE guest that is to request 3494 support through VGS. This job catalogs the unique LIBRCMSV phase to PDR2.COMM.

Server Setup and Execution

On VSE/ESA, a free partition of 1M minimum size is required for running the Librarian Server. A dynamic partition may be used. At least 50K should be allocated by means of the SETPFIX JCL command to fix pages needed for APPC/VM communication with the inventory support server.

For LIBRCMS:

1. APPC/VM definitions are not required when IPLing VSE/ESA (SET XPCC commands.)
2. The name "LIBRCMS" must **not** be defined by SET XPCC TARGET at IPL time.

The additional member of the LIBCMS PACKAGE, LIBRCMS JCL, provides sample job control for starting the Librarian Server, similar to the example shown in Figure 25 on page 120.

```
* $$ JOB JNM=LIBRCMS,CLASS=Y,LDEST=*,DISP=L
// JOB JNM=LIBRCMS
// LIBDEF *,SEARCH=(PRD2.COMM)
// SETPARM SRVNAME='LIBRC1'
// SETPFIX LIMIT=100K
// EXEC LIBRCMSV,SIZE=AUTO
/*
/&
* $$ EOJ
```

Figure 25. Sample LIBRCMS JCL for VSE/ESA

You must edit a LIBRCMS SRVNAMES cross-reference file on the VGS secondary inventory support machine's 191 disk if VGS is to service inventory requests for multiple VSE guests. See “LIBRCMS Server List - LIBRCMS SRVNAMES” on page 136. The SRVNAME specified in the SETPARM statement must be unique for each VSE guest.

Supported Library Control Functions

The VGS services these types of requests for library control:

- Query a volume, checking a single library
- Query a volume, checking all attached libraries
- Query a category count
- Query the 3494 status
- Query status of a drive
- Mount a volume
- Mount from category
- Release a drive
- Demount a drive
- Cancel a mount
- Eject a volume
- Set a volume category
- Query the inventory
- Manage the inventory
- Stopping the VGS program
- Tracing the VGS program

Query Volume, Single Library

The QUERY VOLUME, SINGLE Library (QS) request is used to determine if a specified volume is located in the specified or default library. In job steps involving a single library device but multiple library volumes (or cartridges), VSE can issue a QS request for each volume, prior to requesting mounts, to ensure that all volumes are located in the same target library, prior to beginning the job step.

If a specific library name is not specified as input in the request, the default library is queried. If found, additional information on status, category, and media type for the volume is returned to the caller. See Table 17 on page 127.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Query Volume, All Libraries

The QUERY VOLUME, ALL Libraries (QA) request is used to determine the library location of a specified volume. In job steps involving multiple volumes, VSE may issue a QA request to find the library location of the first volume and then QS requests for subsequent volumes to ensure that all volumes are in the same library.

All libraries are queried, until the volume is found, and the library location is returned to the caller, along with additional information on status, category, and media type for the volume. See Table 17 on page 127.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Query Category Count

The QUERY COUNT (QC) request is used to determine the count of volumes in the 3494 inventory or in a named source category. Valid source category names are:

SCRATCH
Default scratch pool

SCRATCHx
RMS scratch pool name, where x is 0-F

SCRATCHxx
VSE scratch pool name, where xx is 00-31

If the USE_VSE_CATEGORIES is set to YES, then VGS will assign the volume to the appropriate VSE category.

If USE_VSE_CATEGORIES is set to NO, then VGS will map the xx range 00-15 to a physical RMS scratch pool SCRATCHx, where x is 0-F. In this case, VSE range 16-31 will be ignored.

INSERT
Cartridges that have entered 3494 but have not been set to a specified category

hexvalue
Hexadecimal category in the range of 0000 to FFFF.

If no category is specified, the count for the entire 3494 inventory is returned.

Note: Because of RMS limitations, VGS is only able to provide counts for these additional source categories that are supported through the native VSE library control device driver by specifying the *hexvalue* corresponding to:

- PRIVATE - active tapes
- MANEJECT - manually ejected tapes

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Query 3494 Drives

The QUERY DRIVE (QD) request is used to determine the status of a specified tape drive. In order to obtain the tape drive status, it is necessary that RMS be able to attach this drive. Thus, the status cannot be provided unless the drive is either FREE or is currently attached to the requesting machine.

Ordinarily, the CUU passed to VGS by the LBSERV API is a virtual CUU. VGS determines if there is a real drive attached as this CUU.

- If there is no real drive attached at the virtual CUU that LBSERV provided, VGS assumes that the virtual CUU field is to be taken as a real drive address and uses the CUU sent by LBSERV as the real address to be queried. This drive must be FREE or the request will fail. (Communication with the 3494 for a drive query must be accomplished on the channel path for that drive.)
- If there is a real drive currently attached, VGS queries the real real attached as the CUU specified by LBSERV.

  Note: To accomplish the 3494 communication required for the query, the drive must be detached from the VSE guest and attached to VGS and then RMS. Thus, a query should not be requested while the drive is in use. Essentially, the drive should be ASSIGNed to the partition/job requesting the query drive function, just as it would be for a mount, to ensure serialization of its use.

The following information is returned to the requester; see Table 17 on page 127 for details:

- Status of the tape drive
- Volser of mounted volume
- Category of mounted volume

Note that in the case of a free drive, it is typically improbable that there will be a mounted volume, unless the DETACH operation used to free the drive specified the LEAVE option, avoiding rewind/unload which in turn queues a library demount operation. When the drive is attached to the requesting machine when the Query Drive is issued, DETACH operations use the LEAVE option in transferring drive attachment for 3494 communication.

Query 3494 Status

The QUERY LIBRARY (QL) request is used to determine the operational status of the 3494. See Table 17 on page 127 for information on the returned status data.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Mount a Volume

The MOUNT VOLUME (MV) request is used to have a cartridge with a specified external label mounted on a library device attached to the VSE guest at a specified virtual address.

As part of the 3494 mount processing, the category for the mounted cartridge can be optionally changed to a new (target category). The following category names are valid targets:

PRIVATE
Active tape, maps to the RMS VOLspecific category

SCRATCH
Default scratch pool

SCRATCHx
RMS scratch pool name, where x is 0-F

SCRATCHxx
VSE scratch pool name, where xx is 00-31

If the USE_VSE_CATEGORIES is set to YES, then VGS will assign the volume to the appropriate VSE category.

If USE_VSE_CATEGORIES is set to NO, then VGS will map the xx range 00-15 to a physical RMS scratch pool SCRATCHx, where x is 0-F. In this case, VSE range 16-31 will be ignored.

hexvalue
A hexadecimal category in the range of 0000 to FFFF.

If a real device is already attached to the VSE guest at the specified virtual address (e.g., the device is dedicated to the guest), that real device is specified by VGS in the mount request to RMS.

If the VSE guest does not have a tape attached at the specified virtual address, and VGS is not enabled to perform device selection, then VGS does not specify a real device in its request to RMS, and RMS finds a free device to satisfy the request. The request will fail if there are no free drives available. Optionally, the RMS device selection exit may be used in selecting the real device, if any free device may not be suitable.

Note that when VGS performs drive selection, the media type in the request from the VSE machine is used to determine the appropriate device type (3490E, 3590, or 3592). Whether or not VGS selects a real drive is determined thru VGS drive_list customization parameters. Refer to "DRIVE_LIST_3490" on page 131, "DRIVE_LIST_3590" on page 131, and "DRIVE_LIST_3592" on page 131 for further details.

After the mount is complete, RMS attaches the real device to the VSE guest at the virtual *cuu* provided as input from VSE (and passed to by VGS RMS through a mount interface option, along with machine userid of the VSE guest.)

VSE is expected to issue a Release Device request when finished with tape drives so that non-dedicated drives will become free again.

Mount from Category

The MOUNT CATEGORY (MC) is used to request that the next cartridge in a specified scratch category mounted on a library device attached to the VSE guest at a specified virtual address.

One of the following source category names may be specified:

SCRATCH
Default scratch pool

SCRATCHx
RMS scratch pool name, where x is 0-F

SCRATCHxx
VSE scratch pool name, where xx is 00-31

If the USE_VSE_CATEGORIES is set to YES, then VGS will assign the volume to the appropriate VSE category.

If USE_VSE_CATEGORIES is set to NO, then VGS will map the xx range 00-15 to a physical RMS scratch pool SCRATCHx, where x is 0-F. In this case, VSE range 16-31 will be ignored.

hexvalue
A hexadecimal category in the range of 0000 to FFFF.

As part of the 3494 mount processing, the category for the mounted cartridge can be optionally changed to a new (target category). The following category names are valid targets:

PRIVATE
Active tape, maps to the RMS VOLspecific category

SCRATCH
Default scratch pool

SCRATCHx
RMS scratch pool name, where x is 0-F

SCRATCHxx
VSE scratch pool name, where xx is 00-31

If the USE_VSE_CATEGORIES is set to YES, then VGS will assign the volume to the appropriate VSE category.

If USE_VSE_CATEGORIES is set to NO, then VGS will map the xx range 00-15 to a physical RMS scratch pool SCRATCHx, where x is 0-F. In this case, VSE range 16-31 will be ignored.

hexvalue
A hexadecimal category in the range of 0000 to FFFF.

The VGS default is to specify that the cartridge mounted from a scratch pool be set to PRIVATE status if there is no target category specified by the requester.

If a real device is already attached to the VSE guest at the specified virtual address (e.g., the device is dedicated to the guest), that real device is specified by VGS in the mount request to RMS.

If the VSE guest does not have a tape attached at the specified virtual address, and VGS is not enabled to perform device selection, then VGS does not specify a real device in its request to RMS, and RMS finds a free device to satisfy the request. The request will fail if there are no free drives available. Optionally, the RMS device selection exit may be used in selecting the real device, if any free device may not be suitable.

Note that when VGS performs drive selection, the media type in the request from the VSE machine is used to determine the appropriate device type (3490E, 3590, or 3592). Whether or not VGS selects a real drive is determined thru VGS drive_list customization parameters. Refer to "DRIVE_LIST_3490" on page 131, "DRIVE_LIST_3590" on page 131, and "DRIVE_LIST_3592" on page 131 for further details.

After the mount is complete, RMS attaches the real device to the VSE guest at the virtual *cuu* provided as input from VSE (and passed to by VGS RMS through a mount interface option, along with machine userid of the VSE guest.)

The external volser of the mounted tape is returned to the requester.

VSE is expected to issue a Release Device request when finished with tape drives so that non-dedicated drives will become free again.

Release a Drive

The RELEASE DRIVE (RD) request is used to indicate that a tape drive used in a prior mount request is no longer in use by the VSE job or job step. VGS releases the real device attached at the specified virtual address, so that it may be used by another VM guest (or another VSE job), IF AND ONLY IF the real device was not dedicated to the requesting VSE guest at the time the mount was requested. (The VGS machine maintains a device-usage file, mapping real devices that have been used in mount requests to an associated VSE-guest user ID, virtual address, and "previously-attached/dedicated" flag.)

If explicit_demount=YES is specified in the FSMRMVGC configuration file, VGS will issue a demount to RMS for the real device.

Cancel a Mount

The CANCEL MOUNT (CM) request is used to indicate that the mount request for a specified volume serial on a specified virtual *cuu* is to be cancelled. If VGS has an in-process request on file for this volume serial, virtual address, and VSE userid combination, it will send a mount cancel request to RMS, using the mount-cancel programming interface. If the mount has already completed and it is too late for the request to be cancelled, VGS **does not** request a dismount; the VSE guest must issue a rewind/unload for the dismount operation to occur.

Eject a Volume

The EJECT VOLUME (EV) request is used to remove a volume from either the specified or default library. If the volume is not found in the specified or default library, the request will fail. When the specified volume is found, it is place in the EJECT category and moved to the output area for removal from the library by the operator.

If either the convenience I/O station installed, or the high-capacity I/O facility is defined, whichever is present is used. If both are present, then volumes are directed to the convenience I/O station. Explicit use of the high-capacity I/O facility is available through the Set Volcat (SV) function.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Set a Volume Category

The SET VOLCAT (SV) request is used to assign a volume to a specified target category. If an optional source category is specified, the category designation for the volume is changed only if the volume is currently assigned to the specified source category.

Valid target category names are:

PRIVATE
Active tape category

EJECT
Move to the convenience I/O station

EJECTB
Move to the high-capacity I/O facility

SCRATCH
Default scratch pool

SCRATCHx
RMS scratch pool name, where x is 0-F

SCRATCHxx
VSE scratch pool name, where xx is 00-31

If the USE_VSE_CATEGORIES is set to YES, then VGS will assign the volume to the appropriate VSE category.

If USE_VSE_CATEGORIES is set to NO, then VGS will map the xx range 00-15 to a physical RMS scratch pool SCRATCHx, where x is 0-F. In this case, VSE range 16-31 will be ignored.

Valid source category names are:

PRIVATE
Active tape category

INSERT
Tapes that have just entered the 3494

SCRATCH
Default scratch pool

SCRATCHx
RMS scratch pool name, where x is 0-F

SCRATCHxx
VSE scratch pool name, where xx is 00-31

If the USE_VSE_CATEGORIES is set to YES, then VGS will assign the volume to the appropriate VSE category.

If USE_VSE_CATEGORIES is set to NO, then VGS will map the xx range 00-15 to a physical RMS scratch pool SCRATCHx, where x is 0-F. In this case, VSE range 16-31 will be ignored.

hexvalue
Hexadecimal category in the range 0000 to FFFF

Note: INSERT as a source category is not supported by a Ptp VTS.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Query Inventory

Query Inventory (QI) allows the user to request inventory data on volumes currently assigned to a specified category, in a specified library. If source category name is omitted as an input argument, then inventory data for the entire 3494 is provided. If the library name is omitted, then the default 3494 is assumed for this request.

The return code indicates if the query was successful, that is, if inventory data was obtained from the 3494. This request type has the potential to be long-running, and a response is not returned to the VSE guest until the request completes.

The inventory data is placed in a Librarian-managed file in the library.sublibrary specified in the VSE_Q_LIB and VSE_Q_SLIB customization parameters,as described in "VSE_Q_LIB" on page 134 and "VSE_Q_SLIB" on page 134. The library and sublibrary are expected to have been predefined. A QI request fails if the customization options do not specify a defined library.sublibrary.

The member name created (or rebuilt) by a QI request is the up-to-8-character name determined by the source category for the request:

Table 16. Member Name by Source Category

Source Category	Member Name
(blank)	ALL
SCRATCH*xx*	SCR*xx*
SCRATCH	SCR
INSERT	INSERT
hexvalue	*hexvalue*

Refer to "Query Inventory Output Files" on page 129 for information on content and format of inventory data.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Note:

Because of RMS limitations, VGS is only able to provide inventory data for these source categories that are supported through the native VSE library control device driver through the use of the *hexvalue* category corresponding to:

- PRIVATE - active tapes
- MANEJECT - manually ejected tapes

Table 17. Output from Query Requests

Field	Size	Contents	Applies to
Status	4	When Library is queried: • 0000 - Automated mode • 0100 - Paused mode • 0200 - Manual mode When Volume is queried: • 0000 - No special condition • 0002 - Assigned with the Fast Ready Attribute set • 0004 - Manually ejected • 0008 - Used in manual mode • 0010 - Missing or damaged label • 0020 - Misplaced • 0040 - Being audited • 0080 - Queued for audit • 0100 - Being ejected • 0200 - Queued for eject • 0400 - Being demounted • 0800 - Queued for demount • 1000 - Being mounted • 2000 - Queued for mount • 4000 - Mounted • 8000 - Inaccessible When Drive is queried: • 0000 - Installed and available • 0001 - When device is tape encryption capable • 0100 - Not installed or available	Query lib (QL) Query vol all libs (QA) Query vol 1 lib (QS) Query drive (QD)
Volume category	4	Category of queried volume or category of volume mounted on queried drive.	Query vol all libs (QA) Query vol 1 lib (QS) Query drive (QD)
Volume class	4	Class of the queried volume (3480, 3590, or 3592).	Query vol all libs (QA) Query vol 1 lib (QS)
Volume type	4	Name of volume type, as maintained by library manager (CST1, CST2, or CST3).	Query vol all libs (QA) Query vol 1 lib (QS)

Table 17. Output from Query Requests (continued)

Field	Size	Contents	Applies to
Volume mounted	6	Volser of volume mounted on queried device.	Query drive (QD)
Volume count/ cache percentage	5	For a Query Category (QC), the number of volumes in a queried category. (See next field if information is greater than 5 characters long.) For a Query Lib (QL), the cache percentage information.	Query category (QC) Query lib (QL)
Extended volume count	8	The number of volumes in a queried category, if that information is greater than 5 characters long. In this case, the "Volume count" field will be zero.	Query category (QC)

Manage Inventory

Manage Inventory (MI) allows the user to request that a list of volumes in a specific 3494 be assigned to a specified target category. Lists created by the Query Inventory function are pre-formatted for use as input to the Manage Inventory request.

The requestor must supply a target category and the member name for the list file to be used as input.

The list of volsers is read from a Librarian-managed file in the library.sublibrary specified in the VSE_M_LIB and VSE_M_SLIB customization parameters, as described in "VSE_M_LIB" on page 134 and "VSE_M_SLIB" on page 134. The library and sublibrary are expected to have been predefined. A MI request fails if the customization options do not specify a defined library.sublibrary.

The return code indicates if the manage request was processed successfully. This request type has the potential to be long-running, and a response is not returned to the VSE guest until all volsers in the list are processed.

The input list is updated to reflect the outcome of the category change for each volume. Refer to "Manage Inventory Input Files" on page 129 and "Manage Inventory Output Files" on page 130 for information on content and format of inventory data.

When enhanced library control is being used, there is no requirement for a free drive for library communication. Otherwise, RMS requires a free drive to interact with the 3494.

Stopping the VGS Program

The STOP_VGS EXEC is provided to request orderly shutdown of VGS processing; the exec is issued from another CMS machine. Either an immediate or a quiesced shutdown is possible:

```
     STOP_VGS I
     STOP_VGS Q
```

One of the two parameters, I or Q, must be specified. With the quiesced shutdown, in-process requests are completed but no new requests are accepted by VGS. Immediate shutdown does not wait for in-process work to complete.

Tracing the VGS Program

The STOP_VGS EXEC can also be used to provide a trace level request for VGS processing; the exec is issued from another CMS machine. One of the following three trace levels can be specified:

STOP_VGS T0 - To turn off diagnostic messages
STOP_VGS T1 - To turn on level 1 diagnostic messages
STOP_VGS T9 - To turn off level 9 diagnostic messages

For additional information on VGS trace levels, see "TRACE_LEVEL" on page 133.

Inventory Data Formats

This section includes descriptions of the inventory data formats.

Query Inventory Output Files

In the files that are created by a Query Inventory request, an 80-byte inventory record applies to a cartridge and contains the following information.

- External volume label (CHAR 6)
- Media type (CHAR 4):
  - CST1 - Standard Cartridge Tape
  - CST2 - Extended Capacity Cartridge Tape
  - CST3 - Standard Magstar Cartridge Tape
  - CST4 - Extended Magstar Cartridge Tape
  - CST5 - Standard TotalStorage Enterprise Cartridge Tape
  - CST6 - Standard TotalStorage Enterprise WORM Cartridge Tape
  - CST7 - Short TotalStorage Enterprise Cartridge Tape
  - CST8 - Short TotalStorage Enterprise WORM Cartridge Tape
  - CST9 - Extended Enterprise Tape Generation 2
  - CSTA - Extended WORM Enterprise Tape Generation 2
  - CSTB - Enterprise Tape Generation 3
  - CSTC - WORM Enterprise Tape Generation 3
  - CSTD - Short Enterprise Tape Generation 3
- Special attribute byte, EBCDIC represented bit string (CHAR 8)
  - Bit 0, 1 = Volume is present in library, but inaccessible
  - Bit 1, 1 = Volume is mounted or queued for mount
  - Bit 2, 1 = Volume is in eject-pending state
  - Bit 3, 1 = Volume is in process of ejection
  - Bit 4, 1 = Volume is misplaced
  - Bit 5, 1 = Volume has unreadable label or no label
  - Bit 6, 1 = Volume was used during manual mode
  - Bit 7, 1 = Volume was manually ejected
- Friendly category name (CHAR 10)
- LM hex category number, in EBCDIC representation (CHAR 4)

In the file that is built in response to a Query Inventory request, the fields in each record are separated by a blank character to enhance their readability by humans. A sample file record looks like this:

```
CS0010 CST2 01000000 PRIVATE    FFFF
```

A header with the time-of-day of list creation is inserted as the first record in the list.

Manage Inventory Input Files

A file submitted for use in a Manage Inventory request has the format requirement that each 6-character external volume serial number in the list start in column 1 of a file record. The remaining space in each 80-character record is ignored as input. Thus, the possibility exists for sending the Query Inventory output file back to VGS as input to a Manage Inventory request. A record in a Manage Inventory input file may look like this:

```
CS0010 CST2 01000000 SCRATCH0   0080
```

Or it might simply be this:

```
CS0010
```

A header record (as described for Query Inventory output) may be optionally present in the input list; one is inserted during list processing if there was none. Any record starting with a character asterisk (*) is not considered a valid input data record and is ignored.

Manage Inventory Output Files

When a Manage Inventory request has been completed, a return code and reason code are supplied to indicate that processing is complete. This RC/RSN information attest to the success in processing the request (for example, input was valid, file was found, and so forth.) The actual outcome of transferring each volume to a new target category is reflected within the file itself. VGS updates the file by adding a results message in each file record, starting in column 38. An output file record reflecting success looks like this:

```
CS0010 CST2 01000000 PRIVATE    FFFF ** CATEGORY CHANGED TO EJECT
```

An example of a failure looks like this:

```
CS0010 CST2 01000000 PRIVATE    FFFF ** CATEGORY NOT CHANGED, RSN=3340
```

File Name Summary

Table 18 on page 130 summarizes the naming conventions for inventory files used in the Query Inventory and Manage Inventory functions. Libraries and sublibraries must be predefined.

Table 18. Summary of Inventory File Name Conventions

Function	Library	Sublibrary	Member	Type
Query Inventory	Locally selected	Locally selected, 3494 name suggested	Determined by *source_cat* parameter of QI request	L
Manage Inventory	Locally selected	Locally selected, 3494 name suggested	Locally selected and specified as *membname* parameter of MI request	L

PI

Customization Options

The VGS machine calls two user exits. In addition, a library configuration file that cross references VSE and DFSMS/VM library names is expected to reside on the VGS 191 disk if there is more than one attached 3494. When multiple VSE guests are using VGS for inventory functions, a LIBRCMS server cross-reference file is required also, on the secondary server's 191 disk.

Customization Exit - FSMRMVGC

This exit is used for customizing VSE Guest Server support to meet specific installation needs. Some of the customization variables enable required resource identification and must be set locally to describe critical elements of your environment. Others are options that you may modify to take advantage of special features or to change default processing values (for example, timings that can be use to tune performance.)

When the VGS inventory functions Query Inventory and Manage inventory are to be exploited (typically by a tape management product running on the VSE guest), a secondary Inventory Support machine must be defined. A subset of the customization variables apply only when Inventory functions are used that the additional machine is defined.

Note: When editing FSMRMVGC to perform your local customization, do not alter any lines EXCEPT those marked with <=== in the comment to the right of the REXX assignment statements. Ensure that there is a non-blank value for each customization option.

DRIVE_LIST_3490

The list of real 3490E CCUUs in 3494s. The devices may be specified as up-to-4-character entries, with blank delimiter and the entire string enclosed in quotes. For example: DRIVE_LIST_3490 = '290 291 292 293' If the 3494(s) contain one device type, you should retain the default settings with all drive_list variables set to blank. Drive_lists are THE ONLY VARIABLES for which blank settings are allowed. If there are more devices than can be provided on one line, duplicate the line as many times as necessary, remembering to concatenate the previous line.

Important Note:

Typically a tape management product that manages library control for the VSE/ESA guest handles the selection of real drives and VGS involvement is not required for drive selection. When such a tape management system is in use, leave this option blank, even if more than one device type (3490E, 3590, or 3592) are installed in the library.

The VGS device selection process enabled through use of this option is effective only when there is one 3494 library used by VSE guests. VGS device selection does not support multiple library scenarios, including VTS subsystems installed with another 3494 library in the same physical 3494 configuration.

DRIVE_LIST_3590

The list of real 3590 CCUUs in 3494s. The devices may be specified as up-to-4-character entries, with blank delimiter and the entire string enclosed in quotes. For example: DRIVE_LIST_3590 = '290 291 292 293' If the 3494(s) contain one device type, you should retain the default settings with all drive_list variables set to blank. Drive_lists are THE ONLY VARIABLES for which blank settings are allowed. If there are more devices than can be provided on one line, duplicate the line as many times as necessary, remembering to concatenate the previous line.

Important Note:

Typically a tape management product that manages library control for the VSE/ESA guest handles the selection of real drives and VGS involvement is not required for drive selection. When such a tape management system is in use, leave this option blank, even if more than one device type (3490E, 3590, or 3592) are installed in the library.

The VGS device selection process enabled through use of this option is effective only when there is one 3494 library used by VSE guests. VGS device selection does not support multiple library scenarios, including VTS subsystems installed with another 3494 library in the same physical 3494 configuration.

DRIVE_LIST_3592

The list of real 3592 CCUUs in 3494s. The devices may be specified as up-to-4-character entries, with blank delimiter and the entire string enclosed in quotes. For example: DRIVE_LIST_3592 = '290 291 292 293' If the 3494(s) contain one device type, you should retain the default settings with all drive_list variables set to blank. Drive_lists are THE ONLY VARIABLES for which blank settings are allowed. If there are more devices than can be provided on one line, duplicate the line as many times as necessary, remembering to concatenate the previous line.

Important Note:

Typically a tape management product that manages library control for the VSE/ESA guest handles the selection of real drives and VGS involvement is not required for drive selection. When such a tape management system is in use, leave this option blank, even if more than one device type (3490E, 3590, or 3592) are installed in the library.

The VGS device selection process enabled through use of this option is effective only when there is one 3494 library used by VSE guests. VGS device selection does not support multiple library scenarios, including VTS subsystems installed with another 3494 library in the same physical 3494 configuration.

EXPLICIT_DEMOUNT

An indicator for including a synchronous demount call to the RMS machine during release request processing. Supplied default is NO, indicating release processing will only detach a device from the requesting VSE quest if appropriate.

LOCAL_VGS_NAME

The name used by the VGS exec to identify itself as the manager of this resource. When the VSE guest communicates with VGS through the LBSERV macro interface, the LOCAL_VGS_NAME must be VGLIBSRV. Supplied default value is VGLIBSRV and should not be changed.

MI_VALUE

The time-out value in seconds, after which the VGS machine will sever a request from a client, if a connection has been received but no request data is sent. Supplied default value is 180 seconds (three minutes).

NONMOUNT_PACE_MAKER

The minimum amount of time, in tenths of a second, that must elapse between check-back calls to the RMS machine for any non-mount synchronous requests (for example, set volume category or SV) to see if it has completed. Check-back calls sent too frequently add unnecessary overhead for the RMS machine. Supplied default is 100 tenths of a second (10 seconds).

QS_SYNC

An indicator for whether querying specific volume requests should be issued synchronously to the RMS machine. Supplied default is YES; NO to issue request asynchronously to RMS.

RESULTS_MAX_AGE

A criterion for purging completed mount requests from the history file, VGSWTD RESULTS A. The numeric value represents the age in days before records are pruned. Supplied default is 1 day.

RM_WORK_DIRECTORY

The name of the RMS work directory, as provided in the DFSMS master control file (DGTVCNTL DATA), where the bulk file resides. The bulk filename must be provided by the VSE Guest using the member name field of the ‘SV’ (Set Volume category) request. VGS will use ‘BULK’ as the filetype.

SCRTCH_POOL

The name of the DFSMS/VM RMS scratch category to be accessed for mount requests when the designation SCRATCH is used in request syntax instead of a SCRACTHx, where x is a designated poolid. DFSMS/VM RMS provides 16 scratch categories, or pools, where x is 0-A. When VSE shares the library with z/VM it may be desirable to ensure that VSE uses a different default scratch pool than that used by z/VM, as specified in the DFSMS/VM Control File. The supplied default for VSE is SCRATCH0.

SETVOLCAT_SYNC

An indicator for whether set volume category requests should be issued synchronously to the RMS machine. Supplied default is YES; NO to issue request asynchronously to RMS.

SOFT QUERY REAL/VIRT

These two configuration statements, SOFT QUERY REAL and SOFT QUERY VIRT, allow for a real drive device to be mapped to a virtual one. This allows soft query to supply a 3 character device ID (which VSE only supports), in order to determine information on the real tape device when it is defined with 4 characters (CCUU). There must be a one-to-one mapping between the two lists.

TELL_ON_ERROR

An indicator for requesting that the TELL_USERID receive a message when a non-fatal error occurs. Supplied default is 'N' (no); 'Y' (yes) to receive a message.

TELL_ON_FATAL

An indicator for requesting that the notify userid receive a message when a fatal error occurs. Supplied default is 'N' (no); 'Y' (yes) to receive a message.

TELL_USERID

The user ID (or user ID/node ID combination -- xxxxxxxx/xxxxxxxx separated with /) to receive messages about unwholesome server events, IF such notification is requested by means of further options. The default user ID specified initially is ????????. Until you modify this option to reflect a user ID at your installation, or if the tell_userid field is not changed, the messages are displayed only on the server machine console on which the error occurs. To notify multiple users when errors occur, you may create a names file on the server's A-disk and use a nickname to refer to a list of user IDs.

TRACE_LEVEL

The tracing level requested, in the range 0-9. Current supported tracing levels are:

0 - No diagnostic messages
1 - Diagnostic messages showing request states
9 - Full range of diagnostic messages

Each trace level also causes lower numbered levels to also be turned on. For example, 9 will cause 1-8 to be turned on.

USE_VSE_CATEGORIES

An indicator for whether VGS should use VM SCRATCH categories or VSE scratch categories. A 'YES' value indicates to use VSE, 'NO' indicates to use VM.

VGS_RES_TYPE

The type of resource that the main FSMRMVGS exec is to manage; specify LOCAL. (IUCV authorizations are required in the directory entry for the VGS machine as well and for its communication partners when a LOCAL resource is managed.)

V_PACE_MAKER

The minimum number of seconds that must elapse between check-back calls to the RMS machine to see if a mount is complete. Check-back calls sent too frequently add unnecessary overhead for the RMS machine. Supplied default is 10 seconds.

The remaining variables are used to configure the Inventory Helper ID machine.

These variables require customization if you plan to exploit Query and Manage Inventory functions and have defined the secondary Inventory Support server.

Note: If you DO NOT need to define the secondary support machine, change the HELPER_ID value as described in "HELPER_ID" on page 134.

HELPER_ID

The user ID of the Inventory Support machine that supports processing of Manage Inventory and Query Inventory requests. If an Inventory Support machine is not desired, that is, there is no need to maintain inventory lists on, or send lists from the VSE guest, specify a eight asterisks '********'. The supplied default is VGINVHLP.

INV_MAX_TIME

The maximum number of seconds VGS is to keep in-process Query Inventory and Manage Inventory requests active, before responding to the requester with a timeout error. This is a precaution in the unlikely event of the secondary service machine going down between start and finish of the processing of a request. VGS will respond with a timeout error to VSE. Supplied default is 1200 seconds (20 minutes).

I_PACE_MAKER

The minimum number of seconds that must elapse between searches by the Inventory Support machine for new inventory requests. The objective is to moderate the pace of machine processing while allowing it to look for new work often enough to not impact expected response time. Supplied default is 10 seconds.

MASTER_NAME

The user ID of the DFSMS/VM RMS Master machine. The Inventory machine verifies that files received as inventory reports are sent by this user ID. Supplied default is RMSMASTR.

REPORT_FNAME

The file name for the report files sent to the Inventory machine reader from RMS Master machine. By convention, the file name is the value of the DFSMS/VM control file variable, GV_GLOBAL_RESOURCE_ID. This value is used during Query Inventory processing to perform validity checking of reader files. Supplied default is DFSMS001, the control file default.

VSE_M_LIB

The name of the predefined VSE Librarian library in which manage list members are to be found. This library must be defined or Manage Inventory requests will fail. Supplied default is VGSINV.

VSE_M_SLIB

The name of the predefined VSE Librarian sublibrary in which manage list members are to be found. This sublibrary must be defined or Manage Inventory requests will fail. A suggested convention is to use the library name as the sublibrary. Supplied default is ATL1.

VSE_Q_LIB

The name of the predefined VSE Librarian library in which query list members will be written. This library must be defined or Query Inventory requests will fail. Supplied default is VGSINV.

VSE_Q_SLIB

The name of the predefined VSE Librarian sublibrary in which query list members will be written. This sublibrary must be defined or Query Inventory requests will fail. A suggested convention is to use the library name as the sublibrary. Supplied default is ATL1.

Authorization Exit - FSMRMVGA

This exit as shipped authorizes the request by returning an auth code of 0, which tells the main VGS EXEC to maintain the connection and process the request. Any non-0 value will cause main VGS processing to send a failing message to the requester and end the conversation. Unauthorized attempts to connect are reported and logged on the console.

The customer may screen client based on the information input to this EXEC:

- Userid of the requester
- The 8-character control-block identifier in the request message
- The request type
- The VSE ID of the requester
- The volume ID.

The output from this EXEC will be the following:

- The volume ID.

Authorization to connect to VGS at all is available through mechanisms associated with intermachine communication. When VGS manages a *local* resource, IUCV ALLOW is required in the CP directory entry of the machine connecting to VGS.

The FSMRMVGA EXEC provides a capability for adding authorization criteria, or for limiting the kinds of requests a specific user machine can make to VGS. For example, FSMRMVGA can be used to authorize certain CMS users to issue the STOP_VGS command.

VGS Programming Error Handling Exit - FSMRMVGE

This exit is called by the VGS program under the following error conditions passing the indicated information:

1. Non-zero reason code from a RMS CSL request; the following shows the layout of the data passed for this error condition:

Columns	Data
1- 4	Reason code
6-13	VSE user ID
15-16	Request type
18-25	VSE library name
27-31	VSE virtual tape address
32-37	Volume ID

2. CPI-C accept error; CPIC_ACP, the conversation ID, and the VSE user ID will be passed as arguments.
3. CPI-C receive error; CPIC_RCV, the conversation ID and the VSE user ID will be passed as arguments.
4. CPI-C send error; CPIC_SND, the conversation ID and the VSE user ID will be passed as arguments.
5. For a CSL call to RMS taking longer than 10 seconds; LONG_CSL and the VSE user ID will be passed as arguments.
6. If an error is encountered form the CP detach command; DET_ERR, the VSE user ID, and the real drive address will be passed as agruments.

Library Configuration File - LIBCONFG LIST

If the VGS resource is to handle library control successfully for more than one 3494, an installation-customized configuration file named LIBCONFG LIST must be present on the VGS machine's A-disk. This file cross-references the up-to-8-character library names used by the VSE guest with the up-to-32-character library names used DFSMS/VM.

Valid records (lines) in this file have two fields, separated by at least one blank character:

1. VSE library name
2. DFSMS/VM library name

Any characters after the blank character following the DFSMS/VM library name are ignored. Any line that begins with a character asterisk (*) is interpreted as a comment. For an example, see Figure 26 on page 136.

```
* Configuration file last updated by Joe on October 31, 1996
L10134   AUTO_LIB_1
L10137   AUTO_LIB_2
L50045   AUTO_LIB_3
* end of configuration data
```

Figure 26. Example of a LIBCONFG LIST File

LIBRCMS Server List - LIBRCMS SRVNAMES

If the VGS resource is to handle library control successfully for multiple VSE guests that issue inventory requests, a file named LIBRCMS SRVNAMES must be present on the secondary Inventory Support machine's 191 disk. This file simply cross-references the up-to-8-character userid of the VSE guest with the up-to-8 character "server-name" specified in the LIBRCMS // SETPARM SRVNAME job-control card on that guest. Refer to “Librarian Server for CMS Users” on page 120 for details on LIBRCMS. This cross-reference file enables the inventory support server to access Librarian files on the correct VSE guest machine.

Valid records (lines) in this file have two fields, separated by at least one blank character:

1. VSE guest userid
2. LIBRCMS server name

Any characters after the blank character following the LIBRCMS server name are ignored. Any line that begins with a character asterisk (*) is interpreted as a comment. For an example, see Figure 27 on page 136.

```
* LIBRCMS xref file last updated by Juan on October 31, 1996
VSE1   LIBRC1
VSE2   LIBRC2
VSE3   LIBRC3
* end of cross-reference data
```

Figure 27. Example of a LIBRCMS SRVNAMES File

Reason Codes

VGS returns a DFSMS/VM reason code to the requester, when the error is detected by DFSMS/VM RMS during the processing of a request. Appendix A, “Return Codes and Reason Codes for CSL Routines,” on page 85 documents these reason codes.

When the failure is detected by the VGS program itself, one of the reason codes in Table 19 on page 137 is reported.

Table 19. Reason Codes Generated by VGS

Reason	Meaning	Explanation/Action
5000	Request not authorized	The authorization user exit has determined that the request is not authorized.
5003	Insufficient data length	The length of the request data was smaller than the minimum required number of bytes. This is an internal program error; contact IBM service.
5008	Invalid request type	An unrecognized request type was specified. This is an internal program error; contact IBM service.
5011	Request to be cancelled not found	A request to cancel a mount request cannot be completed successfully, because the request is not longer in-process on the VGS machine. Issue rewind/unload to queue a dismount operation in the library.
5017	Request cancelled by a cancel-mount request	This request is the target request for a cancel-mount operation. The reason code in the response message for the original mount request. The mount operation was not completed.
5020	Request cancelled by an immediate machine shutdown	This request is purged from the VGS in-process files during immediate shutdown processing of the VGS machine. The mount operation. may have completed successfully, but the status is not known to VGS at the time of shutdown.
5023	Device not found	The device specified for release is not found in the VGS device-use file. No action is taken.
5026	Library not known	The library name specified in a request is not found in the VGS library configuration file LIBCONFG LIST A. Update or correct this file and retry the request.
5028	Device information not obtained	VGS is not able to get information about real tape drives currently attached to the requesting VSE guest. The request cannot be processed. Ensure that the VGS is machine has privilege-class B authorization in its directory record.
5029	Problem with CP ATTACH or DETACH operation	VGS is not able DETACH a device from the requesting guest or to re-ATTACH a guest device to itself for performing the requested function. Ensure that the VGS is machine has privilege-class B authorization in its directory record.
5030	Virtual cuu not provided	A mount, release, cancel, or query device operation has not specified the required virtual cuu field.
5031	Virtual cuu contains invalid characters	A mount, release, cancel, or query device operation does not use valid numeric or hex-digit characters.
5032	Volume label not provided when required	A mount, query volume, cancel,eject, or set volume category operation has not specified a volume serial number.

Table 19. Reason Codes Generated by VGS (continued)

Reason	Meaning	Explanation/Action
5033	Source category not provided when required	A mount from category operation has not specified a source category name.
5034	Target category not provided when required	A set volume category or manage inventory operation has not specified a target category name.
5035	Member name not provided when required	A manage inventory operation has not specified a member name for the source file containing volume category assignments.
5040	Error reading result of Query or Manage Inventory request	The VGLIBSRV machine was not able to read the request final status information from the secondary inventory support machine's 191 disk. The result of the inventory operation is unknown.
5041	Timeout occurred during Query or Manage Inventory request	The VGLIBSRV machine did not receive final status on the request within the time limit specified by the INV_MAX_TIME customziation variable. The request may still complete successfully, or the secondary machine may have encountered an error. Check the status of the secondary inventory support machine.
5042	Secondary inventory server is not logged on	The VGS machine has determined the there the secondary inventory support server is not logged on, or its logon status is cannot be obtained. The inventory request cannot be processed.
5043	Secondary inventory server's 191 is not linked by VGLIBSRV	Inventory requests cannot be processed by VGS unless the secondary support server is defined, logged on, and it's 191 disk is linked by VGLIBSRV.
5044	Secondary inventory server is not identified to VGLIBSRV	The customiztion variable HELPER_ID in the customization exec does not specify a valid server name. Inventory requests cannot be processed.
5080	Locked failed, Librarian member already locked	The Librarian member specified in a Manage Inventory request or implied by category inventory naming conventions for a Query Inventory request is already locked. Member is locked by another user, wait and resubmit the request.
5081	Lock failed, fully-qualified Librarian member is not valid	The Librarian member specified in a Manage Inventory request could not be locked and most likely does not exist. Ensure that the Library and Sublibrary names specified in the customization exec are valid and that the specified member already exists in the sublibrary.
5082	Unspecified lock failure for Librarian member	The Librarian member specified in an Inventory request could not be locked for some unknown reason. This is most likely an internal error.

Table 19. Reason Codes Generated by VGS (continued)

Reason	Meaning	Explanation/Action
5083	Librarian write failure	The Librarian member specified in an Inventory request could was not successfully written in the Librarian file in the VSE guest machine.
5084	RMS error on Query Library Inventory command	An unspecified error occurred requesting the inventory report from the RMS machine. Retry the request. If the problem persists, logon to the secondary inventory support machine and check the console during request submission, and/or the RMS Master console.
5085	Invalid source category specified for Query Inventory	The source category requested for inventory in not supported or is not a valid category name.
5086	LIBRCMS server name not found for the requesting VSE guest	When multiple VSE guests are supported by VGS, a LIBRCMS server cross-reference file on the secondary inventory support machine 191 disk must provide the name of the LIBRCMS server for each VSE guest. This no server was found for the requesting VSE guest.
5090	Secondary inventory support internal error - Erase	The CMS ERASE command failed for a file in the inventory support server's 191 disk.
5091	Secondary inventory support internal error - Disk I/O	The EXECIO command failed for a file in the inventory support server's 191 disk. Check for disk-full condition.
5092	Secondary inventory support internal error - Copyfile	The CMS COPYFILE command failed for a file in the inventory support server's 191 disk. Check for disk-full condition.
5093	Secondary inventory support internal error - Query Reader	The Query Reader command failed checking the reader for returned inventory report.
5094	Secondary inventory support internal error - Receive	The Receive command failed reading the report file from the reader to the 191 disk. Check for disk-full condition.
5095	Secondary inventory support internal error - Unexpected disk files	At least one previous inventory report was detected on the secondary inventory server's 191 disk. The integrity of inventory data cannot be guaranteed. Report is not updated in the VSE Librarian file.
5096	Secondary inventory support internal error - Mismatch	The name of the category inventoried on the RMS inventory report does not match the source category name in this request. The report is not updated in the VSE Librarian file.

PI end

Appendix E. RSM Copy Export Support

This appendix describes how to use the COPY_EXPORT support with the Removable Media Service (RMS) component of DFSMS/VM.

APAR VM65789 provides support within the Removable Media Services (RMS) component of DFSMS/VM for the IBM Virtualization Engine TS7700 Copy Export functionality.

COPY_EXPORT allows a copy of selected logical volumes written to the TS7700 to be removed and taken offsite for disaster recovery purposes. The benefits of volume stacking, which places many logical volumes on a physical volume, are retained with this function. In addition, since the data being exported is a copy of the logical volume, the logical volume data remains accessible by the production host systems.

The basic steps for using copy export are:

1. Determine the data that is needed for offsite disaster recovery purposes. Through the automatic class selection routines, assign a management class that specifies a secondary volume pool to any data that is to be copy exported. As logical volumes are written to the TS7700 and then closed, they are scheduled to be copied to a physical tape volume. With a management class that specifies a secondary copy pool, the data is copied both to a primary pool physical volume and a secondary pool physical volume.
2. When it is time to remove the physical volumes that have a copy of the data on them, first write an Export List File volume to the TS7700 that is to perform the copy export operation. The export list file volume contains instructions for the copy export operation, one of the key instructions being the number of the physical volume pool to export. The export list file volume will also hold a status file that contains records that are generated by the TS7700 as part of the copy export operation. Second, once the export list file volume has been written and closed, issue the Library Export command.
3. In processing a copy export operation, the TS7700 first copies any logical volumes assigned to the pool being exported that only have a copy of the logical volume in the cache. Next, each of the physical volumes in the specified pool that contains active data is mounted and a database backup of the TS7700 is written to them. Status records are written to the export list file volume and then the physical volume is ejected from the library. Once all the physical volumes in the pool have been processed, the copy export operation is complete.

   On the completion of the copy export operation, a summary status console message is generated. The status file on the export list file volume contains detailed information about the execution of the copy export operation and should be retained for future reference.
4. An operator removes the physical volumes from the I/O station of the library and prepares them for shipment to an offsite location.

A new target category, COPY_EXPORT, has been added to the existing RMS SET VOLCAT command and CSL to initiate a Copy Export operation to the TS7700. A source category is not allowed on the SET VOLCAT command when the target category is COPY_EXPORT.

In addition, a new RMS utility, FSMCPYEX, located on the DFSMSRM 1B5 minidisk, has been provided with APAR VM65789 to assist with building the Export List File Volume previously discussed. The FSMCPYEX EXEC is an interactive front end for copy export processing.

FMSCPYEX performs all necessary mount and demount requests as well as initiate the asynchronous start of the SET VOLCAT TARGET COPY_EXPORT command. After completion of the copy export, it can be used to retrieve the logged information from the used export file list volume.

FSMCPYEX requests the following information:

1. The 6-digit volume number of the export file list volume
2. Whether to start a new copy export process or create a status file from a previously completed Copy Export request.

3. The 2-digit pool number to be exported

Prerequisites for issuing the FSMCPYEX exec:

1. The User ID must have DFSMSRM authorization
2. The TS7700 tape library configured for copy export
3. Range of volumes configured for a secondary (copy export) pool
4. One volume for the export file list that is not part of this pool
5. Free VDEV 0181

RMS Utility for Copy Export Processing

```
►► FSMCPYEX ►◄
```

Input parameters: None.

Results: Depending on the requested actions, a new copy export is started asynchronously or the status information from an export file list volume of a finished copy export is written to CPYEXP STATUS A.

Messages with Return Codes:

RC0
FSMCPE02000 Copy Export Processing has been started asynchronously, error and completion messages will be written to your console

RC4
FSMCPE02100W EXPORT STATUS seems to be malformed!

RC8
FSMCPE02200E Device 181 already defined

RC8
FSMCPE02201E Copy Export aborted, no data has been changed

RC8
FSMCPE02202E Volume number does not consist of 6 characters

RC8
FSMCPE02203E Pool number does not consist of 2 characters

RC8
FSMCPE0220nE An error occurred while processing CSL call XXXXXX

CSL
return code = X

CSL
reason code = XXX

Function
return code = XXX

Function
reason code = XXX

See Appendix A, “Return Codes and Reason Codes for CSL Routines,” on page 85 for further information.

Appendix F. Common Problems and Solutions

This appendix contains the following common problems that one may experience along with solutions to these problems:

- Problem where 3494/3495 devices may not have been available when CP was IPLed, requiring deleting and redefining RDEVS
- Problem caused by incorrectly defining a STDEVOPT directory statement when running a second level guest machine

Deleting and Redefining RDEVS

You may receive an error that a specified subsystem does not exist. The typical reason for getting this error message is that the subsystem was not available at startup time, and the CP dynamic IO block was built without the devices being available. The following sample shows the error messages received when a subsystem does not exist:

```
FSMRMBLC===> DIAG 254 IS AVAILABLE
FSMBBC0505E INTERNAL ERROR FROM ROUTINE 2523, RETURN CODE = 3
FSMBBC2028E THE SPECIFIED SUBSYSTEM 70171 DOES NOT EXIST IN VM REAL IO
            CONFIG
FSMSMS3009E ERRORS HAVE CAUSED DFSMS INITIALIZATION TO STOP
```

When this condition occurs, the devices need to be deleted from the dynamic IO configuration and then re-added in order for DFSMS to obtain the device characteristics. The following example shows the commands to use to correct this problem, with the device range from 980 to 99F:

```
vary offline 980-99F
vary offline subchannel 980-99F
cp delete rdev 980-99F
set rdev 980-99F type tape
vary online 980-99F
```

Ensure to VARY OFF and CLEAR all devices before adding any back, as the rebuild will not start unless all the devices are first deleted.

Incorrectly Defining a STDEVOPT Directory Statement

When running DFSMS on a second level machine and a STDEVOPT directory statement is not defined correctly, you will get the following error message:

```
FSMBAC2006E Library I/O error; reason code = 3740, request identifier =
device = 3490, library =
FSMBAC2019E Sense Data =
C0404000000000200000000000000000000000E900000000C73F409534320000
TAPE 3490 DETACHED BY RMSMASTR
FSMBBD2241W Device 3490 could not be initialized
```

To correct this problem, be sure to include the statement

```
'STDEVOPT LIBRARY CTL'
```

in both the first and second level user's directory entry. See "Authorizing Your RMS Master to Interact with the 3494 or 3495" on page 9 for further details.

Volume Errors in Bulk Processing

When bulk processing a range of volumes, you may receive error messages for volumes outside what you thought were in the range you requested. For example, for the command:

```
   SET VOLCAT BULK 10000-10200
```

you might see error messages:

```
FSMBAD2009E 1001A Requested Volume not in Library
FSMBCV2157W A failure occurred attempting to assign volume 1001A
            to category SCRATCH1
```

Remember that when specifying a range for a bulk insert, the range will include A-Z, then 0-9. For example, if you start at 10000, RMS will process volumes in the range 10000-10009, and then try processing 1001A-1001Z before processing 10010-10019.

You may wish to specify specific ranges in the bulk processing file to exclude attempts to process unwanted tape ranges.

FSMTOS510E Error Message Received

If you receive the error message:

```
FSMTOS0510E Internal error, return code = 4, reason code=41
```

Clear out the contents of the VMSYSU:DFSMS.WORK. directory.

Ejecting a Logical Volume from a Virtual Tape Subsystem ATL

The following error may occur when trying to eject a logical volume from a Virtual Tape Subsystem (VTS) ATL:

```
FSMBAC2003E Request cancelled because the subsystem is functionally incompatible;
request identifier = 'request_identifier'
FSMBAC2019E Sense Data = C041A0290000002306000000004040404040400000000DD684CE1F0520F0EB0F00
```

To avoid this error, ensure that the volume is assigned to a fast ready category before attempting to eject it.

Notices

Programming Interface Information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of z/VM.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

PI

<...Programming Interface information...>

PI end

Trademarks

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Index

Product Number: 5741-A09

Printed in USA

SC24-6278-74